MB FAKULTÄT FÜR MASCHINENBAU

Otto-von-Guericke-Universität Magdeburg

Fakultät für Maschinenbau

Institut für Mobile Systeme

Bachelorarbeit

von

Magnus Hanses

Steuerungsarchitektur für die Bahnführung eines mobilen Roboters innerhalb eines Produktionssystems

Erstprüfer:

apl. Prof. Dr.-Ing. habil Arndt Lüder

Zweitprüfer:

Prof. Dr.-Ing. Roland Kasper

Betreuer:

apl. Prof. Dr.-Ing. habil Arndt Lüder

Dipl.-Ing. Stephan Schmidt

15. Juli 2012

Vorwort

Diese Bachelorarbeit entstand im Rahmen der Projektgruppe robOTTO an der Otto-von-Guericke-Universität Magdeburg. Ich möchte allen danken, die zu ihrem Gelingen beigetragen haben.

Besonderer Dank gilt apl. Prof. Dr.-Ing. habil Arndt Lüder und Dipl.-Ing. Stephan Schmidt für die Betreuung der Arbeit und ihrer Begleitung durch Hinweise und Anregungen.

Weiterhin gilt mein Dank Erik Sommer und Josephine Möller für ihre Unterstützung und Ratschläge. Prof. Dr.-Ing. Roland Kasper danke ich für die Erstellung des Zweitgutachtens.

Kurzfassung

Um den Materialfluss in modernen Produktionssystemen sicherzustellen, kommen im zunehmenden Maße autonome Transportsysteme zum Einsatz. Deren Realisierung erfordert eine dynamische Bahnplanung und effiziente Umsetzung der Bewegungen.

Die vorliegende Arbeit behandelt die Entwicklung einer Methode zur Bahnplanung und Bahnführung eines omnidirektionalen Roboters. Mit Hilfe von natürlichen kubischen Splines wird aus diskreten Wegpunkten eine kontinuierliche Bahn erzeugt. Ziel ist es den Roboter mit hoher Genauigkeit entlang dieser Bahn zu bewegen. Dazu werden unter Berücksichtigung der Kinematik und Dynamik der verwendeten Roboterplattform Regler entworfen, die dies ermöglichen. Zusätzlich wird durch Anwendung von Geschwindigkeitsprofilen ein weiches Anfahren und Abbremsen garantiert.

Abstract

In modern production systems autonomous guided vehicles are deployed to ensure the flow of material. To realize such autonomous systems, dynamic path planning and efficient motion control are required.

This thesis deals with the development of methods for path planning and motion control for omni-directional robots. Using natural cubic splines a continuous path reconstructing a set of discrete waypoints can be generated. A control system will be developed considering kinematics and dynamics of the robot to ensure precise movement along this path. In addition acceleration and deceleration is realized through velocity profiles to achieve smooth and stable motion.

Erklärung der Selbstständigkeit

Hiermit versichere ich, die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie die Zitate deutlich kenntlich gemacht zu haben.

Magdeburg, den 15. Oktober 2012

Magnus Hanses

Inhaltsverzeichnis

Abbildungsverzeichnis

Symbolverzeichnis

Symbol	Bedeutung	Einheit
a	Absolutglied des Polynoms	1
a_m	Vorgabe der Roboterbeschleunigung	$\mathrm{m/s}$
b	lineares Glied des Polynoms	1
c	quadratisches Glied des Polynoms	1
c_e	Kopplungskonstante der Elektrik	1
c_m	Kopplungskonstante der Mechanik	1
c_r	Reibkonstante	1
$C(s)$	charakteristisches Polynom	1
d	kubisches Glied des Polynoms	1
$D(s)$	Übertragungsfunktion der Dynamik	1
e_a	Regelabweichung des äußeren Regelkreises	1
e_i	Regelabweichung des inneren Regelkreises	1
f	Vortrieb des Rads	N
${}^rF_x$	Vortrieb in x-Richtung des Roboters	N
${}^rF_y$	Vortrieb in y-Richtung des Roboters	N
g	Substitution (Gl. 3.19)	m
$G(s)$	Übertragungsfunktion der Regelstrecke	1
h	Substitution (Gl. 3.11)	m
$\mathbf{H}$	Trägheitsmatrix	1
i	Übersetzungsverhältnis	1
I	Motorstrom	A
j	Zählvariable	1
J_{AS}	Trägheitsmoment des Antriebsstrangs	$\mathrm{kgm^2}$
J_{Motor}	Trägheitsmoment des Motors	$\mathrm{kgm^2}$
J_r	Trägheitsmoment des Roboters	$\mathrm{kgm^2}$
J_{Rad}	Trägheitsmoment des Rads	$\mathrm{kgm^2}$
k	äquidistanter Parameter	1
k_d	Verstärkung der proportionalen Regelabweichung	1
k_i	Verstärkung der integralen Regelabweichung	1

Symbol	Bedeutung	Einheit
k_p	Verstärkung der differentiellen Regelabweichung	1
$K(s)$	Übertragungsfunktion des Reglers	1
l	Länge der Bahn	m
L	Induktivität	H
$M_{Abtrieb}$	Abtriebsmoment	Nm
$M_{Antrieb}$	Antriebsmoment	Nm
m_r	Masse des Robotino®	kg
M_P	Moment um den Mittelpunkt des Robotino®	Nm
M_{Reib}	Reibmoment	Nm
n	Anzahl	1
O	Ursprung des Weltkoordinatensystems	$[\,\mathrm{m}\;\mathrm{m}]$
p	Radabstand zum Mittelpunkt	m
P	Ursprung des Roboterkoordinatensystems	$[\,\mathrm{m}\;\mathrm{m}]$
$\dot{q}$	translatorische Radgeschwindigkeit	$\mathrm{m/s}$
r	Radius	m
R	elektrischer Widerstand	Ω
$\mathbf{R}$	Rotationsmatrix	1
$\vec{r}$	Ortsvektor	$[\,\mathrm{m}\;\mathrm{m}]$
s	Strecke	m
s_e	Gesamtstrecke	m
S	natürlicher kubischer Spline	m
$\mathbf{T}$	Transformationsmatrix	1
t	Zeit	s
T	Zeitkonstante	s
t_b	Beschleunigungszeit	s
t_e	Fahrtzeit	s
t_v	Verzögerungszeit	s
U	Motorspannung	V
U_{ind}	induzierte Gegenspannung	V
v	Robotergeschwindigkeit	$\mathrm{m/s}$
v_m	Vorgabe der Robotergeschwindigkeit	$\mathrm{m/s}$
v_{max}	maximal erreichbare Robotergeschwindigkeit	$\mathrm{m/s}$
${}^{r}x$	x-Achse des Roboterkoordinatensystems	m
${}^{w}x$	x-Achse des Weltkoordinatensystems	m
${}^{r}y$	y-Achse des Roboterkoordinatensystems	m
${}^{w}y$	y-Achse des Weltkoordinatensystems	m
ω	Motorgeschwindigkeit	$\mathrm{rad/s}$

Symbol	Bedeutung	Einheit
φ	Winkeldifferenz der Koordinatensysteme	rad
θ	Orientierung der Räder	rad
${}^{r}\xi$	Pose des Roboters in Roboterkoordinaten	$[\,\mathrm{m}\ \mathrm{m}\ \mathrm{rad}]$
${}^{w}\xi$	Pose des Roboters in Weltkoordinaten	$[\,\mathrm{m}\ \mathrm{m}\ \mathrm{rad}]$

1 Einleitung

1.1 Motivation

Moderne Produktionsbetriebe streben aus Gründen der Wettbewerbsfähigkeit kurze Durchlaufzeiten, geringe Bestände und hohe Flexibilität an [Sch01]. Die Umsetzung dieser Ziele erfordert eine hohe Dynamik des Materialflusses. Dabei sind eine optimale Auslastung der Stationen und eine hohe Gesamtverfügbarkeit zu gewährleisten [FW06]. Ein wichtiger Prozess im Materialfluss ist das Transportieren, also die zielgerichtete Ortsveränderung von Gütern. Dieses Anwendungsfeld wird momentan von fahrerlosen Transportsystemen (FTS) dominiert. Die Notwendigkeit nach hoher Flexibilität erfordert jedoch zunehmend den Einsatz von autonomen mobilen Robotern. Deren Integration in die hochdynamische Umgebung einer Produktionsstätte gilt es voranzutreiben.

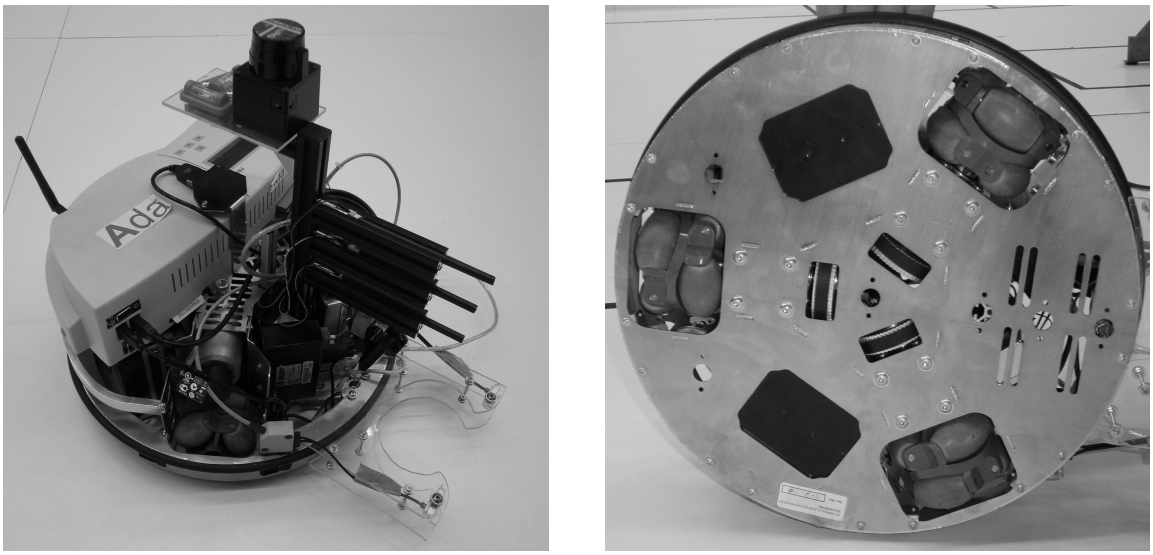
Dieses Ziel wird von der Logistics Leaque sponsered by Festo (LLSF) im Rahmen des RoboCup verfolgt. Unter den Teilnehmern befindet sich auch das seit 2009 an der Otto-von-Guericke-Universität bestehende Team robOTTO. Dabei handelt es sich um eine Initiative von Studenten aus verschiedenen Fakultäten. Eine bestehende Herausforderung der LLSF ist die dynamische Bahnplanung innerhalb einer bekannten Produktionsstätte und deren effiziente Umsetzung. Ziel dieser Arbeit ist es, Algorithmen zu entwickeln, mit deren Hilfe die Lösung dieser Aufgabe ermöglicht wird.

1.2 Robotino®

Als Roboterplattform dient der von der Firma Festo Didactics entwickelte Robotino®. Dieses mobile Robotersystem wird aufgrund seiner einfachen Handhabung und Erweiterbarkeit für die Ausbildung an Berufs- und Hochschulen verwendet. Des Weiteren dient der Robotino® als Standardplattform für die seit 2012 offizielle und im Rahmen des RoboCup ausgetragene Logistics League. Diese wird in Kapitel 1.3 genauer beschrieben.

1.2.1 Hardware

Die Basis des Robotino® (Abb. 1.1a) bildet ein rundes Edelstahl-Chassis mit einem Durchmesser von $370\ \mathrm{mm}$. Als Antrieb dienen drei separat angesteuerte, um jeweils $120°$ verdrehte, Allseitenräder (Abb. 1.1b). Durch die Anordnung erreicht der Robotino® die maximal möglichen Freiheitsgrade in der Ebene. Es handelt sich also um ein holonomes System.



(a) Robotino® **(b)** Antriebseinheiten

Abbildung 1.1: Ansichten des Robotino®

Die Versorgung wird über zwei $12\ \mathrm{V}$ Blei-Gel-Akkumulatoren mit einer Gesamtkapazität von $5\ \mathrm{Ah}$ sichergestellt. Die Steuereinheit des Robotino® bildet ein PC/104-System. Dieses verfügt über eine Rechenleistung von $500\ \mathrm{MHz}$ und greift als Betriebssystem auf Ubuntu Linux mit einem auf Echtzeitverhalten optimierten Kernel zurück. Das Betriebssystem befindet sich auf einer austauschbaren CF-Karte mit einer Speicherkapazität von bis zu $4\ \mathrm{GB}$. Die Ansteuerung der Motoren bzw. Auswertung der Sensorik wird über externe Boards realisiert, die mit dem PC/104-System kommunizieren. Für die Ein- und Ausgabe stehen diverse Schnittstellen zur Verfügung.

- USB
- Ethernet
- VGA
- acht digitale und acht analoge Eingänge
- acht digitale Ausgänge

1.2.2 Sensorik

In der Grundausstattung verfügt der Robotino® über folgende Sensorik:

- Inkrementalgeber zur Messung der Motordrehzahl
- Gyroskop zur Wahrnehmung rotatorischer Bewegungen
- neun Infrarotsensoren zur Abstandsmessung
- ein Induktivsensor zur Erkennung metallischer Markierungen
- zwei optische Sensoren zur Erkennung farblicher Markierungen
- ein Kontaktsensor zur Kollisionserkennung
- eine Webcam für die visuelle Auswertung

Diese solide Basis wurde um einen Laserscanner erweitert (siehe Abbildung 1.1a). Er sitzt auf dem Robotino® und dient der globalen Lokalisierung. Des Weiteren werden Photodioden genutzt, um binäre Informationen in Form von Lichtsignalen auszuwerten. Als Halter für die zusätzliche Sensorik dient ein Gerüst aus Aluminiumprofilen.

1.2.3 Programmierung

Der Robotino® kann sowohl grafisch als auch textuell programmiert werden. Folgende Programmiersprachen werden unterstützt:

grafische Programmiersprachen

- Labview
- Simulink
- Robotino®-View

textuelle Programmiersprachen

- C/C++
- C#
- .Net und Java
- Matlab

Dabei ist es möglich, die erstellten Programme sowohl direkt auf dem Robotino® als auch aus der Ferne mit Hilfe des integrierten WLAN-Client auszuführen.

1.3 Logistics League sponsored by Festo

Die Logistics League sponsored by Festo (LLSF) ist ein seit 2012 im Rahmen des RoboCup eingeführter Wettbewerb mit dem Fokus, industrielle Prozesse abzubilden. Ziel ist es, innerhalb einer fiktiven Produktionsstätte mit Hilfe von drei autonomen Robotern einen maximalen Produktionsoutput zu erreichen. Dieser definiert sich durch das Ausliefern von Produkten, die aus Rohstoffen über Zwischenstufen hergestellt werden müssen. Um diesen Ablauf zu optimieren, müssen die Roboter kollaborieren. Dies ist eine der größten Herausforderungen an die teilnehmenden Teams und stellt somit den Forschungsschwerpunkt der LLSF dar. Um ein Spiel gewinnen zu können, müssen innerhalb von 15 Minuten so viele Punkte wie möglich erzielt werden. Punkte werden für das Erreichen von Produktionsschritten vergeben. Die genaue Wertigkeit dieser Schritte ist im offiziellen Regelwerk der LLSF definiert [Did12]. In einem Spiel treten zwei Teams zeitgleich auf identischen Spielfeldern gegeneinander an. Als Roboterplattform dient dabei der in Kapitel 1.2 vorgestellte Robotino®.

1.3.1 Spielfeld

Die LLSF wird auf einem quadratischen Spielfeld (Abb. 1.2) mit einer Seitenlänge von $5625\ \mathrm{mm}$ ausgetragen. Umgeben ist das Spielfeld von $500\ \mathrm{mm}$ hohen HPL-Platten. Diese begrenzen die Produktionsstätte und definieren somit den Arbeitsraum des Roboters. Auf dem Spielfeld befinden sich neben den Robotern Pucks, Maschinen und Markierungen.

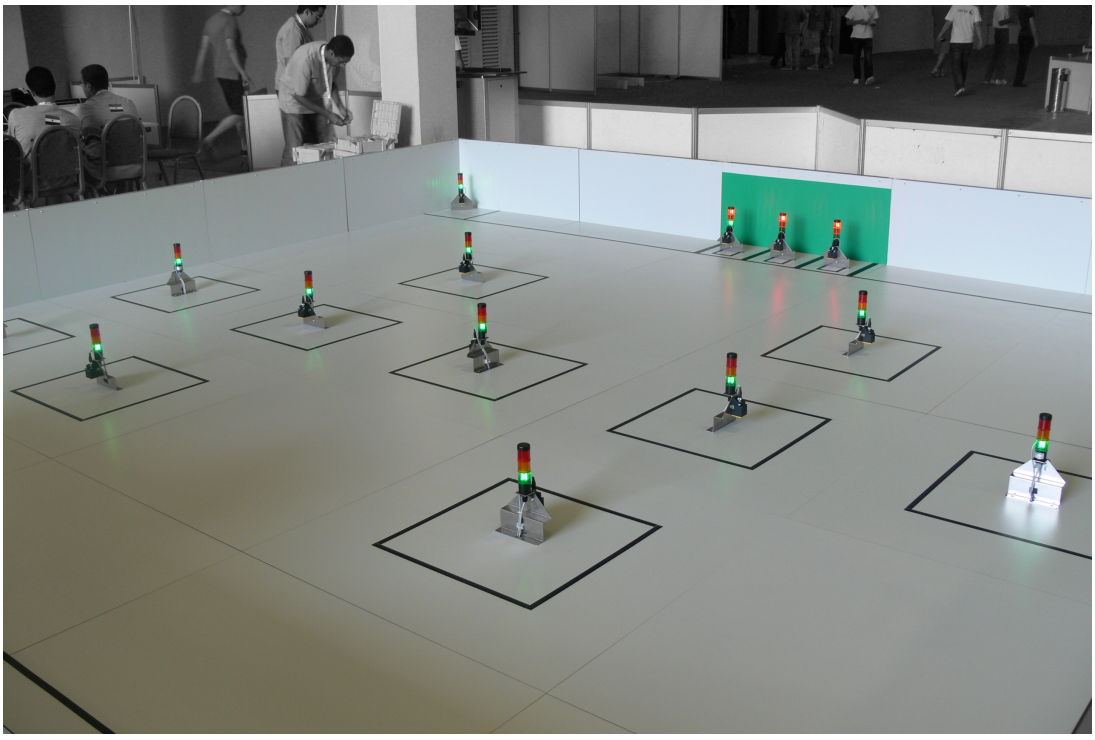


Abbildung 1.2: Spielfeld der LLSF

Pucks

Die Hockeypucks (Abb. 1.3) dienen als Transportplattform. Mit Hilfe der Schiebevorrichtung kann der Robotino® sie auf dem Spielfeld bewegen. Der Puck wird als Palette betrachtet. Auf den Paletten liegen Güter, die von den Maschinen benötigt werden. In einem auf dem Puck befindlichen RFID-Chip wird gespeichert, welche Güter auf der Palette liegen oder ob eine Palette leer ist. Folgende Möglichkeiten kommen infrage:

- leere Palette (LP)
- Rohstoffe (S0)
- Zwischenprodukt 1 (S1)
- Zwischenprodukt 2 (S2)
- Endprodukt (P)

Abbildung 1.3: Hockeypuck mit RFID-Chip

Maschinen

Die Maschinen (Abb. 1.4) sind feste Bestandteile des Spielfeldes. Sie werden mit einem abgewinkelten Edelstahlblech, einer Signal-Ampel und einem RFID-Lese- und Schreibgerät simuliert. Auf dem Spielfeld befinden sich insgesamt 16 Maschinen mit unterschiedlichen Aufgaben. Es kann folgende Unterteilung vorgenommen werden:

- drei Arten von Produktionsmaschinen
  - Typ 1 (M1)
  - Typ 2 (M2)
  - Typ 3 (M3)
- drei Auslieferungsmaschinen (MA)
- zwei Recyclingmaschinen (MR)
- eine Lesemaschine (ML)

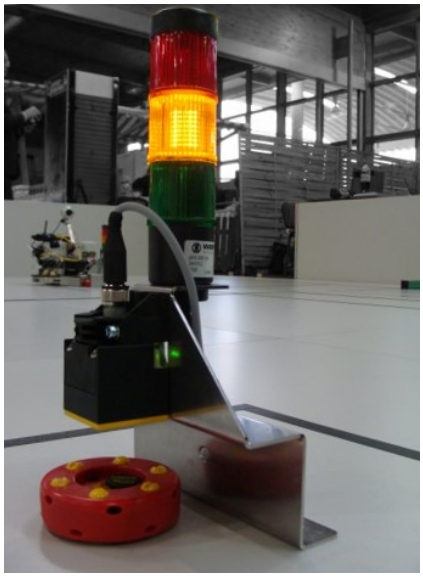


Abbildung 1.4: Ampel mit RFID-Unit

Produktionsmaschinen sind in der Lage, Zwischen- bzw. Endprodukte herzustellen. Dazu benötigen sie Güter, also Pucks mit entsprechenden Informationen. Diese werden mit Hilfe des RFID-Lesegerätes ausgewertet. Die Signalampel dient dazu, den Status der Maschine zu übermitteln. Farblich kodiert kann somit der Produktionsfortschritt, sowie der Produktionserfolg bzw. -misserfolg mitgeteilt werden. Nach einem erfolgreichen Produktionsschritt wird der RFID-Chip mit einer neuen Information beschrieben. Zu Beginn eines Spiels ist nur die Position der Produktionsmaschinen bekannt, nicht jedoch ihre Orientierung oder ihr Typ. Auslieferungsmaschinen dienen dazu, ein fertiges Produkt entgegenzunehmen und somit den Produktionsprozess abzuschließen. Zu jedem Zeitpunkt ist nur eine Auslieferungsmaschine aktiv und nutzbar. Dieser Zustand wird mit Hilfe der Signalampel kommuniziert. Recylingmaschinen werden benötigt, um leere Paletten wieder mit Rohstoffen zu belegen. Für den Produktionsprozess nicht erforderlich ist die Lesemaschine. Sie dient lediglich dazu, den unter ihr liegenden Puck zu identifizieren.

Markierungen

Die verschiedenen Bereiche des Spielfeldes werden mit Hilfe von schwarzen Linien voneinander abgegrenzt. Es wird zwischen Wareneingang, Warenausgang und Maschinenbereichen unterschieden. Der Wareneingang beherbergt eine definierte Anzahl an Pucks und befindet sich am Startpunkt des Spielfeldes. Der Warenausgang befindet sich direkt gegenüber und umrandet die Auslieferungsmaschinen. Um diese Bereiche besser auseinanderhalten zu können, wurden sie zusätzlich mit den Farben grün und blau kodiert. Der Maschinenbereich dient dazu, an dem Produktionsschritt beteiligte Pucks einer Maschine zuzuordnen. Wird also ein Puck außerhalb dieses Bereichs abgelegt, steht er der Maschine nicht zur Verfügung.

1.3.2 Produktionsablauf

Zu Beginn eines Spiels befinden sich die drei Roboter hinter der Startlinie, direkt neben dem Wareneingang. Dort lagern Paletten mit Rohprodukten, die mit Hilfe der Produktionsmaschinen zu Endprodukten verarbeitet werden müssen. Abbildung 1.5 zeigt die Syntaktik des Produktionsablaufs.

Die Herstellung eines Produktes erfordert drei Produktionsschritte. Jeder dieser Schritte kann genau einem Maschinentyp (M1 bis M3) zugeordnet werden. Dabei werden fortlaufend Rohprodukte (S0) und die Erzeugnisse der vorangegangenen Schritte (S1,S2) weiterverarbeitet. Werden für den Produktionsprozess einer Maschine mehrere Paletten benötigt, so wird die zuletzt angelieferte mit dem Produkt belegt und die restlichen Paletten werden leer hinterlassen. Leere Paletten (LP) müssen zur weiteren Nutzung an den Recyclingmaschinen (MR) wieder mit Rohprodukten belegt werden. Nach Erhalt eines Endproduktes (P) muss dieses in den Warenausgang (MA) transportiert werden, um den Produktionsprozess abzuschließen.

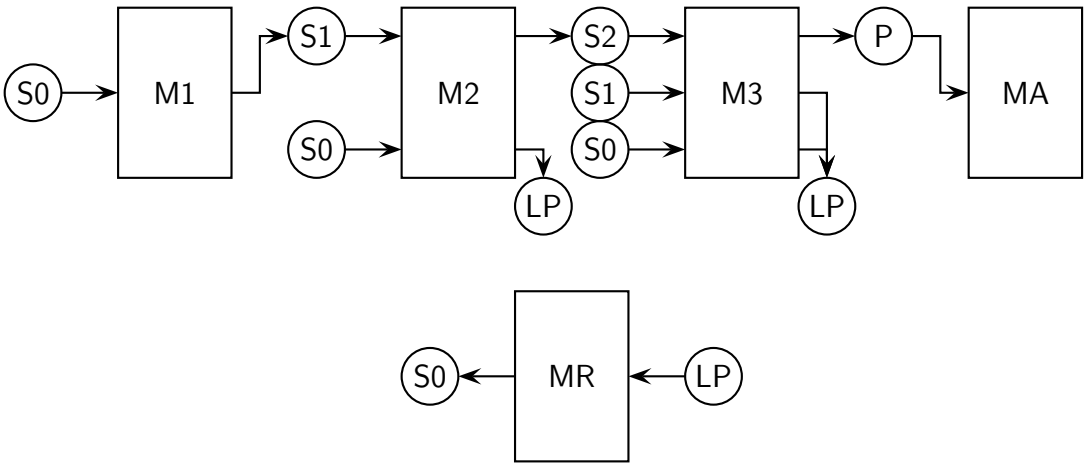


Abbildung 1.5: Schema des logistischen Ablaufes [Som12]

1.4 Zielstellung

Die bisherige Herausforderung der LLSF bestand in der Implementierung grundlegender Algorithmen. Ziel war es, den Produktionsablauf mit Hilfe des Robotino® zu realisieren. Dabei stand die Erfüllung der Aufgabe im Vordergrund, nicht jedoch ihre optimale Umsetzung.

Eine der wichtigsten Aufgaben ist das kollisionsfreie Transportieren von Gütern. Dazu muss eine Bahnkurve geplant und mit hoher Genauigkeit umgesetzt werden. Weiterhin muss der Transport zeitlich vorhersehbar sein, um eine optimale Auslastung der Maschinen garantieren zu können.

Der bisherige Algorithmus approximiert die zu fahrende Bahnkurve durch mehrere Geraden. Dies hat zur Folge, dass der Roboter in jedem Schnittpunkt anhalten, sich ausrichten und erneut anfahren muss. Dadurch werden hohe Transportzeiten verursacht. Weiterhin ist das ungeregelte Anfahren und Abbremsen für Schlupf verantwortlich und schafft somit eine unnötige Fehlerquelle. Eine Vorhersage der Transportzeiten ist bisher noch nicht möglich.

Um weiterhin konkurrenzfähig an der LLSF teilzunehmen, müssen diese Defizite beseitigt werden. Ziel dieser Arbeit ist es, den bestehenden Algorithmus hinsichtlich seiner Transparenz, Transportzeit und Bahngenauigkeit zu überarbeiten. Dabei müssen Anforderungen der LLSF, der Roboterplattform und des Arbeitsraumes berücksichtigt werden.

1.5 Aufbau der Arbeit

Die vorliegende Bachelorarbeit unterteilt sich in mehrere Kapitel. Im Kapitel 2 wird die Roboterplattform mathematisch beschrieben. Dabei werden die Kinematik und die Dynamik des Systems betrachtet. Dies bildet die Grundlage zur Entwicklung der in Kapitel 3 und 4 vorgestellten Algorithmen zur Bahnplanung und Bahnführung. Jedes dieser Kapitel enthält eine kurze Einleitung. In dieser wird an die Aufgabenstellung herangeführt und Voraussetzungen für die erfolgreiche Umsetzung besprochen. Die Implementierung der Algorithmen wird in Kapitel 5 betrachtet. Die bei deren Evaluierung erzielten Ergebnisse werden in Kapitel 6 vorgestellt. Abgeschlossen wird die Arbeit durch das Kapitel 7, das die vorgenommenen Verfahren zusammenfasst und einen Ausblick auf zukünftige gibt.

2 Kinematische und dynamische Modellierung

2.1 Einleitung

Der Robotino® wird als ein mechatronisches System betrachtet. Darunter wird allgemein die Verknüpfung von vorwiegend mechanischen mit elektrischen Systemen verstanden. Als Schnittstelle dient eine Informationsverarbeitung, die anhand von Sensordaten die Aktorik des Systems ansteuert. Die gezielte Ansteuerung erfordert eine Systembeschreibung mit Hilfe eines äquivalenten mathematischen Modells. Um dies zu erstellen, wird die Kinematik und Dynamik des Robotino® untersucht.

Bei der kinematischen Modellierung wird die Bewegung des Roboters, ohne Hinsicht auf Kräfte und Drehmomente, die diese verursachen, betrachtet. Dies ermöglicht eine rein geometrische Beschreibung und dient der räumlichen Zuordnung der Bewegungsachsen. Als Hilfsmittel der Modellierung stehen dabei die Pose, also die Position und Orientierung im Raum, sowie die Geometrie des Roboters und seiner Mechanismen zur Verfügung. Mit Hilfe dieser wird die inverse und direkte Kinematik entwickelt. Die direkte Kinematik bestimmt anhand der Motorumdrehungen die Pose des Roboters und stellt damit das Gegenstück zur inversen Kinematik dar [Lü12].

Die dynamische Modellierung entfernt sich von der reinen Bewegungslehre. Es werden die für die Bewegung verantwortlichen Kräfte und Momente betrachtet. Diese werden durch drei Gleichstrommotoren erzeugt. Ziel der dynamischen Modellierung ist es, anhand der manipulierbaren Eingangsgrößen den zeitlichen Verlauf der Motorbewegung zu ermitteln. Mit Hilfe der kinematischen Modellierung kann dieser Verlauf in die Roboterbewegung überführt werden.

2.2 Kinematische Modellierung

Das kinematische Modell wird durch Abarbeitung folgender Schritte entwickelt:

1. geometrische Betrachtung des Robotino®
2. Definition der Pose
3. Entwicklung einer Drehmatrix zwischen Roboter- und Weltkoordinaten
4. Entwicklung einer Transformationsmatrix zwischen Motor- und Roboterkoordinaten

Der Robotino® besitzt drei zueinander um $120°$ verdrehte Antriebseinheiten. Diese bestehen jeweils aus einem Allseitenrad mit einem Radius r von $40\ \mathrm{mm}$ und einem Gleichstrommotor. Sie sind über ein zweistufiges Getriebe miteinander verbunden. Eine schematische Zeichnung kann Abbildung 2.1 entnommen werden.

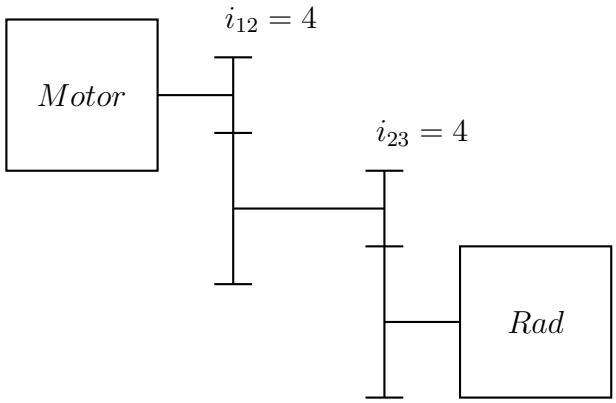


Abbildung 2.1: Getriebebild

Es handelt sich um eine Übersetzung ins Langsame. Das Verhältnis der Übersetzung i bestimmt sich aus der Übersetzung der einzelnen Stufen.

$$i = i_{13} = i_{12} \cdot i_{23} = 16 \tag{2.1}$$

Um die Pose des Roboters zu definieren, werden zwei Koordinatensysteme eingeführt. Das ortsfeste Weltkoordinatensystem $({}^{w}x, {}^{w}y)$ mit einem frei wählbaren Ursprung O und das mobile Roboterkoordinatensystem $({}^{r}x, {}^{r}y)$ mit seinem Ursprung im Mittelpunkt P des Roboters (Abb. 2.2).

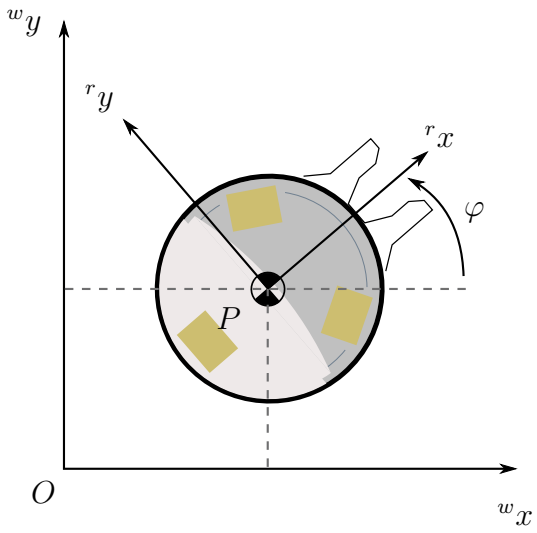


Abbildung 2.2: Globales Weltkoordinatensystem und lokales Roboterkoordinatensystem

Die Achse $^r x$ zeigt in Blickrichtung des Robotino®. Der Punkt P und somit die Position des Roboters kann direkt in Weltkoordinaten abgelesen werden. Die Orientierung φ ergibt sich durch den Winkelunterschied der beiden x-Achsen. Die Pose des Roboters kann eindeutig mit Hilfe des Vektors aus Gleichung 2.2 dargestellt werden.

$$^w\xi = \begin{bmatrix} ^w x \\ ^w y \\ \varphi \end{bmatrix} \tag{2.2}$$

Die Geschwindigkeit des Roboters ergibt sich aus der zeitlichen Ableitung der Pose. Um zwischen der Bewegung in Welt- und Roboterkoordinaten umzurechnen, wird die ortogonale Rotationsmatrix $\mathbf{R}$ verwendet [SN04].

$$^r\dot{\xi} = \begin{bmatrix} ^r\dot{x} \\ ^r\dot{y} \\ \dot{\varphi} \end{bmatrix} = \mathbf{R} \cdot {}^w\dot{\xi} \tag{2.3}$$

$$\mathbf{R} = \begin{bmatrix} \cos(\varphi) & \sin(\varphi) & 0 \\ -\sin(\varphi) & \cos(\varphi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

Es ist zu erkennen, dass $\mathbf{R}$ eine quadratische Matrix mit vollem Rang ist. Dies ermöglicht die Findung einer inversen Matrix. Die Koodinatentransformation wird mit deren Hilfe invertiert.

$$^w\dot{\xi} = \begin{bmatrix} ^w\dot{x} \\ ^w\dot{y} \\ \dot{\varphi} \end{bmatrix} = \mathbf{R}^{-1} \cdot {}^r\dot{\xi} \tag{2.5}$$

$$\mathbf{R}^{-1} = \begin{bmatrix} \cos(\varphi) & -\sin(\varphi) & 0 \\ \sin(\varphi) & \cos(\varphi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.6}$$

Zur Entwicklung der inversen Kinematik muss nun eine Beziehung zwischen der Geschwindigkeit in Weltkoordinaten und der Motorgeschwindigkeiten ω_j mit $j = 1 \ldots 3$ herstellt werden. Der Roboter wird dabei als fester Körper ohne Gelenke betrachtet.

Die Motorgeschwindigkeit wird mit Hilfe der translatorischen Geschwindigkeitsvektoren der Räder $\dot{q}_j$ beschrieben.

$$\omega_j = \frac{\dot{q}_j \cdot i}{r} \qquad \text{für} \quad j = 1 \ldots 3 \tag{2.7}$$

In Abbildung 2.3 kann erkannt werden, dass die Angriffspunkte von $\dot{q}_j$ nicht im Mittelpunkt des Roboters liegen. Dies hat zur Folge, dass sowohl eine translatorische als auch eine rotatorische Bewegung des Roboters zu einer Radbewegung führt.

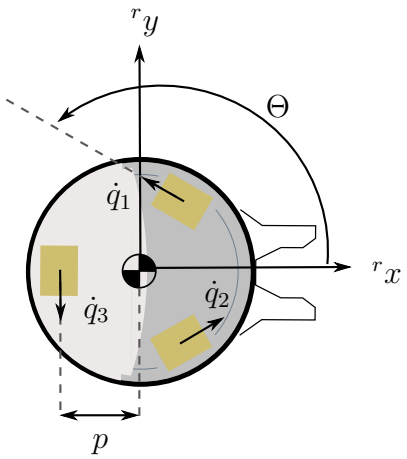


Abbildung 2.3: Lokales Roboterkoordinatensystem und Radkoordinaten

Es wird zwischen $\dot{q}_{j_{trans}}$ und $\dot{q}_{j_{rot}}$ unterschieden.

$$\dot{q}_j = \dot{q}_{j_{trans}} + \dot{q}_{j_{rot}} \tag{2.8}$$

Wird der Roboter um seinen Mittelpunkt gedreht, also eine rein rotatorische Bewegung, ergibt sich aufgrund der symmetrischen Geometrie dieselbe Geschwindigkeit für alle drei Räder. Das Verhältnis zwischen Dreh- und Radgeschwindigkeit wird durch den Abstand p vom Rad zum Mittelpunkt des Roboters bestimmt.

$$\dot{q}_{j_{rot}} = p \cdot \dot{\varphi} \tag{2.9}$$

Der Einfluss der translatorischen Geschwindigkeit des Roboters hängt von der Orientierung Θ der Räder ab. Dieser lässt sich allgemein mit Hilfe der trigonometrischen Funktionen bestimmen.

$$\dot{q}_{j_{trans}} = \cos(\Theta_j) \cdot {}^r\dot{x} + \sin(\Theta_j) \cdot {}^r\dot{y} \tag{2.10}$$

Aus Gleichung 2.8 folgt:

$$\dot{q}_j = \cos(\Theta_j) \cdot {}^r\dot{x} + \sin(\Theta_j) \cdot {}^r\dot{y} + p \cdot \dot{\varphi} \tag{2.11}$$

Die Orientierung Θ der einzelnen Räder ergibt sich aus der Geometrie des Antriebes. Folgende Werte können in Bezug auf die $^r x$-Achse ermittelt werden [Fes07]:

$$\Theta_1 = 150° \tag{2.12}$$

$$\Theta_2 = 270° \tag{2.13}$$

$$\Theta_3 = 30° \tag{2.14}$$

Aus der Orientierung und mit Hilfe der Gleichungen 2.10 und 2.7 kann die Motorgeschwindigkeit bestimmt werden.

$$\omega_1 = \frac{i}{r} \cdot \left(-\frac{\sqrt{3}}{2} \cdot {}^r\dot{x} + 0.5 \cdot {}^r\dot{y} + \cdot l \cdot \dot{\varphi} \right) \tag{2.15}$$

$$\omega_2 = \frac{i}{r} \cdot \left(- {}^r\dot{y} + l \cdot \dot{\varphi} \right) \tag{2.16}$$

$$\omega_3 = \frac{i}{r} \cdot \left(\frac{\sqrt{3}}{2} \cdot {}^r\dot{x} + 0.5 \cdot {}^r\dot{y} + l \cdot \varphi \right) \tag{2.17}$$

Die Ergebnisse werden in Matrixschreibweise zusammengefasst:

$$\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} = \frac{i}{r} \cdot \mathbf{T} \begin{bmatrix} {}^r\dot{x} \\ {}^r\dot{y} \\ \dot{\varphi} \end{bmatrix} \tag{2.18}$$

$$\mathbf{T} = \begin{bmatrix} -\frac{\sqrt{3}}{2} & 0.5 & l \\ 0 & -1 & l \\ \frac{\sqrt{3}}{2} & 0.5 & l \end{bmatrix} \tag{2.19}$$

Mit Hilfe von Gleichung 2.4 wird nun die inverse Kinematik entwickelt.

$$\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} = \frac{i}{r} \cdot \begin{bmatrix} -\frac{\sqrt{3}}{2} & 0.5 & l \\ 0 & -1 & l \\ \frac{\sqrt{3}}{2} & 0.5 & l \end{bmatrix} \cdot \begin{bmatrix} \cos(\varphi) & \sin(\varphi) & 0 \\ -\sin(\varphi) & \cos(\varphi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} {}^w\dot{x} \\ {}^w\dot{y} \\ \dot{\varphi} \end{bmatrix} \tag{2.20}$$

Da es sich auch bei 2.19 um eine quadratische Matrix mit vollem Rang handelt, kann mit Hilfe der Inversen und Gleichung 2.6 die direkte Kinematik bestimmt werden.

$$\begin{bmatrix} {}^w\dot{x} \\ {}^w\dot{y} \\ \dot{\varphi} \end{bmatrix} = \frac{r}{i} \cdot \mathbf{R}^{-1} \cdot \mathbf{T}^{-1} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} \tag{2.21}$$

$$\mathbf{T}^{-1} = \begin{bmatrix} -\frac{\sqrt{3}}{3} & 0 & \frac{\sqrt{3}}{3} \\ \frac{1}{3} & \frac{2}{3} & \frac{1}{3} \\ \frac{1}{3 \cdot l} & \frac{1}{3 \cdot l} & \frac{1}{3 \cdot l} \end{bmatrix} \tag{2.22}$$

2.3 Dynamische Modellierung

Die dynamische Modellierung wird auf Basis des Roboterkoordinatensystems durchgeführt. Zur Entwicklung werden folgende Schritte abgearbeitet:

1. Aufstellen der Kräftegleichungen
2. Erstellen eines Rechenmodells für die Mechanik des Getriebemotors
3. Erstellen eines Rechenmodells für die Elektrik des Getriebemotors
4. Kopplung der Mechanik und Elektrik

Der Roboter wird als starrer Körper mit der Masse m_r und dem Trägheitsmoment J_r betrachtet.

Die Beschreibung findet auf Basis der Newton-Eulerschen Gleichung statt. Der Impulssatz nach Newton beschreibt die translatorischen Bewegungen des Körpers [Ise06].

$$\begin{bmatrix} {}^rF_x \\ {}^rF_y \end{bmatrix} = m_r \cdot \left(\begin{bmatrix} {}^r\ddot{x} \\ {}^r\ddot{y} \end{bmatrix} + \dot{\varphi} \times \begin{bmatrix} {}^r\dot{x} \\ {}^r\dot{y} \end{bmatrix} \right) \quad (2.23)$$

Die rotatorische Bewegung wird aus dem Drallsatz von Euler entwickelt. M_P beschreibt das angreifende Moment am Mittelpunkt des Robotino®.

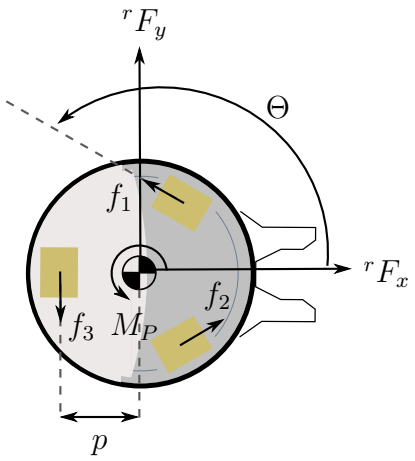


Abbildung 2.4: Angreifende Kräfte und Momente am Robotino®

$$M_P = \ddot{\varphi} \cdot J_r \quad (2.24)$$

Durch Zusammenfassen der Gleichungen 2.23 und 2.24 kann folgende dynamische Beziehung aufgestellt werden:

$$\begin{bmatrix} {}^r\ddot{x} \\ {}^r\ddot{y} \\ \ddot{\varphi} \end{bmatrix} = \begin{bmatrix} \dot{y}_r \cdot \dot{\varphi} \\ \dot{x}_r \cdot \dot{\varphi} \\ 0 \end{bmatrix} + \mathbf{H} \begin{bmatrix} {}^rF_x \\ {}^rF_y \\ M_P \end{bmatrix} \quad (2.25)$$

$\mathbf{H}$ ist die Trägheitsmatrix des Systems.

$$\mathbf{H} = \begin{bmatrix} \frac{1}{m_r} & 0 & 0 \\ 0 & \frac{1}{m_r} & 0 \\ 0 & 0 & \frac{1}{J_r} \end{bmatrix} \quad (2.26)$$

Zur Bestimmung der Kräfte ${}^rF_x$ und ${}^rF_y$ und dem Moment M_P wird das in Abbildung 2.4 dargestellte Kräftegleichgewicht aufgestellt.

$$\begin{bmatrix} {}^rF_x \\ {}^rF_y \\ M_P \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{3}}{2} & 0 & -\frac{\sqrt{3}}{2} \\ -0.5 & 1 & -0.5 \\ l & l & l \end{bmatrix} \cdot \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} \tag{2.27}$$

Verantwortlich für die Erzeugung der Kräfte f_1, f_2 und f_3 ist ein Getriebemotor. Zur Modellierung wird das Getriebe als ideal betrachtet. Unter Voraussetzung der Schlupffreiheit der Räder kann folgende Beziehung aufgestellt werden[Wei10]:

$$J_{AS} \cdot \dot{\omega} + M_{Reib} = M_{Antrieb} + M_{Abtrieb} \tag{2.28}$$

Das Trägheitsmoment des Antriebsstrangs J_{AS} ergibt sich aus den einzelnen Trägheitsmomenten des Antriebs J_{Motor} und Abtriebs J_{Rad}.

$$J_{AS} = J_{Motor} \cdot i^2 + J_{Rad} \tag{2.29}$$

Es wird angenommen, dass das Reibmoment linear von der Motorgeschwindigkeit abhängt. Das Verhältnis wird mit Hilfe der Reibkonstante c_r bestimmt.

$$M_{Reib} = c_r \cdot \omega \tag{2.30}$$

Das Antriebsmoment wird von einem Gleichstrommotor erzeugt. Dieser besitzt eine lineare Kopplung zwischen seiner Mechanik und Elektrik. Es wird die allgemeine Motorgleichung mit dem Widerstand R, der Induktivität L, dem Motorstrom I und der Motorspannung U betrachtet [Kas09]:

$$L \cdot \dot{I} + R \cdot I = U - U_{ind} \tag{2.31}$$

Die induzierte Gegenspannung U_{ind} ist linear von der Motordrehzahl abhängig. Es wird die Kopplungskonstante c_e eingeführt.

$$U_{ind} = c_e \cdot \omega \cdot i \tag{2.32}$$

Da die elektrische Zeitkonstante des Motors sehr viel kleiner als die mechanische Zeitkonstante ist, kann die Dynamik des elektrischen Kreises vernachlässigt werden[LWJZL03].

$$\frac{dI}{dt} = 0 \tag{2.33}$$

Dies führt zu folgender Motorgleichung:

$$R \cdot I = U - c_e \cdot \omega \cdot i \tag{2.34}$$

Um die Beziehung zwischen dem Motorstrom und dem Antriebsmoment zu beschreiben, wird eine weitere Kopplungskonstante eingeführt.

$$M_{Antrieb} = I \cdot c_m = \frac{c_m}{R} \cdot (U - c_e \cdot \omega \cdot i) \tag{2.35}$$

An dieser Stelle kann das dynamische Modell des Getriebemotors aufstellt werden.

$$M_{Abtrieb} = r \cdot \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} = J_{AS} \cdot \begin{bmatrix} \dot{\omega}_1 \\ \dot{\omega}_2 \\ \dot{\omega}_3 \end{bmatrix} + c_r \cdot \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} - \frac{c_m}{R} \cdot \left(\begin{bmatrix} U_1 \\ U_2 \\ U_3 \end{bmatrix} - c_e \cdot i \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} \right) \tag{2.36}$$

Mit Hilfe der inversen Kinematik kann die Gleichung 2.36 umgeschrieben werden.

$$\begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} = \frac{1}{r} \cdot \left(J_{AS} \cdot \frac{i}{r} \cdot \mathbf{T} \cdot \begin{bmatrix} {}^r\ddot{x} \\ {}^r\ddot{y} \\ \ddot{\varphi} \end{bmatrix} + \left(c_r + \frac{c_m \cdot c_e \cdot i}{R} \right) \cdot \frac{i}{r} \cdot \mathbf{T} \cdot \begin{bmatrix} {}^r\dot{x} \\ {}^r\dot{y} \\ \dot{\varphi} \end{bmatrix} - \frac{c_m}{R} \cdot \begin{bmatrix} U_1 \\ U_2 \\ U_3 \end{bmatrix} \right) \tag{2.37}$$

Aus den Gleichungen 2.25, 2.27 und 2.37 kann unter Verwendung von Maple die Dynamik des Systems aufgestellt werden. Der verwendete Maple-Code ist dem Anhang A zu entnehmen.

$$\begin{bmatrix} {}^r\ddot{x} \\ {}^r\ddot{y} \\ \ddot{\varphi} \end{bmatrix} = \begin{bmatrix} a_1 & 0 & 0 \\ 0 & a_1 & 0 \\ 0 & 0 & a_2 \end{bmatrix} \cdot \begin{bmatrix} {}^r\dot{x} \\ {}^r\dot{y} \\ \dot{\varphi} \end{bmatrix} + \begin{bmatrix} a_2 \cdot {}^r\dot{y} \cdot \dot{\varphi} \\ a_2 \cdot {}^r\dot{x} \cdot \dot{\varphi} \\ 0 \end{bmatrix} + \begin{bmatrix} -\sqrt{3} \cdot b_1 & 0 & \sqrt{3} \cdot b_1 \\ b_1 & -2 \cdot b_1 & b_1 \\ b_2 & b_2 & b_2 \end{bmatrix} \cdot \begin{bmatrix} U_1 \\ U_2 \\ U_3 \end{bmatrix} \tag{2.38}$$

mit

$$a_1 = -\frac{3 \cdot i \cdot (c_r \cdot R + c_m \cdot c_e \cdot i)}{R \cdot (3 \cdot J_{AS} \cdot i + 2 \cdot r^2 \cdot m)} \tag{2.39}$$

$$a_2 = \frac{2 \cdot r^2 \cdot m}{3 \cdot J_{AS} \cdot i + 2 \cdot r^2 \cdot m} \tag{2.42}$$

$$a_3 = -\frac{3 \cdot l^2 \cdot i \cdot (c_r \cdot R + c_m \cdot c_e \cdot i)}{R \cdot (3 \cdot l^2 \cdot J_{AS} \cdot i - r^2 \cdot J_r)} \tag{2.40}$$

$$b_1 = \frac{c_m \cdot r}{R \cdot (3 \cdot J_{AS} \cdot i + 2 \cdot r^2 \cdot m)} \tag{2.43}$$

$$b_2 = \frac{l \cdot c_m \cdot r}{R \cdot (3 \cdot l^2 \cdot J_{AS} \cdot i - r^2 \cdot J_r)} \tag{2.41}$$

3 Bahnplanung

3.1 Einleitung

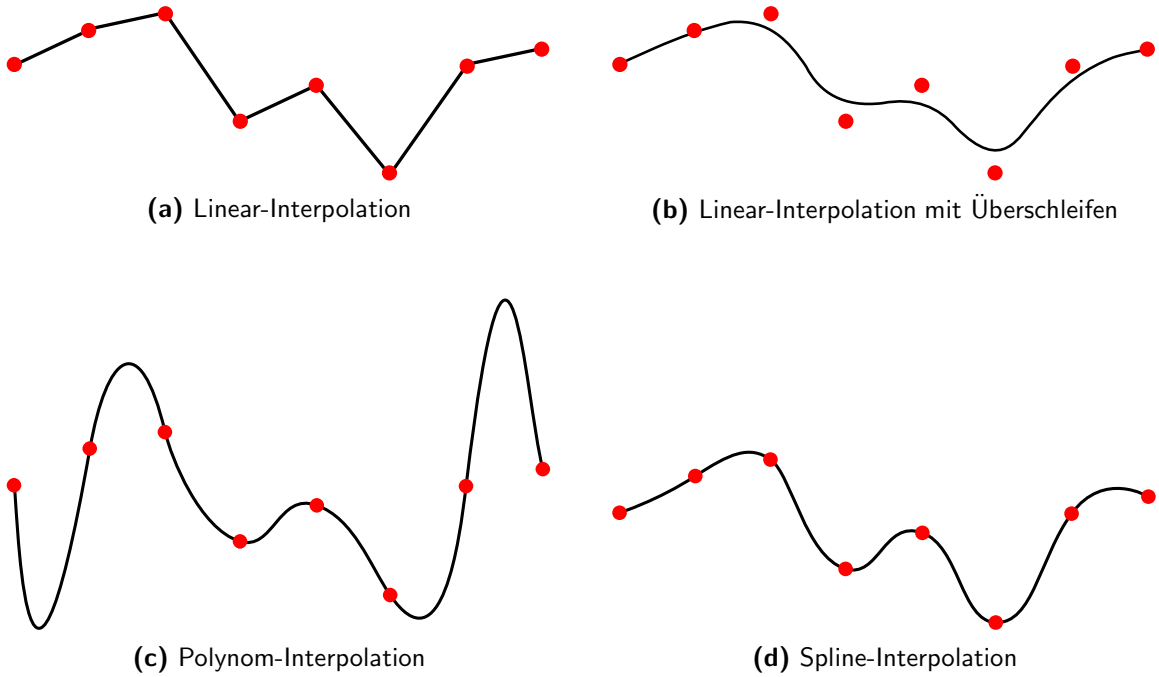
Ziel der in diesem Kapitel beschriebenen Bahnplanung ist es, einen kollisionsfreien Weg durch eine bekannte Produktionsstätte zu finden. Diese Aufgabe umfasst zwei Teilbereiche. Im ersten werden zulässige Knotenpunkte auf dem Spielfeld definiert. Diese können statisch sein oder dynamisch zur Laufzeit bestimmt werden. Zur Erfüllung der Zielstellung müssen, anhand eines Start- und Zielpunktes benachbarte Knotenpunkte möglichst sinnvoll miteinander verbunden und nacheinander abgefahren werden. Als Knotenpunkte dienen Tupel aus Ortspunkten und einer Zeit. Diese werden mit Hilfe eines modifizierten A*-Algorithmus verarbeitet. Dabei handelt es sich um einen Suchalgorithmus, der auf Basis von Schätzfunktionen in der Lage ist, zielgerichtet den kürzesten Weg zu finden. Die Besonderheit des A*-Algorithmus ist es, dass sich durch dessen Anwendung die optimale Lösung ergibt, falls diese existiert. Die genaue Erläuterung dieses Verfahrens ist nicht Teil dieser Arbeit und dessen Implementierung wird als gegeben hingenommen. Allgemeine Informationen zur Entwicklung eines A*-Algorithmus kann [Cho05] entnommen werden.

Um in der Lage zu sein, die zulässigen Ortspunkte abzufahren, müssen im zweiten Schritt die Wege dazwischen mathematisch beschrieben werden. Dies geschieht mit Hilfe von Interpolation. Dabei wird über die Art der Interpolation die Gestalt der Bahn bestimmt. Je nach Zielstellung bieten die verschiedenen Interpolationsarten Vor- und Nachteile. Im Abschnitt 3.2 werden daher die Anforderungen besprochen, die zur Findung einer geeigneten Interpolationsart geführt haben. Mit deren Hilfe werden Trajektorien für die translatorische und rotatorische Bewegung des Roboters entwickelt.

Im Abschnitt 3.3.2 werden die verwendeten Geschwindigkeitsprofile besprochen. Der omnidirektionale Antrieb kann aufgrund der Beschaffenheit der Räder keine hohen Beschleunigungen realisieren und neigt zum Durchdrehen [ES10]. Schlupf führt zu Odometriefehlern und somit zu Abweichungen zwischen Bahnplanung und Bahnführung. Um diese Fehler zu minimieren, ist es notwendig, das Anfahren und Abbremsen mit Hilfe von Geschwindigkeitsprofilen zu regeln.

3.2 Auswahl der Interpolationsart

Interpolation ist ein numerisches Verfahren, um aus gegebenen diskreten Werten eine stetige Funktion zu extrahieren. Diese Funktion beschreibt die Bahn zwischen zwei Wegpunkten. Folgende Interpolationsarten wurden im Laufe der Recherche betrachtet:



(a) Linear-Interpolation

(b) Linear-Interpolation mit Überschleifen

(c) Polynom-Interpolation

(d) Spline-Interpolation

Abbildung 3.1: Interpolationsarten

Um eine geeignete Interpolationsart auszuwählen, werden Anforderungen an die Gestalt der Bahn gestellt. Diese ergeben sich aus der zu lösenden Aufgabe, der gewählten Roboterplattform und dem Arbeitsraum.

Zuerst wird die an den Roboter gestellte Aufgabe betrachtet. Um erfolgreich an der LLSF teilzunehmen, müssen in gegebener Zeit möglichst viele Produkte hergestellt werden. Dazu ist eine schnelle Fahrt erforderlich, ohne dass beim Transport ein Produkt verloren wird. Transportiert wird mit Hilfe einer speziell angefertigten Schiebevorrichtung. Diese ist in der Lage, die Pucks innerhalb eines bestimmten Winkels zur Fahrtrichtung zu bewegen.

Die lineare Interpolation (Abb. 3.1a) ergibt den kürzesten Weg zum Ziel. Da der Robotino® sich zu jedem Zeitpunkt innerhalb des kritischen Winkels bewegen muss, ist das Anhalten und Drehen in einem Zwischenpunkt unumgänglich. Dieser Vorgang benötigt Zeit und ist somit nicht zweckmäßig.

Mit Hilfe von Überschleifen (Abb. 3.1b) kann dieses Defizit beseitigt werden. Die harten Ecken werden abgerundet und es ergibt sich eine vollständig differenzierbare Funktion. Die Bahn ver-

läuft dabei an den Knotenpunkten vorbei. Aufgrund der Wahl der Knotenpunkte ist es jedoch notwendig, sie zu durchfahren, um Kollisionen mit dem Spielfeld bzw. anderen Robotern zu vermeiden. Dies und die vollständige Differenzierbarkeit wird von den restlichen Interpolationsarten gewährleistet.

Die Polynominterpolation (Abb. 3.1c) tendiert bei zunehmender Anzahl an Stützstellen zum Überschwingen und es kann zu starken Oszillationen an den Intervallgrenzen kommen. Dieses Verhalten wird als *Runges Phänomen* bezeichnet [AHK+09]. Die Strecke wird dadurch unnötig verlängert und das Kollisionsrisiko erhöht. Darüber hinaus weist die Kurve starke Krümmungen auf. Diese stellen große Herausforderungen an die Bahnführung, da auftretende Corioliskräfte kompensiert werden müssen.

Die optimale Bahn ergibt sich durch Anwendung der Spline-Inerpolation (Abb. 3.1d). Natürliche kubische Splines besitzen eine Minimaleigenschaft gegenüber der Krümmung [Pla06] und erfüllen somit alle an die Bahn gestellten Forderungen. Ein weiterer Vorteil sind die einfache programmiertechnische Umsetzung und geringe Berechnungszeiten.

3.3 Trajektoriengenerierung

Als Trajektorie wird der zeitliche Verlauf einer Zustandsgröße des Systems bezeichnet. Es werden Trajektorien für die x- und y-Position des Roboters, sowie für seine Orientierung bestimmt. Dies geschieht in folgenden Schritten:

1. Entwicklung eines allgemeinen Algorithmus zur Splineinterpolation
2. Erstellen eines Ortsvektors zur Beschreibung der Bahn
3. Umparametrisierung des Ortsvektors
4. Anwendung eines Geschwindigkeitsprofils
5. Bestimmung der Orientierung des Roboters

3.3.1 Splineinterpolation

Abbildung 3.2 zeigt eine Spline-Interpolation für diskrete Wertepaare (x_j, y_j).

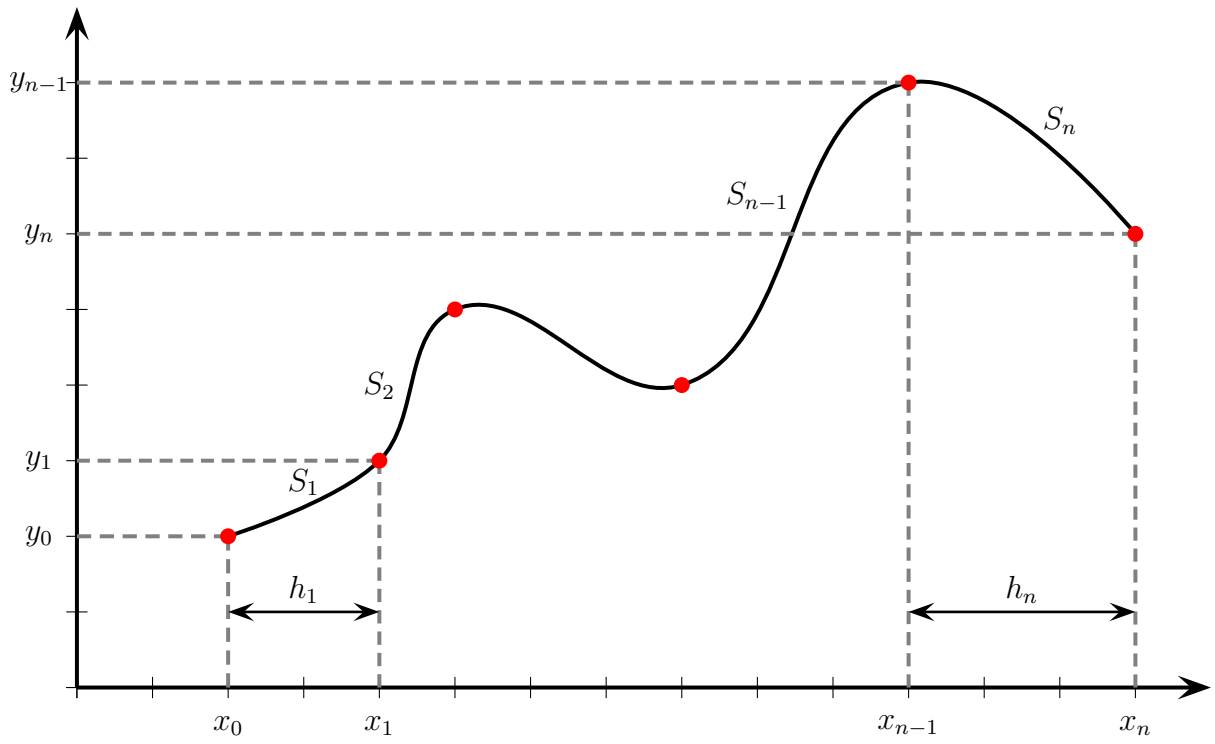


Abbildung 3.2: Natürlicher kubischer Spline

Bei einem kubischen Spline S ist S_k in jedem Teilintervall $[x_{j-1}, x_j]$ ein Polynom dritten Grades (Gl. 3.1).

$$S_j(x) = a_j + b_j \cdot (x - x_{j-1}) + c_k \cdot (x - x_{j-1})^2 + d_j \cdot (x - x_{j-1})^3 \tag{3.1}$$

Der kubische Spline ist mit Hilfe von vier Koeffizienten vollständig beschrieben. Dies führt zu $4 \cdot n$ Unbekannten bei $n+1$ Stützpunkten. Zur Bestimmung der Unbekannten benötigen werden $4 \cdot n$ Gleichungen benötigt. Es werden folgende Forderungen an den Spline gestellt [FH07]:

- Die linke Intervallgrenze x_{j-1} von S_j muss dem Funktionswert y_{j-1} entsprechen.

$$S_k(x_{j-1}) = y_{j-1} \qquad \text{für} \qquad j = 1 \ldots n \tag{3.2}$$

- Die rechte Intervallgrenze x_j von S_j muss dem Funktionswert y_j entsprechen.

$$S_i(x_j) = y_j \qquad \text{für} \qquad j = 1 \ldots n \tag{3.3}$$

- Der Anstieg sowie die Krümmung benachbarter Polynome muss an der Intervallgrenze x_j gleich sein.

$$S'_j(x_j) = S'_{j+1}(x_j) \qquad \text{für} \qquad j = 1 \ldots n \tag{3.4}$$

$$S''_j(x_j) = S''_{j+1}(x_j) \qquad \text{für} \qquad j = 1 \ldots n \tag{3.5}$$

Zur Erfüllung der Forderungen wird Gleichung 3.1 zweimal abgeleitet.

$$S'_j(x) = b_j + 2 \cdot c_j \cdot (x - x_{x_j - 1}) + 3 \cdot d_j \cdot (x - x_{x_j - 1})^2 \tag{3.6}$$

$$S''_j(x) = 2 \cdot c_j + 6 \cdot d_j \cdot (x - x_{x_j - 1}) \tag{3.7}$$

Mit Hilfe von 3.2 kann der erste Parameter direkt bestimmt werden.

$$S_k(x_{j-1}) = a_j = y_{j-1} \tag{3.8}$$

Unter Berücksichtigung der Gleichheit der Krümmung benachbarter Polynome (Gl. 3.5) ergibt sich:

$$2 \cdot c_j + 6 \cdot d_j \cdot (x_j - x_{j-1}) = 2 \cdot c_{j+1} \tag{3.9}$$

Daraus folgt

$$d_j = \frac{(c_{j+1} - c_j)}{3 \cdot h_j} \tag{3.10}$$

mit

$$h_j = x_j - x_{j-1} \tag{3.11}$$

Unter Verwendung von Gleichung 3.3 und den bereits ermittelten Beziehungen 3.8, 3.10 und 3.11 ergibt sich:

$$a_j + b_j \cdot (x_j - x_{j-1}) + c_j \cdot (x_j - x_{j-1})^2 + d_j \cdot (x_j - x_{j-1})^3 = a_{j+1} \tag{3.12}$$

$$y_{j-1} + b_j \cdot h_j + c_j \cdot h_j^2 + d_j \cdot h_j^3 = y_j \tag{3.13}$$

$$y_{j-1} + b_j \cdot h_j + c_j \cdot h_j^2 + \frac{(c_{j+1} - c_j) \cdot h_j^2}{3} = y_j \tag{3.14}$$

Es wird nach b_j aufgelöst.

$$b_j = \frac{y_j - y_{j-1}}{h_j} - \frac{h_j \cdot (2 \cdot c_j + c_{j+1})}{3} \tag{3.15}$$

Die Splineparameter a_j, b_j und d_j können direkt ermittelt werden oder sind nur noch von c_j abhängig. Um c_j zu bestimmen, wird die Gleichheit des Anstieges benachbarter Polynome betrachtet (Gl. 3.4). Unter Verwendung der Gleichungen 3.8, 3.10, 3.11 und 3.15 kann folgende Beziehung aufgestellt werden:

$$b_{j-1} + 2 \cdot c_{j-1} \cdot (x_{j-1} - x_{j-2}) + 3 \cdot d_{j-1} \cdot (x_{j-1} - x_{j-2})^2 = b_j \tag{3.16}$$

$$\begin{aligned} \frac{y_{j-1} - y_{j-2}}{h_{j-1}} - \frac{h_{j-1} \cdot (2 \cdot c_{j-1} + c_j)}{3} + 2 \cdot c_{j-1} \cdot h_{j-1} + 3 \cdot d_{j-1} \cdot h_{j-1}^2 \\ = \frac{y_j - y_{j-1}}{h_j} - \frac{h_j \cdot (2 \cdot c_j + c_{j+1})}{3} \end{aligned} \tag{3.17}$$

Daraus folgt

$$h_{j-1} \cdot c_{j-1} + 2 \cdot (h_{j-1} + h_j) \cdot c_j + h_j \cdot c_{j+1} = g_j \tag{3.18}$$

mit

$$g_j = 3 \cdot \left(\frac{y_j - y_{j-1}}{h_j} - \frac{y_{j-1} - y_{j-2}}{h_{j-1}} \right) \tag{3.19}$$

Zur Lösung der Gleichung 3.18 wird sowohl x_{j-2} als auch y_{j-2} benötigt. Da diese im Bereich von $0 \ldots n$ definiert sind, ist die Gleichung nur für $j = 2 \ldots n$ lösbar. Sie enthält jedoch $n+1$ Unbekannte $(c_1 \ldots c_{n+1})$. Um dennoch eine Lösung zu finden, müssen Randbedingungen gefordert werden.

Der Vorsatz „natürliche“ vor kubische Splines beschreibt die Momentenfreiheit an deren Intervallgrenzen x_0 und x_n.

$$S_1''(x_0) = S_n''(x_n) = 0 \tag{3.20}$$

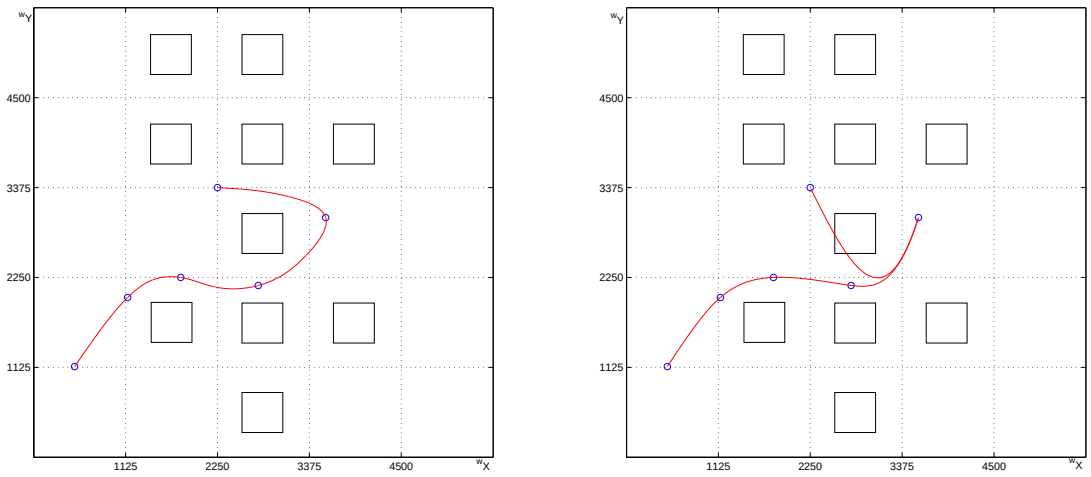
Unter Verwendung von 3.7 und 3.20 ergibt sich:

$$c_1 = 0 \qquad \text{und} \qquad c_{n+1} = 0 \tag{3.21}$$

Zur Bestimmung von c_2 bis c_n wird folgendes Gleichungssystem aufgestellt:

$$\begin{bmatrix} 2\cdot(h_1+h_2) & h_2 & 0 & \dots & & \\ h_2 & 2\cdot(h_2+h_3) & h_3 & 0 & \dots & \\ 0 & h_3 & 2\cdot(h_3+h_4) & \dots & & \\ \dots & & & & & \\ & & & \dots & 0 & h_{n-1} & 2\cdot(h_{n-1}+h_n) \end{bmatrix} \cdot \begin{bmatrix} c_2 \\ c_3 \\ . \\ . \\ . \\ c_n \end{bmatrix} = \begin{bmatrix} g_2 \\ g_3 \\ . \\ . \\ . \\ g_n \end{bmatrix} \tag{3.22}$$

Die Lösung des Gleichungssystems führt zu n bereichsweise definierten Polynomen zur Beschreibung der Bahn. Die Gleichungen 3.10 und 3.15 zeigen die Einschränkungen des Algorithmus. Wird h_j null, führt dies zu einer nicht definierten Division. Das bedeutet, dass aufeinander folgende Knotenpunkte nicht denselben x-Wert besitzen dürfen. Des Weiteren darf h_j keine wechselnden Vorzeichen haben. Dies führt zu ungewollten Mehrdeutigkeiten. Um die Auswirkungen auf den Anwendungsfall zu betrachten, wird das Weltkoordinatensystem in die linke untere Ecke des Spielfeldes gelegt.



(a) gewünschte Bahn **(b)** resultierende Bahn

Abbildung 3.3: Splineinterpolation

Die in Abbildung 3.3a dargestellte Bahn ist mathematisch nicht definiert. Sie besitzt an genau einer Stelle einen unendlich großen Anstieg. Dies würde Gleichung 3.6 widersprechen und führt zu der in Abbildung 3.3b dargestellten Bahn. Um die Einschränkungen des Algorithmus zu umgehen, werden die x- und y-Richtung getrennt voneinander betrachtet. Dazu wird der äquidistante Parameter k eingeführt.

$$k = 0 \dots n \tag{3.23}$$

Die weitere Betrachtung findet auf Basis des Weltkoordinatensystems $({}^{w}x, {}^{w}y)$ statt. Es werden zwei unabhängige Splines für die Wertepaare $(k_j, {}^{w}x_j)$ und $(k_j, {}^{w}y_j)$ entwickelt. Dadurch wird $h_j = 1 = h$ garantiert, was wiederum alle Sonderfälle ausschließt.

$${}^{w}x_j(k) = a_{xj} + b_{j,x} \cdot (k - (j-1)) + c_{j,x} \cdot (k - (j-1))^2 + d_{j,x} \cdot (k - (j-1))^3 \tag{3.24}$$

$${}^{w}y_j(k) = a_{j,y} + b_{j,y} \cdot (k - (j-1)) + c_{j,y} \cdot (k - (j-1))^2 + d_{j,y} \cdot (k - (j-1))^3 \tag{3.25}$$

Werden die Splines getrennt voneinander betrachtet, beschreiben sie den Verlauf der ${}^{w}x$- und ${}^{w}y$-Komponente der Bahn (Abb. 3.4). Um in der Lage zu sein, die in Abbildung 3.3a dargestellte Bahn zu beschreiben, werden die Gleichungen 3.24 und 3.25 zu einem veränderlichen Ortsvektor $\vec{r}(k)$ zusammengefasst.

$$\vec{r}(k) = \begin{bmatrix} {}^{w}x_j(k) \\ {}^{w}y_j(k) \end{bmatrix} \tag{3.26}$$

Der Bezugspunkt von $\vec{r}(k)$ liegt im Ursprung des Weltkoordinatensystems.

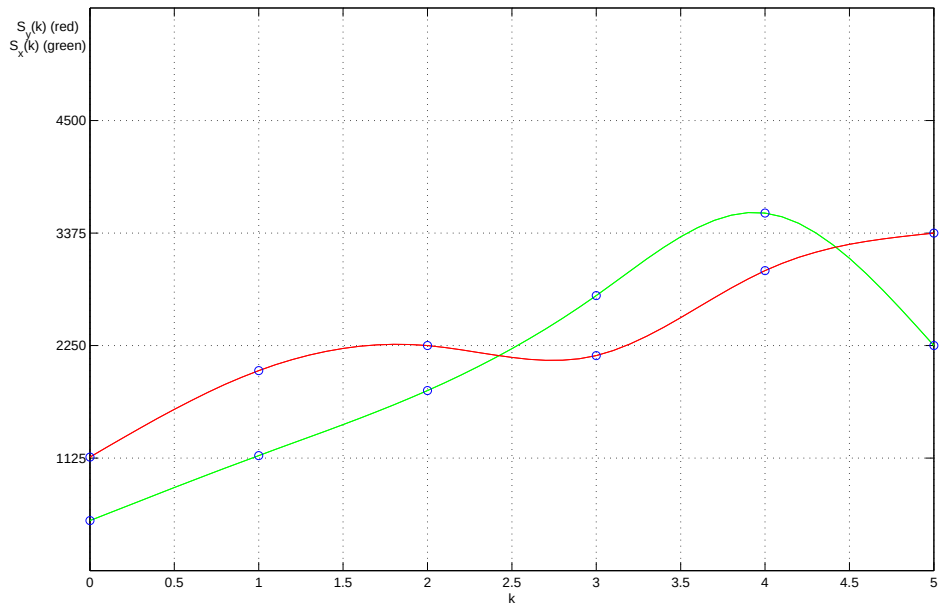


Abbildung 3.4: Bahnkurve in Parameterdarstellung

Ein weiterer Vorteil des äquidistanten Parameters ist die folgende Vereinfachung der Gleichungen 3.10, 3.15, 3.19 und 3.22:

$$d_j = \frac{1}{3} \cdot (c_{j+1} - c_j) \tag{3.27}$$

$$b_j = (y_j - y_{j-1}) - \frac{1}{3} \cdot (2 \cdot c_j + c_{j+1}) \tag{3.28}$$

$$g_j = 3 \cdot (y_j - 2 \cdot y_{j-1} + y_{j-2}) \tag{3.29}$$

$$\begin{bmatrix} 4 & 1 & 0 & \dots & & \\ 1 & 4 & 1 & 0 & \dots & \\ 0 & 1 & 4 & \dots & & \\ \dots & & & & & \\ & & & \dots & 0 & 1 & 4 \end{bmatrix} \cdot \begin{bmatrix} c_2 \\ c_3 \\ . \\ . \\ . \\ c_n \end{bmatrix} = \begin{bmatrix} g_2 \\ g_3 \\ . \\ . \\ . \\ g_n \end{bmatrix} \tag{3.30}$$

3.3.2 Anwendung von Geschwindigkeitsprofilen

Geschwindigkeitsprofile dienen dem geregelten Anfahren und Abbremsen und somit der Schlupfminimierung. Zur Anwendung der Geschwindigkeitsprofile müssen die Gleichungen 3.24 und 3.25 in einen physikalischen Kontext gebracht werden. Dazu wird eine Beziehung zwischen dem frei gewählten Parameter k und der Zeit t aufgestellt. Dies geschieht über die zurückgelegte Wegstrecke s. Durch Vorgabe einer absoluten Geschwindigkeit v des Roboters kann diese bestimmt werden.

$$s(t) = v \cdot t \tag{3.31}$$

Die berechnete Wegstrecke muss der Länge der durch den Ortsvektor $\vec{r}(k)$ beschriebenen Bahn entsprechen. Um diese Länge zu ermitteln, wird der Betrag der Ableitung des Vektors aufintegriert (Gl. 3.32).

$$l(k) = \int \sqrt{{}^{w}x'(k)^2 + {}^{w}y'(k)^2} dk \tag{3.32}$$

Es ergibt sich folgende Beziehung zwischen k und t:

$$s(t) = l(k) \tag{3.33}$$

Durch das Auflösen der Gleichung 3.32 nach k könnte die aktuelle Position zu jedem Zeitpunkt t bestimmt werden. Auf analytischem Wege gibt es für das Problem jedoch keine eindeutige Lösung. Es ist an dieser Stelle möglich, mit Hilfe von numerischen Verfahren ein passendes k zu einer vorgegebenen Wegstrecke zu finden. Dies ist jedoch je nach angestrebter Genauigkeit und Auflösung von s sehr rechenaufwendig und damit nicht tragbar für diese Anwendung. Um dennoch zu einem Ergebnis zu kommen, wird ein von der Wegstrecke abhängiger Ortsvektor $\vec{r}(s)$ erzeugt, dessen beschriebene Bahn der von $\vec{r}(k)$ hinreichend ähnlich ist. Dazu werden diskrete Stützpunkte mit Hilfe der Gleichung 3.32 ermittelt.

$$s_j = \int_{k_{j-1}}^{k_j} \sqrt{{}^w x_j'(k)^2 + {}^w y_j'(k)^2}\, dk \qquad \text{für} \qquad j = 1 \dots n \tag{3.34}$$

Es ergeben sich die Wertepaare $(s_j, {}^w x_j)$ und $(s_j, {}^w y_j)$. Mit Hilfe des in Abschnitt 3.3 vorgestellten Algorithmus werden die von der Wegstrecke abhängigen Splines $x_j(s)$ und $y_j(s)$ entwickelt und zum folgenden Ortsvektor zusammengefasst:

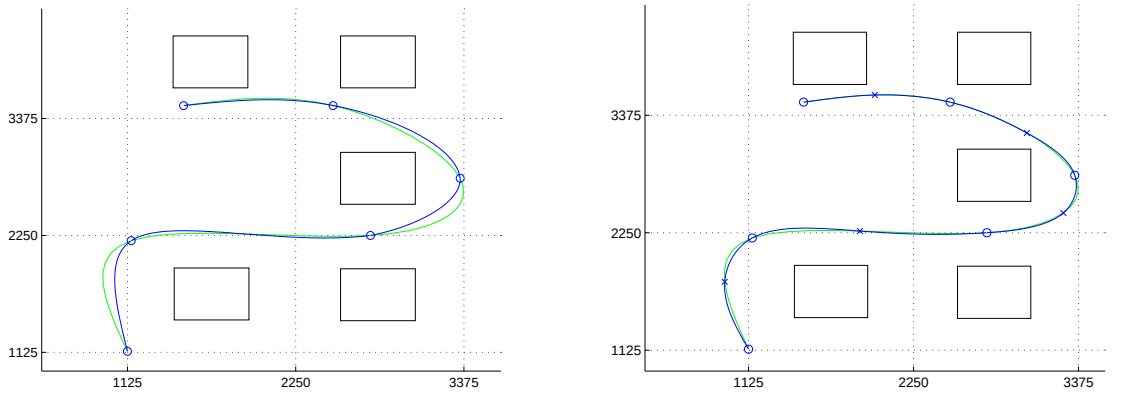
$$\vec{r}(s) = \begin{bmatrix} {}^w x_j(s) \\ {}^w y_j(s) \end{bmatrix} \tag{3.35}$$

mit

$${}^w x_j(s) = a_{xj} + b_{xj} \cdot (s - s_{j-1}) + c_{xj} \cdot (s - s_{j-1})^2 + d_{xj} \cdot (s - s_{j-1})^3 \tag{3.36}$$

$${}^w y_j(s) = a_{yj} + b_{yj} \cdot (s - s_{j-1}) + c_{yj} \cdot (s - s_{j-1})^2 + d_{yj} \cdot (s - s_{j-1})^3 \tag{3.37}$$

Im Zuge der Umparametrisierung wurde der äquidistante Parameter k durch s ersetzt. Die bisherige Annahme $h_j = 1 = h$ trifft nicht mehr zu. Durch die entstandene Ungleichheit von h_j kommt es zu unterschiedlichen Gewichtungen der Polynome des Splines. Dies führt zu Abweichungen der durch $\vec{r}(k)$ und $\vec{r}(s)$ beschriebenen Bahnen (Abb. 3.5a). Dabei ist die Größe der Abweichung abhängig von der Anzahl der Stützpunkte. Reichen diese für ein zufriedenstellendes Ergebnis nicht aus, müssen Zwischenpunkte mit passenden Weginformationen generiert werden.

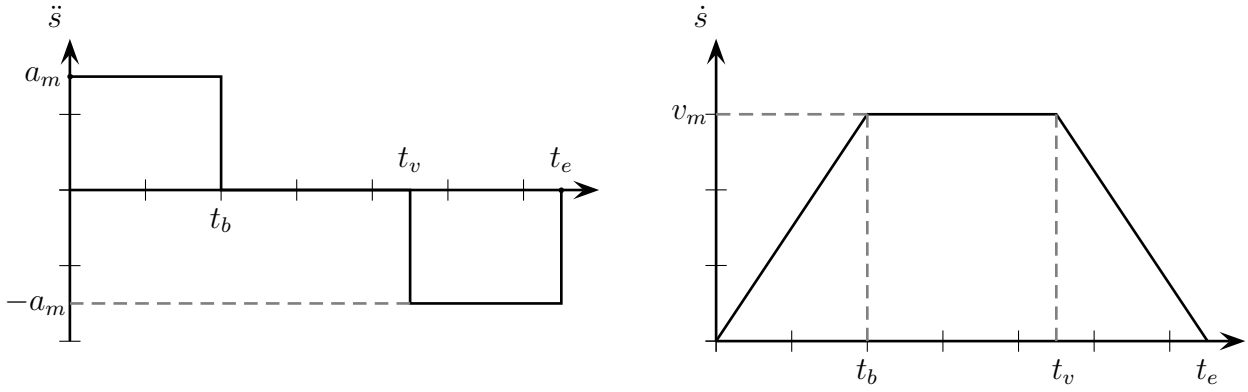


(a) Umparametrisierung ohne Zwischenpunkte **(b)** Umparametrisierung mit Zwischenpunkten

Abbildung 3.5: Vergleich der Umparametrisierung mit und ohne Zwischenpunkte

Wie in Abbildung 3.5b zu erkennen ist, kann durch Zuhilfenahme jeweils eines Zwischenpunktes die Abweichung deutlich minimiert werden.

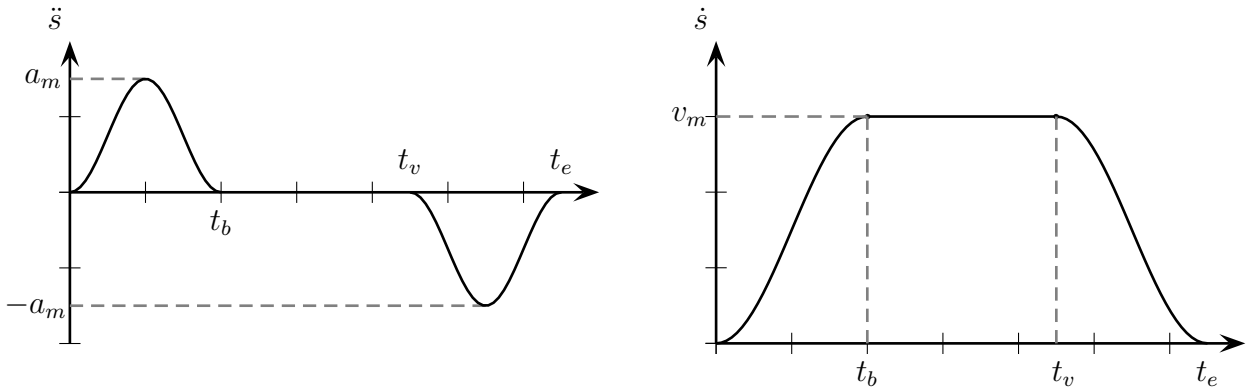
Die Position des Roboters kann nun anhand der zurückgelegten Strecke bestimmt werden. Diese ergibt sich durch Vorgabe einer Geschwindigkeit (Gl. 3.31). Der Verlauf der Geschwindigkeit wird mit Hilfe eines Profils bestimmt. Die einfachste Form eines Geschwindigkeitsprofils ist das Rampenprofil (Abb. 3.6). Dabei wird mit konstanter Beschleunigung a_m innerhalb der Beschleunigungszeit t_b die vorgegebene Geschwindigkeit v_m angefahren. Die Verzögerung setzt zum Zeitpunkt t_v ein und endet bei t_e.



(a) zeitlicher Verlauf der Beschleunigung **(b)** zeitlicher Verlauf der Geschwindigkeit

Abbildung 3.6: Rampenprofil

In Abbildung 3.6a ist zu erkennen, dass sich die Beschleunigung sprunghaft ändert. Daraus folgt, dass die zeitliche Ableitung der Beschleunigung - der Ruck - nicht beschränkt ist. Dies kann zur Anregung von Eigenschwingungen führen und somit zu Vibrationen, die zur Verminderung der Lebensdauer mechanischer Bauteile und des Getriebes führen können [Web07]. Um den Ruck zu verringern, wird anstelle des Rampenprofils ein Sinoidenprofil verwendet. Dadurch kann ein weicher Bewegungsablauf garantiert werden.



(a) zeitlicher Verlauf der Beschleunigung **(b)** zeitlicher Verlauf der Geschwindigkeit

Abbildung 3.7: Sinoidenprofil

Der Verlauf des Sinoidenprofils (Abb. 3.7) lässt sich in drei Phasen unterteilen. Begrenzt sind diese durch die Zeiten t_b, t_v und t_e. Die Zeiten bestimmen sich aus der vorgegebenen Geschwindigkeit v_m und Beschleunigung a_m, sowie aus der Länge der Strecke s_e [Web07].

$$t_b = \frac{2 \cdot v_m}{a_m} \qquad (3.38) \qquad t_e = \frac{s_e}{v_m} + t_b \qquad (3.39) \qquad t_v = t_e - t_b \qquad (3.40)$$

Die Beschleunigungsphase ist durch folgende Funktion festgelegt:

$$\ddot{s}(t) = a_m \cdot sin^2\left(\frac{\pi}{t_b} \cdot t\right) \qquad \text{für} \qquad 0 \leq t \leq t_b \qquad (3.41)$$

Durch Integration erhält man die passende Geschwindigkeit und die zurückgelegte Wegstrecke.

$$\dot{s}(t) = a_m \cdot \left(\frac{1}{2} \cdot t - \frac{t_b}{4 \cdot \pi} \cdot sin\left(\frac{2 \cdot \pi}{t_b} \cdot t\right)\right) \qquad \text{für} \qquad 0 \leq t \leq t_b \qquad (3.42)$$

$$s(t) = a_m \cdot \left(\frac{1}{4} \cdot t^2 + \frac{t_b^2}{8 \cdot \pi^2} \cdot \left(cos\left(\frac{2 \cdot \pi}{t_b} \cdot t\right) - 1\right)\right) \qquad \text{für} \qquad 0 \leq t \leq t_b \qquad (3.43)$$

Nach dem Erreichen von t_b beginnt die Phase der gleichförmigen Geschwindigkeit.

$$\dot{s}(t) = v_m \qquad \text{für} \qquad t_b \leq t \leq t_v \qquad (3.44)$$

$$s(t) = v_m \cdot (t - t_b) + s(t_b) \qquad \text{für} \qquad t_b \leq t \leq t_v \qquad (3.45)$$

Die Verzögerung setzt zum Zeitpunkt t_v ein. Sie hat dieselbe Form wie die Beschleunigung, mit negativem Vorzeichen und um t_v verschoben.

$$\ddot{s}(t) = a(t) = -a_m \cdot sin^2\left(\frac{\pi}{t_b} \cdot (t - t_v)\right) \qquad (3.46)$$

Um die Wegstrecke der Verzögerung zu bestimmen, wird Gleichung 3.46 integriert.

$$\begin{aligned}\dot{s}(t) &= v_m - \int_{t-t_v}^{t} a(\tau - t_v) \cdot d\tau \qquad \text{für} \qquad t_v \leq t \leq t_e \\ &= v_m - a_m \cdot \left(\frac{1}{2} \cdot (t - t_v) - \frac{t_b}{4 \cdot \pi} \cdot sin \left(\frac{2 \cdot \pi}{t_b} \cdot (t - t_v) \right) \right)\end{aligned} \tag{3.47}$$

$$\begin{aligned}s(t) &= s(t_v) + \int_{t-t_v}^{t} \dot{s}(\tau - t_v) \cdot d\tau \qquad \text{für} \qquad t_v \leq t \leq t_e \\ &= \frac{a_m}{2} \cdot \left[t_e \cdot (t + t_b) - \frac{(t^2 + t_e^2 + 2 \cdot t_b^2)}{2} + \frac{t_b^2}{4 \cdot \pi^2} \cdot \left(1 - cos \left(\frac{2 \cdot \pi}{t_b} \cdot (t - t_v) \right) \right) \right]\end{aligned} \tag{3.48}$$

Mit Hilfe der Wegstrecken kann der Verlauf der Bahn zu jedem Zeitpunkt t bestimmt werden.

$$\vec{r}(t) = \frac{x_j(t)}{y_j(t)} \tag{3.49}$$

mit

$${}^{w}x_j(t) = a_{xj} + b_{xj} \cdot (s(t) - s_{j-1}) + c_{xj} \cdot (s(t) - s_{j-1})^2 + d_{xj} \cdot (s(t) - s_{j-1})^3 \tag{3.50}$$

$${}^{w}y_j(t) = a_{yj} + b_{yj} \cdot (s(t) - s_{j-1}) + c_{yj} \cdot (s(t) - s_{j-1})^2 + d_{yj} \cdot (s(t) - s_{j-1})^3 \tag{3.51}$$

Bei der Vorgabe der Geschwindigkeit v_m muss gewährleistet sein, dass diese auch erreicht werden kann. Liegt v_m über der von der maximalen Beschleunigung b_m und der Gesamtstrecke s_e abhängigen maximalen Geschwindigkeit $v_{m,max}$, muss die vorgegebene Geschwindigkeit reduziert werden. Der zeitoptimale Weg ergibt sich bei Vorgabe der maximal möglichen Geschwindigkeit.

$$v_{m,max} = \sqrt{\frac{s_e \cdot a_m}{2}} \tag{3.52}$$

Zu beachten ist, dass $v_{m,max}$ von der Aktorik des Roboters begrenzt wird.

3.3.3 Bestimmung der Orientierung

Die in den vorangegangenen Abschnitten vorgestellten Algorithmen dienen als Grundlage für eine Fahrt entlang einer vorgegebenen Bahn. Um der Forderung nachzukommen, den Puck während der Fahrt nicht zu verlieren, muss der Roboter zu jedem Zeitpunkt in Fahrtrichtung ausgerichtet sein. Daraus folgt, dass die Orientierung dem Winkel des resultierenden Geschwindigkeitsvektors entsprechen muss.

Der Geschwindigkeitsvektor ergibt sich durch Ableitung des Ortsvektors $\vec{r}(t)$.

Mit Hilfe des Arkustangens kann die Orientierung des Roboters zu jedem Zeitpunkt t bestimmt werden.

$$\varphi_j(t) = arctan\left(\frac{\dot{y}_j(t)}{\dot{x}_j(t)}\right) \tag{3.53}$$

Zu beachten ist die Periodizität des Arkustangens. Er ist nur in positiver x-Richtung definiert. Befindet sich der Geschwindigkeitsvektor im Quadranten II oder III, wird dieser am Ursprung gespiegelt (Abb 3.8). Das bedeutet, er wird um π gedreht.

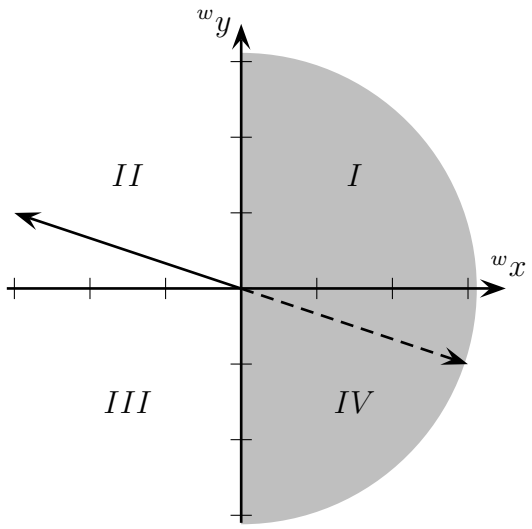


Abbildung 3.8: Wertebereich des Arkustangens

Mit Hilfe der Vorzeichen der Komponenten $^w\dot{x}_j(t)$ und $^w\dot{y}_j(t)$ kann die tatsächliche Lage des Vektors bestimmt und somit der resultierende Winkel korrigiert werden.

4 Bahnführung

4.1 Einleitung

Im folgenden Kapitel wird eine Regelstruktur entwickelt, mit der der Robotino® entlang einer vorberechneten Bahn geführt werden kann. Diese Aufgabe beinhaltet die Lösung der drei grundlegenden Probleme mobiler Roboter. Anhand des „Navigation, Guidance and Control"-Ansatzes lassen sich diese erläutern [NXST08].

1. Navigation → Wo bin ich?
2. Guidance → Wo will ich hin?
3. Control → Wie komme ich dahin?

Eine sinnvolle Ansteuerung des Roboters ist nur möglich, wenn diese Fragen beantwortet werden können.

Die Navigation findet in zwei Schritten statt. Im ersten wird die absolute Pose des Robotino® auf dem Spielfeld mit Hilfe eines Laserscanners ermittelt. Da dieser aufgrund seiner Drehgeschwindigkeit an physikalische Grenzen stößt, ist diese Lokalisationsmethode nicht echtzeitfähig. Um dennoch in der Lage zu sein, sich während der Bahnführung zu lokalisieren, wird aufbauend auf der globalen Pose die relative Bewegung des Robotino® mit Hilfe von Odometrie aufgenommen. Eine detaillierte Beschreibung der verwendeten Algorithmen zur Lokalisierung können den Quellen [Som12] und [Ler11] entnommen werden.

Um die Frage „Wo will ich hin?" zu klären, wird die nun bekannte aktuelle Pose mit der gewünschten Pose aus der in Kapitel 3 besprochenen Bahnplanung verglichen. Der daraus ermittelte Geschwindigkeitsvektor dient der Regelung (Control) als Vorgabe für die Roboteransteuerung.

Es ergibt sich ein Kreislauf für die Bahnführung (Abb. 4.1).

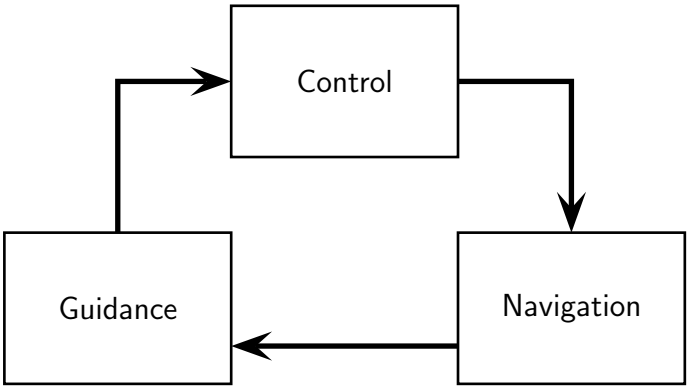


Abbildung 4.1: Kreislauf des „Navigation, Guidance and Control"-Ansatzes

In den folgenden Abschnitten wird dieser Kreislauf in einen Regelkreis umgesetzt und die enthaltenen Regler entsprechend ihren Anforderungen ausgelegt.

4.2 Regelkreis

Um die Idee von „Navigation, Guidance and Control" umzusetzen, werden Regelaufgaben definiert. Diese lassen sich in eine Kaskadenstruktur (Abb. 4.2) integrieren. Dabei entstehen zwei einander überlagerte Regelkreise. Den äußeren Regelkreis bildet ein Positionsregler. Er übergibt seinen Ausgang an den darunterliegenden Geschwindigkeitsregler und übernimmt damit die Aufgabe der Guidance. Der Vorteil dieser Struktur ist die Zerlegung des Reglers in zwei Teile. Dadurch entsteht eine zusätzliche Messgröße, mit deren Hilfe Störungen schneller wahrgenommen werden können. Die Regelung dieser Größe ermöglicht eine Verbesserung der Regelgüte gegenüber dem einschleifigen Regelkreis [Lun10a]. Bei diesem Entwurf muss sichergestellt werden das der innere Regelkreis auch ohne das Wirken des äußeren Kreises stabil ist. Zur Modellierung wird das in Kapitel 2 erstellte kinematische und dynamische Modell des Robotino® verwendet.

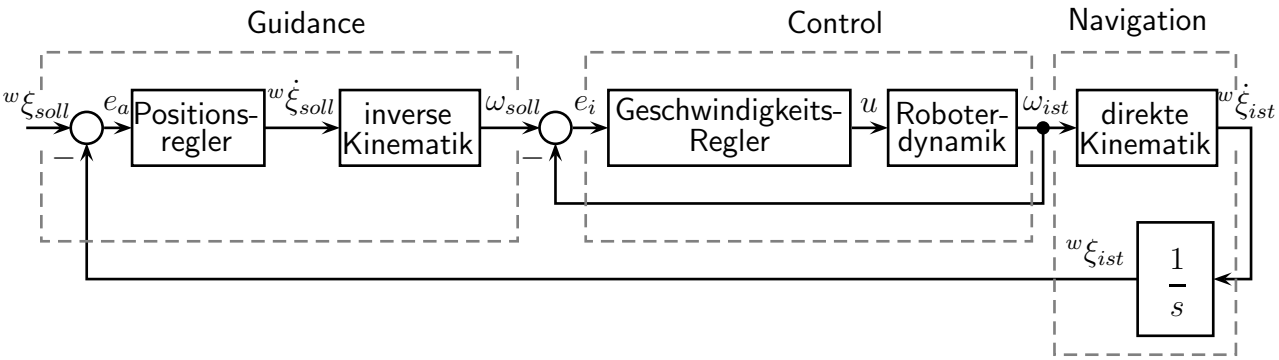


Abbildung 4.2: Regelkreis auf Basis des „Navigation, Guidance and Control"-Ansatzes

4.2.1 Geschwindigkeitsregler

Die Auslegung des Geschwindigkeitsreglers wird auf Basis des in Abschnitt 2.3 entwickelten dynamischen Modells (Gl. 4.1) des Robotino® durchgeführt.

$$\begin{bmatrix} {}^r\ddot{x} \\ {}^r\ddot{y} \\ \ddot{\varphi} \end{bmatrix} = \begin{bmatrix} a_1 & 0 & 0 \\ 0 & a_1 & 0 \\ 0 & 0 & a_2 \end{bmatrix} \cdot \begin{bmatrix} {}^r\dot{x} \\ {}^r\dot{y} \\ \dot{\varphi} \end{bmatrix} + \begin{bmatrix} a_2 \cdot {}^r\dot{y} \cdot \dot{\varphi} \\ a_2 \cdot {}^r\dot{x} \cdot \dot{\varphi} \\ 0 \end{bmatrix} + \begin{bmatrix} -\sqrt{3} \cdot b_1 & 0 & \sqrt{3} \cdot b_1 \\ b_1 & -2 \cdot b_1 & b_1 \\ b_2 & b_2 & b_2 \end{bmatrix} \cdot \begin{bmatrix} U_1 \\ U_2 \\ U_3 \end{bmatrix} \quad (4.1)$$

Es ergibt sich ein MIMO-System, also ein System mit mehreren Ein- und Ausgängen, die sich untereinander beeinflussen. Je nach Stärke der Verkopplung ist ein Mehrgrößenregler erforderlich [Lun10b]. Besteht nur eine schwache Kopplung, kann der Mehrgrößenregler durch separate Eingrößenregler ersetzt werden.

Das dynamische System besitzt eine Verkopplung der Eingangsgrößen mit den Ausgangsgrößen, sowie eine Verkopplung der Ausgangsgrößen untereinander. Die Kopplung untereinander lässt sich auf die Corioliskräfte zurückführen und entsteht nur bei einer Überlagerung von translatorischer und rotatorischer Bewegung. Aus der Forderung, den Roboter in Fahrtrichtung zu orientieren, und der Minimalbedingung der Bahnkrümmung (Kapitel 3) können folgende Behauptungen aufstellt werden:

$$ {}^r\dot{y} \cdot \dot{\varphi} \ll {}^r\dot{x} \quad \text{und} \quad {}^r\dot{x} \cdot \dot{\varphi} \ll {}^r\dot{y} \quad (4.2)$$

Dies erlaubt die Vereinfachung des dynamischen Systems.

$$\begin{bmatrix} {}^r\ddot{x} \\ {}^r\ddot{y} \\ \ddot{\varphi} \end{bmatrix} = \begin{bmatrix} a_1 & 0 & 0 \\ 0 & a_1 & 0 \\ 0 & 0 & a_2 \end{bmatrix} \cdot \begin{bmatrix} {}^r\dot{x} \\ {}^r\dot{y} \\ \dot{\varphi} \end{bmatrix} + \begin{bmatrix} -\sqrt{3} \cdot b_1 & 0 & \sqrt{3} \cdot b_1 \\ b_1 & -2 \cdot b_1 & b_1 \\ b_2 & b_2 & b_2 \end{bmatrix} \cdot \begin{bmatrix} U_1 \\ U_2 \\ U_3 \end{bmatrix} \quad (4.3)$$

Mit Hilfe der inversen Transformationsmatrix $\mathbf{T}^{-1}$ (Gl. 2.22) kann die Robotergeschwindigkeit durch die Geschwindigkeit der einzelnen Räder repräsentiert werden.

$$\mathbf{T}^{-1} \cdot \begin{bmatrix} \dot{\omega}_1 \\ \dot{\omega}_2 \\ \dot{\omega}_3 \end{bmatrix} = \begin{bmatrix} a_1 & 0 & 0 \\ 0 & a_1 & 0 \\ 0 & 0 & a_2 \end{bmatrix} \cdot \mathbf{T}^{-1} \cdot \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} + \frac{i}{r} \begin{bmatrix} -\sqrt{3} \cdot b_1 & 0 & \sqrt{3} \cdot b_1 \\ b_1 & -2 \cdot b_1 & b_1 \\ b_2 & b_2 & b_2 \end{bmatrix} \cdot \begin{bmatrix} U_1 \\ U_2 \\ U_3 \end{bmatrix} \quad (4.4)$$

Durch die rechtsseitige Multiplikation mit der Transformationsmatrix $\mathbf{T}$ ergibt sich folgendes entkoppeltes System:

$$\begin{bmatrix} \dot{\omega}_1 \\ \dot{\omega}_2 \\ \dot{\omega}_3 \end{bmatrix} = \begin{bmatrix} a_1 & 0 & 0 \\ 0 & a_1 & 0 \\ 0 & 0 & a_2 \end{bmatrix} \cdot \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} + \frac{i}{r} \begin{bmatrix} 3 \cdot b_1 & 0 & 0 \\ 0 & 3 \cdot b_1 & 0 \\ 0 & 0 & 3 \cdot b_2 \cdot l \end{bmatrix} \cdot \begin{bmatrix} U_1 \\ U_2 \\ U_3 \end{bmatrix} \quad (4.5)$$

Die Geschwindigkeit des Roboters kann anhand der Radgeschwindigkeiten geregelt werden. Dazu werden drei separate Eingrößenregler mit folgender Struktur verwendet:

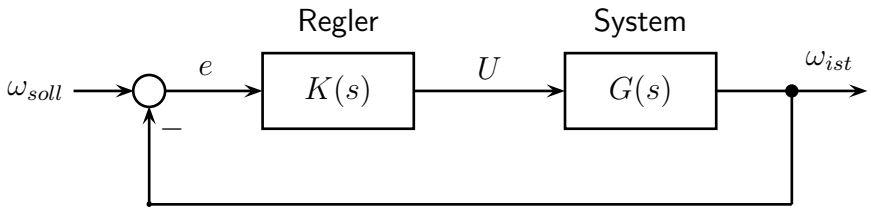


Abbildung 4.3: Regelkreis Geschwindigkeitsregler

Als Regler dient ein PID-Regler. Dieser lässt sich durch folgende Übertragungsfunktion beschreiben:

$$K(s) = \frac{k_p \cdot s + k_i + k_d \cdot s^2}{s} \tag{4.6}$$

Zur Auslegung passender Regelparameter müssen die Systemparameter identifiziert werden. Diese Aufgabe umfasst mehrere aufwendige Verfahren und ist nicht Teil dieser Arbeit. Um dennoch mit geeigneten Regelparametern zu arbeiten, werden die voreingestellten Parameter der bereits vorhandenen Motoransteuerung des Robotino® genutzt.

Zur Identifizierung der Dynamik $D(s)$ des geschlossenen Regelkreises werden Sprungantworten für die translatorische und rotatorische Bewegung aufgenommen. Diese können dem Anhang B entnommen werden. Anhand der Sprungantworten kann ein PT1-Verhalten mit einer Zeitkonstanten T von $0.05\ \mathrm{s}$ und einer Verstärkung k von 0.9 ermittelt werden.

$$D(s) = \frac{k}{1 + T \cdot s} \tag{4.7}$$

Die Übertragunsfunktion besitzt nur eine negative Polstelle. Dadurch lässt sich die Stabilität des inneren Regelkreises feststellen.

4.2.2 Positionsregler

Die Aufgabe der Positionsregelung ist es, Sollgrößen für die Geschwindigkeitsregelung zu generieren. Dabei muss sichergestellt werden, dass sich zu jedem Zeitpunkt t der Robotino® auf der vorgegebenen Bahn befindet. Die benötigten Geschwindigkeiten für die translatorische und rotatorische Bewegung ergeben sich aus der Ableitung des Ortsvektors $\vec{r}(t)$ (Gl. 3.49).

$$^{w}\dot{x}_j(t) = \dot{s}(t) \cdot (b_{j,x} + 2 \cdot c_{j,x} \cdot (s(t) - s_{j-1}) + 3 \cdot d_{j,x} \cdot (s(t) - s_{j-1})^2) \tag{4.8}$$

$$^{w}\dot{y}_j(t) = \dot{s}(t) \cdot (b_{j,y} + 2 \cdot c_{j,y} \cdot (s(t) - s_{j-1}) + 3 \cdot d_{j,y} \cdot (s(t) - s_{j-1})^2) \tag{4.9}$$

$$\dot{\varphi}(t) = arctan\left(\frac{^{w}\dot{y}_j(t)}{^{w}\dot{x}_j(t)}\right) \tag{4.10}$$

Unter idealen Bedingungen würde mit Hilfe von 4.8, 4.9 und 4.10 ein Abfahren der vorgegebenen Bahn möglich sein.

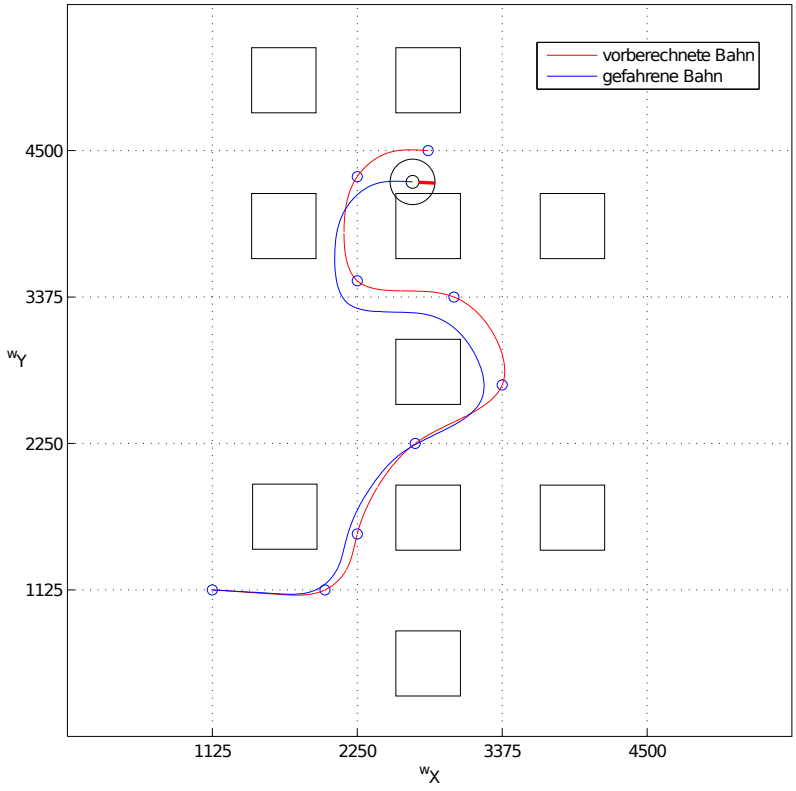


Abbildung 4.4: Fahrt auf Basis der Vorsteuerung

Abbildung 4.4 zeigt eine Fahrt auf Basis der berechneten Sollgeschwindigkeiten. Es ist eine deutliche Abweichung zwischen der vorgegebenen roten Bahn und gefahrenen blauen Bahn zu erkennen. Aufgrund von Störgrößen entstehen Bahnfehler. Diese können durch Vergleich der aktuellen mit der gewünschten Pose bestimmt werden.

Der Positionsregler versucht die Bahnfehler zu minimieren. Dazu werden die berechneten Geschwindigkeiten (Gl. 4.8, 4.9 u. 4.10) als Vorsteuerung genutzt und mit entsprechenden Korrekturwerten beaufschlagt. Dies kann zu einer höheren Geschwindigkeit als geplant führen und muss bei der Anwendung des Geschwindigkeitsprofils (Abschn. 3.3.2) beachtet werden. Wird die vorgegebene Robotergeschwindigkeit zu groß gewählt, stoßen die Aktoren an ihre Grenzen und die Regelung ist nicht mehr in der Lage, Bahnfehler auszugleichen. Mit Hilfe der inversen und direkten Kinematik (Gl. 2.20 und Gl. 2.21) kann folgender Regelkreis für den Positionsregler entworfen werden:

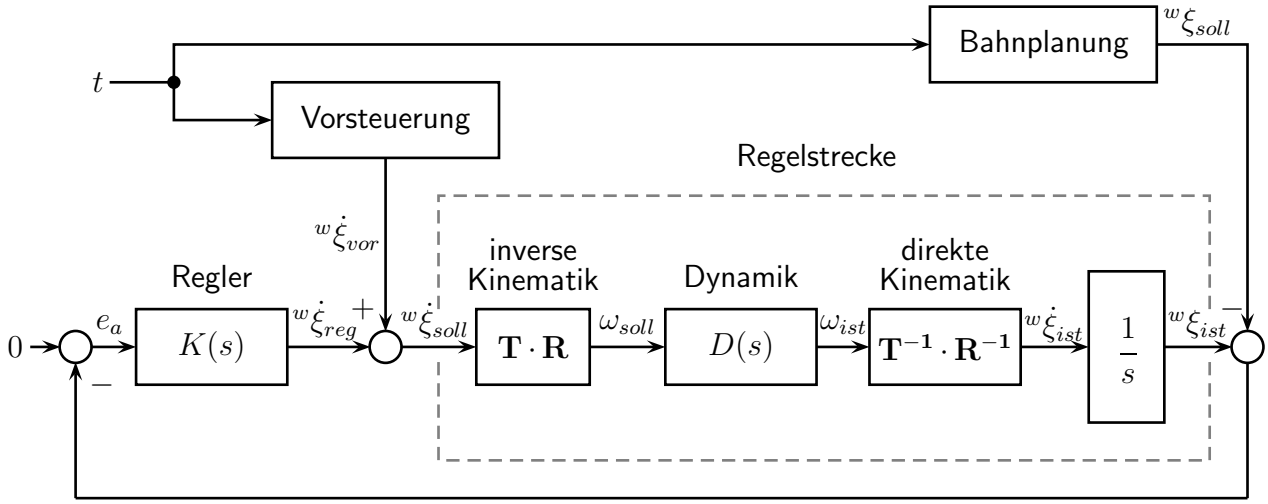


Abbildung 4.5: Regelkreis Positionsregler

Folgendes Übertragungsverhalten für die Regelstrecke des Positionsreglers wird ermittelt:

$$G(s) = \frac{1}{s} \cdot D(s) = \frac{k}{s \cdot (1 + T \cdot s)} \tag{4.11}$$

Es handelt sich um einen verzögerten Integrator (IT1-Glied). Zur Regelung dieses Verhaltens wird ein P-Regler mit der Regelverstärkung k_p verwendet.

$$K(s) = k_p \tag{4.12}$$

Bei der Auswahl einer geeigneten Reglerverstärkung handelt es sich um ein iteratives Verfahren. Die getroffenen Entscheidungen können solange revidiert werden, bis ein befriedigender Regler gefunden ist [Lun10a]. Zur groben Einstellung der Regelverstärkung wird das Übertragungsverhalten des geschlossenen Kreises $G_g(s)$ ermittelt.

$$G_g(s) = \frac{G(s) \cdot K(s)}{1 + G(s) \cdot K(s)} = \frac{1}{1 + \frac{1}{k_p \cdot k} \cdot s + \frac{T}{k_p \cdot k} \cdot s^2} \tag{4.13}$$

Es lässt sich ein PT2-Verhalten erkennen. Zur Bestimmung der Zeitkonstante $\hat{T}$ und Dämpfung

$\hat{D}$ des geschlossenen Kreises wird folgende Beziehung mittels Koeffizientenvergleich aufgestellt.

$$G_g(s) = \frac{1}{1 + 2 \cdot \hat{D} \cdot \hat{T} + \hat{T^2} \cdot s^2} \tag{4.14}$$

mit

$$\hat{T^2} = \frac{T}{k_p \cdot k_i} \tag{4.15}$$

$$2 \cdot \hat{D} \cdot \hat{T} = \frac{1}{k_p \cdot k} \tag{4.16}$$

Mit Hilfe von k_p kann einer der Parameter des PT2 vorgegeben werden. Um eine ruhige Fahrt zu gewährleisten, wird die Dämpfung so gewählt, dass sich der aperiodische Grenzfall einstellt.

$$k_p = \frac{1}{4 \cdot \hat{D} \cdot T \cdot k} = \frac{1}{4 \cdot 1 \cdot 0,05 \cdot 0,9} \approx 5,56 \tag{4.17}$$

Das beste Regelverhalten ist bei einer Reglerverstärkung von $5,56$ zu erwarten. Um die Stabilität des Systems zu bestimmen, wird das charakteristische Polynom $C(s)$ betrachtet.

$$C(s) = 1 + 0,2 \cdot s + 0,001 \cdot s^2 = 0.001 \cdot (s + 194,87) \cdot (s + 5,13) \tag{4.18}$$

Die Polstellen des Systems befinden sich in der linken Halbebene. Somit ist das System stabil. Trotz der Stabilität des Systems sollte zur Sicherheit bei Fehlzuständen oder großen Messfehlern der Reglerausgang begrenzt werden.

5 Umsetzung

5.1 Einleitung

Die Algorithmen für die Bahnplanung und Bahnführung wurden in 2 Schritten umgesetzt. Im ersten wurde eine Implementierung in Matlab vorgenommen. Dieser Schritt diente der einfachen Umsetzung und schnellen Evaluierung der Algorithmen. Matlab verfügt über eine Vielzahl vordefinierter mathematischer Routinen und eine einfache grafische Aufarbeitung von Messwerten. Im zweiten Schritt wurde der entstandene Quellcode in C++ überführt, um in das bestehende System des Team robOTTO integriert zu werden. Ein Ausführen des C++ Codes außerhalb des Systems ist nicht möglich.

Beide Umsetzungen nutzen dabei das von der Firma REC entwickelte Application-Programming-Interface (API), um mit dem Robotino® zu kommunizieren und hinterlegte Befehlsroutinen auszuführen. Der vollständige Code sowie die für die Nutzung benötigte API befinden sich im Anhang D.

5.2 Umsetzung in Matlab

Die Programmierung des Roboters erfolgte durch die schrittweise Umsetzung der einzelnen Module. Zur Evaluierung wurde mit Hilfe der in Matlab integrierten „Graphical User Interface Development Environment“ (kurz GUIDE) eine grafische Oberfläche erstellt (Abb. 5.1). Mit dieser ist es möglich, das Verhalten des Roboters zu interpretieren. Dazu werden Sensordaten zu jedem Zeitpunkt offengelegt und grafisch aufgearbeitet.

Mit Hilfe der GUI können die Bahnplanung und Bahnführung durchgeführt werden. Dazu können alle nötigen Parameter vorgegeben und manipuliert werden. Gestartet wird die GUI mit dem Matlab-File *ControlCenter.m*. Für die Nutzung wichtige Informationen können in Form von *readme*-Dateien dem Anhang D entnommen werden.

Zur Implementierung der Bahnplanung mussten das lineare Gleichungssystem in Gleichung 3.22 sowie das Integral in Gleichung 3.32 gelöst werden. Dies geschah unter Verwendung des Backslash-Operator und der Quadratur-Funktion von Matlab. Nähere Beschreibungen dieser Verfahren können der Quelle [Mat12] entnommen werden.

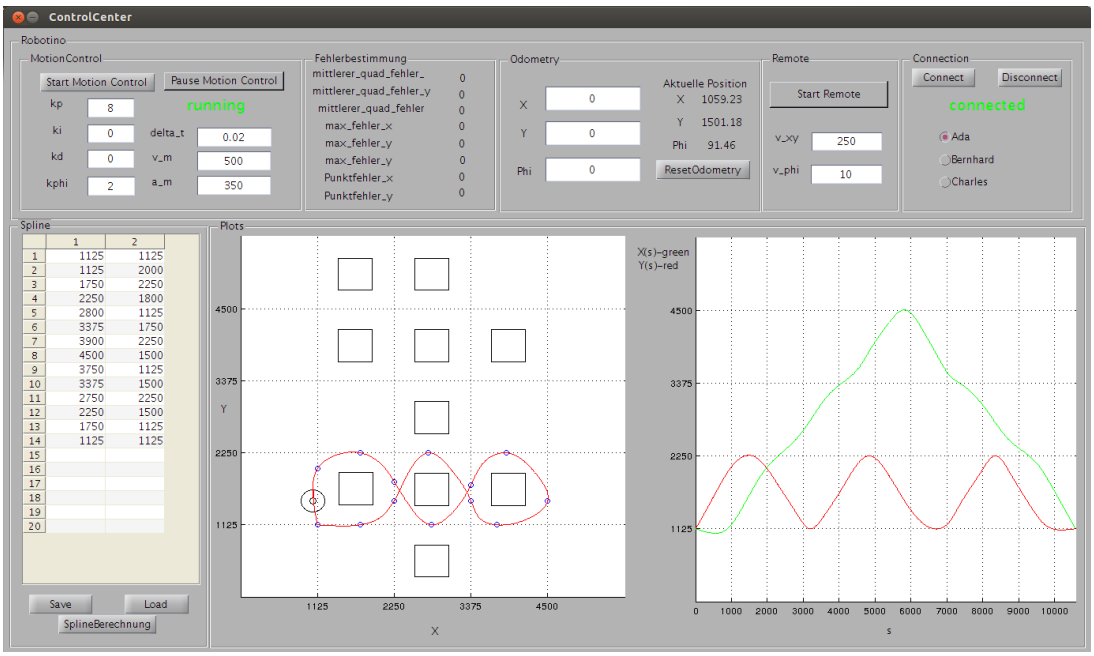


Abbildung 5.1: Matlab-GUI

Die Umsetzung der Bahnführung erforderte die Diskretisierung der zeitkontinuierlichen Regler aus Kapitel 4. Dies konnte mit Hilfe der bilinearen Transformation realisiert werden. Dazu wurde aus dem kontinuierlichen Regler $K(s)$, durch Substitution der Variable s, der zeitdiskrete Regler $K(z)$ entwickelt. Folgende Substitution wurde verwendet [Kas10a]:

$$s = \frac{2}{\Delta t} \cdot \frac{1 - z^{-1}}{1 + z^{-1}} \tag{5.1}$$

Mit Hilfe von Gl. 5.1 kann die Diskretisierung durchgeführt werden.

$$K(s) = k_p + \frac{k_i}{s} + k_d \cdot s \quad \rightarrow \quad K(z) = k_p + \frac{k_i \cdot \Delta t}{2} \cdot \frac{1 + z^{-1}}{1 - z^{-1}} + \frac{2 \cdot k_d}{\Delta t} \cdot \frac{1 - z^{-1}}{1 + z^{-1}} \tag{5.2}$$

Durch Anwendung der inversen z-Transformation ergeben sich folgende Differenzengleichungen mit dem Reglereingang u und Reglerausgang y.

$$y = k_p \cdot u(j) + k_i \cdot y_i(j) + k_d \cdot y_d(j) \tag{5.3}$$

mit

$$y_i(j) = \frac{\Delta t}{2} \cdot (u(j) + u(j-1)) + y_i(j-1) \tag{5.4}$$

$$y_d(j) = \frac{2}{\Delta t} \cdot (u(j) - u(j-1)) - y_i(j-1) \tag{5.5}$$

Um Approximationsfehler zu vermeiden, ist Δt hinreichend klein zu wählen. Dabei muss gewährleistet sein, dass die Zeit ausreichend ist, um erforderliche Berechnungen durchzuführen.

5.3 Umsetzung in C++

Die Umsetzung in C++ erforderte die Implementierung eigener Routinen für die bisher genutzten mathematischen Funktionen von Matlab. Davon unabhängig konnte der Rest äquivalent übernommen werden.

Zur Lösung des linearen Gleichungssystems in Gleichung 3.22 wurde der Thomas-Algorithmus verwendet [Wes08]. Dabei handelt es sich um eine vereinfachte Form des Gaußschen Eliminationsverfahrens, welche die besondere Form der Tridiagonalmatrix berücksichtigt. Dadurch wird die benötigte Schrittzahl zur Findung der Lösung verringert.

Es wird folgendes Gleichungssystem betrachtet:

$$\begin{bmatrix} a_1 & b_1 & 0 & \dots & & & \\ b_1 & a_2 & b_2 & 0 & \dots & & \\ 0 & b_2 & a_3 & b_3 & & & \\ \dots & & & & & & \\ & & & & & & \\ & & & \dots & 0 & b_{n-1} & a_n \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ x_n \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ . \\ . \\ . \\ c_n \end{bmatrix} \tag{5.6}$$

Die Form der Matrix vereinfacht die Gauß-Elimination zu:

$$d = b_{j-1}/a_{j-1} \tag{5.7}$$

$$a_j = a_j - d \cdot b_{j-1} \qquad \text{für} \quad j = 2 \dots n \tag{5.8}$$

$$c_j = c_j - d \cdot c_{j-1} \tag{5.9}$$

Die Lösung des Systems ergibt sich durch das Rückwärtsauflösen.

$$x_n = \frac{c_n}{a_n} \tag{5.10}$$

$$x_j = \frac{c_j - b_j \cdot x_{j+1}}{a_j} \qquad \text{für} \quad j = n-1 \dots 1 \tag{5.11}$$

Zur Implementierung des Integrals in Gleichung 3.32 wurde die zuvor entwickelte Methode zur Bestimmung der integralen Regelabweichung (Gl. 5.4) verwendet. Dazu wurde die Abtastzeit Δt durch die Abtastweite Δd ersetzt. Diese bestimmt die Schrittweite der Abtastung der Funktion. Je kleiner diese gewählt wird, desto genauer wird das Ergebnis und desto höher wird die Berechnungsdauer.

6 Ergebnisse

6.1 Einleitung

Es konnten Trajektorien für die Bewegung des Roboters in x- und y-Richtung sowie für die Drehbewegung erzeugt werden. Mit deren Hilfe kann die zeitliche Einordnung der Pose des Roboters in eine Bewegungsbahn vorgenommen werden. Die Gestalt dieser Bahn erfüllt die an sie gestellten Anforderungen. Weiterhin konnte mit Hilfe eines Kaskadenreglers der Robotino® entlang der Trajektorien geführt werden.

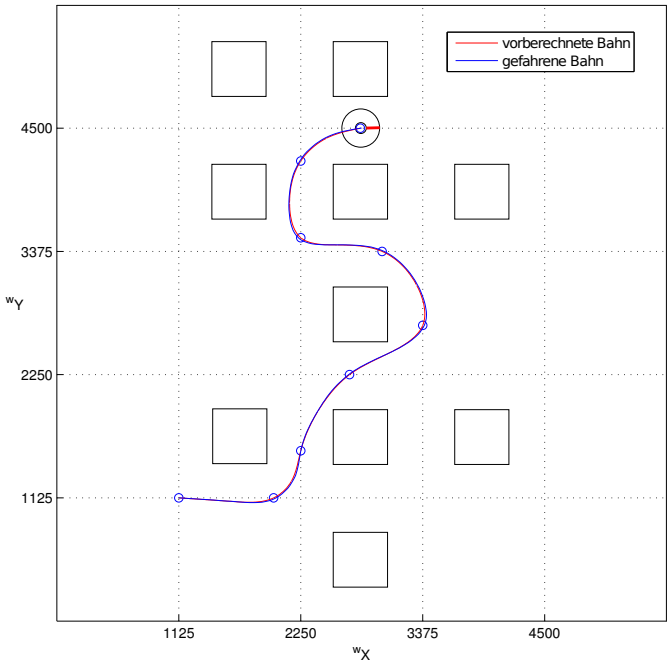


Abbildung 6.1: Verlauf der Bahnführung des Robotino®

Abbildung 6.1 zeigt den Verlauf einer berechneten Bahn (rot) und den Bewegungsverlauf (blau) des Robotino®. Es ist zu erkennen, dass die Bahnführung in der Lage ist die Vorgaben mit hoher Genauigkeit umzusetzen.

6.2 Validierung des Algorithmus

Die Validierung der Bahnführung erfolgte durch das mehrfache Abfahren vordefinierterTeststrecken. Dabei wurde eine Geschwindigkeit von $500\ \mathrm{mm/s}$ und eine Beschleunigung von $350\ \mathrm{mm/s^2}$ vorgegeben. Die Reglerverstärkung k_p wurde mit acht bestimmt. Bei diesem Wert und einer Abtastzeit von $0.02\ \mathrm{s}$ konnte die höchste Güte der Bahn erzielt werden, ohne ein Überschwingen des Systems zu erlauben. Dies ermöglichte eine ruhige Fahrt. Als Gütekriterien dienten der mittlere quadratische Bahnfehler, die maximale Bahnabweichung und die Abweichung vom Zielpunkt. Bei der Validierung wurde deutlich, dass der maximale sowie der mittlere quadratische Fehler von der Form der Bahn abhängen, die Genauigkeit der Zielankunft davon jedoch unbeeinflusst ist. Auch war die Bahngüte unabhängig vom Ladezustand des Robotino®.

Die Teststrecken weisen eine deutlich höhere Krümmung als die regulär im Spiel befahrenen Bahnen auf. Dadurch wird die Anforderung an die Bahnführung absichtlich erhöht. Erzielte Ergebnisse sind somit schlechter als zu erwarten. Der maximale Bahnfehler betrug $20,44\ \mathrm{mm}$ und der Zielpunkt konnte mit einer Abweichung von $\pm 1.04\ \mathrm{mm}$ angefahren werden. Die aufgenommenen Testreihen sind im Anhang C einzusehen.

6.3 Einschränkungen

Die verwendeten Geschwindigkeitsprofile zielen darauf ab, das Anfahren und Abbremsen des Roboters zu regeln. Zwischen diesen Phasen wird eine konstante Geschwindigkeit eingestellt. Diese wird mit Hilfe der inversen Kinematik auf Motorgeschwindigkeiten umgerechnet. Bei einer Kurvenfahrt ändern sich zwar die Motorgeschwindigkeiten, nicht jedoch die absolute Geschwindigkeit des Roboters. Je nach Bahnkrümmung und vorgegebener Fahrgeschwindigkeit kann diese Änderung groß genug sein, um Schlupf zu verursachen. Dies führt zu Odometriefehlern und somit zu Abweichungen zwischen der Bahnplanung und Bahnführung. Die entstandene Abweichung kann während der Fahrt nicht mit der vorhandenen Messtechnik aufgenommen werden. Eine Korrektur ist erst nach Erreichen des Zielpunktes mit Hilfe des Laserscanners realisierbar. Um eine durch starke Abweichungen mögliche Kollision zu vermeiden, sollte die Geschwindigkeit entsprechend der Krümmung der Bahn angepasst werden. Dazu muss die Gestalt der Bahn in die Entwicklung des Geschwindigkeitsprofils einfließen. Ein einfaches Abbremsen zur Laufzeit ist nicht zweckmäßig, da dadurch die Transportzeit nicht vorhersagbar bleibt. Der in dieser Arbeit vorgestellte Algorithmus berücksichtigt nicht die Krümmung der Bahn, wodurch die maximal erreichbare Geschwindigkeit eingeschränkt ist.

6.4 Schlussfolgerung

Schlussfolgernd kann festgestellt werden, dass der entwickelte Algorithmus eine deutliche Verbesserung des bestehenden darstellt. Die maximale Fahrgeschwindigkeit konnte von $300\ \mathrm{mm/s}$ auf $500\ \mathrm{mm/s}$ erhöht werden. Zusätzlich konnte durch die Wahl der Interpolationsart die Gestalt der Bahn so verändert werden, dass ein Anhalten in den Zwischenpunkten unnötig wurde. Dies führte trotz einer Verlängerung der Wegstrecke zu einer deutlich geringeren Transportzeit.

Die erreichte Bahngüte ermöglicht es, bei der Wegfindung verwendete Sicherheitsabstände zu verringern. Dadurch erhöht sich die Anzahl möglicher Wege zum Zielpunkt und ein flexiblerer Produktionsablauf kann erreicht werden. Des Weiteren wird die Wegfindung in einem Multiagentensystem durch die zeitliche Vorhersehbarkeit der Transportwege vereinfacht. Die geschaffene Transparenz ermöglicht auch eine Optimierung der Produktionsplanung.

Durch die Anwendung sinoider Geschwindigkeitsprofile konnten der Schlupf und die Beanspruchung des Materials verringert werden. Dies gewährleistet die Vertrauenswürdigkeit der durch Odometrie ermittelten Pose und die Langlebigkeit der Antriebseinheiten.

7 Zusammenfassung und Ausblick

7.1 Zusammenfassung

In dieser Arbeit wurden Algorithmen für die Bahnplanung und Bahnführung eines mobilen Robotersystems mit omnidirektionalem Antrieb entwickelt und exemplarisch auf einem Robotino® umgesetzt.

Im Zuge der Bahnplanung wurden die Anforderungen an die Gestalt der Bahnkurve besprochen und natürliche kubische Splines als die optimale Lösung für das bestehende Problem ermittelt. Mit Hilfe der getroffenen Auswahl konnten Trajektorien für die Pose des Roboters entwickelt werden. Diese unterliegen einem sinoiden Geschwindigkeitsprofil. Dadurch konnte sowohl der Schlupf als auch die Belastung der Mechanik minimiert werden. Zur Beschreibung des Robotino® wurde dieser hinsichtlich seiner Kinematik und Dynamik untersucht. Dabei wurden äquivalente mathematische Modelle erstellt. Auf Basis der Modelle wurde ein Kaskadenregler zur Bahnführung entworfen. Dieser ist in der Lage, die Geschwindigkeits- und Positionsregelung unabhängig voneinander zu bearbeiten.

Mit Hilfe der entstandenen Algorithmen konnte die Transportzeit verringert und die Güte der Bahnführung erhöht werden. Des Weiteren ist es möglich, Transportzeiten vorherzusagen.

7.2 Ausblick

Trotz der Anwendung von sinoiden Geschwindigkeitsprofilen ist das Schlupfen der Räder nicht gänzlich auszuschließen. Dies führt zum einen zu Odometriefehlern und zum anderen zu einer Verschlechterung der Traktion. Es gilt also Schlupf zu minimieren. Dies kann z.B mit Hilfe einer Schlupfregelung erreicht werden. Dazu muss das Durchdrehen der Räder erkannt und durch gezielte Ansteuerung der Motoren beseitigt werden. Vegleichbar ist ein solches Verfahren mit der aus der Automobilbranche bekannten Anti-Schlupf-Regelung (ASR).

Um ein schlupfendes Rad zu detektieren, wird die tatsächliche Geschwindigkeit des Roboters mit der aus den Radgeschwindigkeiten berechneten verglichen. Ziel der Schlupfregelung ist es, den Geschwindigkeitsunterschied zu beseitigen. Zur Ermittlung der tatsächlichen Geschwindigkeit können Rechenmodelle oder zusätzliche Messtechnik verwendet werden.

Ein vielversprechender Ansatz ist das Aufnehmen der Bewegung des Roboters mit Hilfe optischer Sensoren, wie sie z.B. in Computermäusen verwendet werden. Dabei nimmt der Sensor in Form von Graustufenbildern den Untergrund kontinuierlich auf. Durch Vergleich aufeinander folgender Bilder können Rückschlüsse auf die zurückgelegte Strecke und Geschwindigkeit des Roboters gezogen werden. Die erhaltenen Informationen sind vom Schlupf der Räder unabhängig und können somit als Referenz genutzt werden. Diese Technologie ist preiswert, einfach umzusetzen und besitzt eine hohe Genauigkeit.

Literaturverzeichnis

[AHK+09] Arens, Tilo ; Hettlich, Frank ; Karpfinger, Christian ; Kockelkorn, Ulrich ; Lichtenegger, Klaus ; Stachel, Hellmuth: *Mathematik*. 1. Aufl. 2008. Korr. Nachdruck 2010. Spektrum Akademischer Verlag, 2009. – ISBN 3827417589

[Cho05] Choset, Howie: *Principles of Robot Motion*. Cambridge : MIT Press, 2005. – ISBN 9780262033275

[Cra05] Craig, John: *Introduction to robotics : mechanics and control*. Upper Saddle River, N.J : Pearson/Prentice Hall, 2005. – ISBN 0131236296

[Did12] Didactic, Festo: *Logistics League Rulebook*. Bd. 1.805.2012. Festo, 2012

[ES10] El-Shenawy, A.: *Motion Control of Holonomic Wheeled Mobile Robot with Modular Actuation*, Universitätsbibliothek der Universität Mannheim, Diss., 2010. http://deposit.ddb.de/cgi-bin/dokserv?idn=1002189764&dok_var=d1&dok_ext=pdf&filename=1002189764.pdf

[Fes07] Festo: *Robotino Handbuch*. Festo Didactic GmbH & Co. KG, 2007 www.festo-didactic.com

[FH07] Freund, Roland W. ; Hoppe, Ronald W.: *Stoer/Bulirsch: Numerische Mathematik 1*. 10., neu bearb. Aufl. Springer Berlin Heidelberg, 2007. – ISBN 354045389X

[FW06] Feldmann, Klaus ; Wolf, Wolfgang ; Levi, Paul (Hrsg.) ; Schanz, Michael (Hrsg.) ; Lafrenz, Reinhard (Hrsg.) ; Avrutin, Viktor (Hrsg.): *Autonome Mobile Systeme*. Springer Berlin Heidelberg, 2006 (Informatik aktuell). http://www.springerlink.com/content/p7hk0258551865v7/abstract/. – ISBN 978–3–540–30292–6

[Her12] Hertzberg, Joachim: *Mobile Roboter: Eine Einführung aus Sicht der Informatik (eXamen.press)*. 2012. Springer, 2012. – ISBN 3642017258

[Ise06] Isermann, Rolf: *Fahrdynamik-Regelung: Modellbildung, Fahrerassistenzsysteme, Mechatronik*. 2006. Vieweg+Teubner Verlag, 2006. – ISBN 3834801097

[Kas09] Kasper, Roland: *Vorlesung: Mechatronik I*. IMS, 2009

[Kas10a] Kasper, Roland: *Vorlesung: Eingebettete Systeme*. IMS, 2010

[Kas10b] Kasper, Roland: *Vorlesung: Mechatronik II*. IMS, 2010

[Ler11] Lerez, Christoph: *Lokalisierung mobiler Roboter mittels 2D- Laser Range Finder in strukturierter Umgebung*, Otto-von-Guericke-Universität Magdeburg, Studienarbeit, 2011

[Lun10a] Lunze, Jan: *Regelungstechnik 1: Systemtheoretische Grundlagen, Analyse und Entwurf einschleifiger Regelungen (Springer-Lehrbuch)*. 8., neu bearb. Aufl. Springer, 2010. – ISBN 3642138071

[Lun10b] Lunze, Jan: *Regelungstechnik 2: Mehrgrößensysteme, Digitale Regelung (Springer-Lehrbuch)*. 6., neu bearb. Aufl. Springer, 2010. – ISBN 3642101976

[LWJZL03] Liu, Y. ; Wu, X. ; Jim Zhu, J. ; Lew, J.: Omni-directional mobile robot controller design by trajectory linearization. In: *American Control Conference, 2003. Proceedings of the 2003* Bd. 4, 2003, 3423–3428

[LWZ07] Li, X. ; Wang, M. ; Zell, A.: Dribbling control of omnidirectional soccer robots. In: *Robotics and Automation, 2007 IEEE International Conference on*, 2007, 2623–2628

[Lü12] Lüder, Arndt: *Vorlesung: Industrieroboter*. IMS, 2012

[Mat12] MathWorks: *Matlab Documentation R2012b*. MathWorks, 2012

[NXST08] Naeem, W. ; Xu, T. ; Sutton, R. ; Tiano, A.: The design of a navigation, guidance, and control system for an unmanned surface vehicle for environmental monitoring. In: *Proceedings of the Institution of Mechanical Engineers, Part M: Journal of Engineering for the Maritime Environment* 222 (2008), Nr. 2, 67. http://pim.sagepub.com/content/222/2/67.short

[Pla06] Plato, Robert: *Numerische Mathematik kompakt*. Vieweg, 2006 http://www.springerlink.com/content/r50p27k04512622u/abstract/. – ISBN 978–3–8348–9059–7

[Sch01] Schulte, Gerd: *Material- und Logistikmanagement*. Oldenbourg Verlag, 2001. – ISBN 9783486254587

[Sch09] Schmucker, Ulrich: *Vorlesung: Grundlagen mobiler Roboter*. IMS, 2009

[SF96] Schmidt, Günther ; Freyberger, Franz: *Autonome Mobile Systeme 1996: 12. Fachgespräch München, 14.-15. Oktober 1996*. 1. Springer, 1996. – ISBN 3540617515

[SK08] Siciliano, Bruno ; Khatib, Oussama: *Springer Handbook of Robotics*. 1. Springer Berlin Heidelberg, 2008. – ISBN 354023957X

[SN04] Siegwart, Roland ; Nourbakhsh, Illah R.: *Introduction to Autonomous Mobile Robots*. New. Mit Pr, 2004. – ISBN 026219502X

[Som12] Sommer, Erik: *Erarbeitung und Implementierung eines globalen Orientierungsansatzes für eine autonome Roboterplattform in einer zweidimensionalen Umgebung*, Otto-von-Guericke-Universität Magdeburg, Bachelorarbeit, 2012

[Web07] Weber, Wolfgang: *Industrieroboter: Methoden der Steuerung und Regelung*. 2., neu bearbeitete Auflage. Carl Hanser Verlag GmbH & CO. KG, 2007. – ISBN 3446410317

[Wei10] Weist, Norbert: *Vorlesung: Mechanische Antriebssystem*. IMS, 2010

[Wes08] Westermann, Thomas: *Mathematik für Ingenieure: Ein anwendungsorientiertes Lehrbuch*. 5., neu bearb. Aufl. Springer Berlin Heidelberg, 2008. – ISBN 354077730X

[Wu05] Wu, J.: *Dynamic path planning of an omni-directional robot in a dynamic environment*, Diss., 2005. http://etd.ohiolink.edu/view.cgi/ris.cgi?acc_num=ohiou1113839523

A Maple-Code zur Aufstellung des dynamischen Systems

```
restart;
with(LinearAlgebra),
with(linalg);
```

Transformationsmatrix

$$T := Matrix\left(\left[\left[-\frac{\sqrt{3}}{2}, 0.5, l\right], [0, -1, l], \left[\frac{\sqrt{3}}{2}, 0.5, l\right]\right]\right);$$

$$\begin{bmatrix} -\frac{1}{2}\sqrt{3} & 0.5 & l \\ 0 & -1 & l \\ \frac{1}{2}\sqrt{3} & 0.5 & l \end{bmatrix} \quad (1)$$

Kräftegleichgewicht

$$\mathrm{Ff} := Matrix\left(\left[\left[\frac{\sqrt{3}}{2}, 0, -\frac{\sqrt{3}}{2}\right], [-0.5, 1, -0.5], [1, 1, 1]\right]\right);$$

$$\begin{bmatrix} \frac{1}{2}\sqrt{3} & 0 & -\frac{1}{2}\sqrt{3} \\ -0.5 & 1 & -0.5 \\ l & l & l \end{bmatrix} \quad (2)$$

Geschwindigkeitsvektor

xdot = Geschwindigkeit in x Richtung des Roboters
ydot = Geschwindigkeit in y Richtung des Roboters
phidot = Drehgeschwindigkeit des Roboters

$$v := Matrix([[xdot], [ydot], [phidot]]);$$

$$\begin{bmatrix} xdot \\ ydot \\ phidot \end{bmatrix} \quad (3)$$

Beschleunigungsvektor

xdotdot = Beschleunigung in x Richtung des Roboters
ydotdot = Beschleunigung in y Richtung des Roboters
phidotdot = Drehbeschleunigung des Roboters

$$vdot := Matrix([[xdotdot], [ydotdot], [phidotdot]]);$$

$$\begin{bmatrix} xdotdot \\ ydotdot \\ phidotdot \end{bmatrix} \quad (4)$$

Dynamik des Getriebemotors

$$f := \frac{1}{r}\left(J_{GM}\cdot\left(\frac{\mathrm{i}}{r}\right)\cdot Multiply(T, vdot) + \left(d + \frac{c_m\cdot c_e\cdot i}{R}\right)\cdot\left(\frac{\mathrm{i}}{r}\right)\cdot Multiply(T, v) - \frac{c_m}{R}\cdot Matrix([[u1],[u2],[u3]])\right);$$

$$\left[\left[\frac{1}{r}\left(\frac{J_{GM}\, i\left(-\frac{1}{2}\sqrt{3}\ xdotdot + 0.5\ ydotdot + l\ phidotdot\right)}{r} + \frac{\left(d + \frac{c_m\, c_e\, i}{R}\right) i\left(-\frac{1}{2}\sqrt{3}\ xdot + 0.5\ ydot + l\ phidot\right)}{r} - \frac{c_m\, u1}{R}\right)\right], \left[\frac{J_{GM}\, i\ (-ydotdot + l\ phidotdot)}{r} + \frac{\left(d + \frac{c_m\, c_e\, i}{R}\right) i\ (-ydot + l\ phidot)}{r} - \frac{c_m\, u2}{R}\right], \left[\frac{1}{r}\left(\frac{J_{GM}\, i\left(\frac{1}{2}\sqrt{3}\ xdotdot + 0.5\ ydotdot + l\ phidotdot\right)}{r} + \frac{\left(d + \frac{c_m\, c_e\, i}{R}\right) i\left(\frac{1}{2}\sqrt{3}\ xdot + 0.5\ ydot + l\ phidot\right)}{r} - \frac{c_m\, u3}{R}\right)\right]\right] \tag{5}$$

Trägheitsmatrix

$$H := Matrix\left(\left[\left[\frac{1}{m}, 0, 0\right], \left[0, \frac{1}{m}, 0\right], \left[0, 0, \frac{1}{J_r}\right]\right]\right);$$

$$\begin{bmatrix} \frac{1}{m} & 0 & 0 \\ 0 & \frac{1}{m} & 0 \\ 0 & 0 & \frac{1}{J_r} \end{bmatrix} \tag{6}$$

Corioliskraft

$$G := Matrix([[ydot\cdot phidot], [xdot\cdot phidot], [0]]);$$

$$\begin{bmatrix} ydot\ phidot \\ xdot\ phidot \\ 0 \end{bmatrix} \tag{7}$$

Systemdynamik

$$vdotdot := simplify(G + Multiply(H, Multiply(\mathrm{Ff}, f)));$$

$$\left[\left[-\frac{1}{2}\ \frac{1}{r^2 R m}\left(-2\ ydot\ phidot\ r^2 R m + 3 J_{GM} i R\ xdotdot + 3\ i\ d R\ xdot + 3\ c_m c_e i^2\ xdot + \sqrt{3}\ c_m u1\ r - \sqrt{3}\ c_m u3\ r\right)\right],\right. \tag{8}$$

$$\left[-\frac{1}{r^2 R m}\left(0.5000000000\left(-2.\ xdot\ phidot\ r^2 R m + 2.\ c_m u2\ r - 1.\ c_m u1\ r - 1.\ c_m u3\ r + 3.\ J_{GM} i R\ ydotdot + 3.\ i\ d R\ ydot + 3.\ c_m c_e i^2\ ydot\right)\right)\right],$$

$$\left.\left[\frac{1}{r^2 R J_r}\left(l\left(-c_m u2\ r + 3 J_{GM} i R\ l\ phidotdot + 3\ i\ d R\ l\ phidot + 3\ c_m c_e i^2\ l\ phidot - c_m u1\ r - c_m u3\ r\right)\right)\right]\right]$$

$$xdotdot = collect(expand(solve(xdotdot = vdotdot(1), xdotdot)), xdot);$$

$$xdotdot = \left(-\frac{3\ i\ d}{3 J_{GM} i + 2 r^2 m} - \frac{3 c_m c_e i^2}{R\left(3 J_{GM} i + 2 r^2 m\right)}\right) xdot + \frac{2\ ydot\ phidot\ r^2 m}{3 J_{GM} i + 2 r^2 m} + \frac{\sqrt{3}\ c_m u3\ r}{R\left(3 J_{GM} i + 2 r^2 m\right)} - \frac{\sqrt{3}\ c_m u1\ r}{R\left(3 J_{GM} i + 2 r^2 m\right)} \tag{9}$$

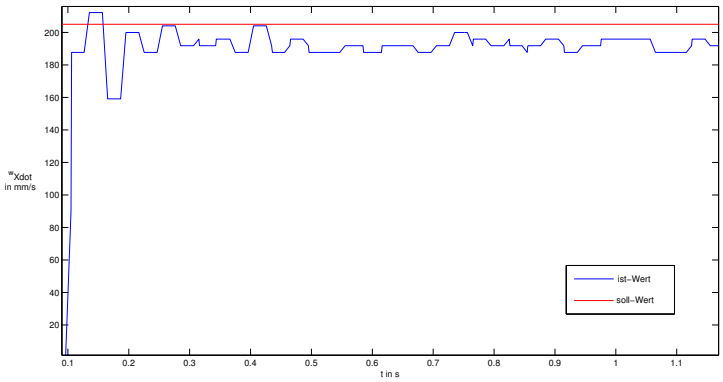
$$ydotdot = collect(expand(solve(ydotdot = vdotdot(2), ydotdot)), ydot);$$

$$ydotdot = \left(-\frac{3.\ i\ d}{3.\ J_{GM} i + 2.\ r^2 m} - \frac{3.\ c_m c_e i^2}{R\left(3.\ J_{GM} i + 2.\ r^2 m\right)}\right) ydot + \frac{2.\ xdot\ phidot\ r^2 m}{3.\ J_{GM} i + 2.\ r^2 m} - \frac{2.\ c_m u2\ r}{R\left(3.\ J_{GM} i + 2.\ r^2 m\right)} + \frac{1.\ c_m u1\ r}{R\left(3.\ J_{GM} i + 2.\ r^2 m\right)} + \frac{1.\ c_m u3\ r}{R\left(3.\ J_{GM} i + 2.\ r^2 m\right)} \tag{10}$$

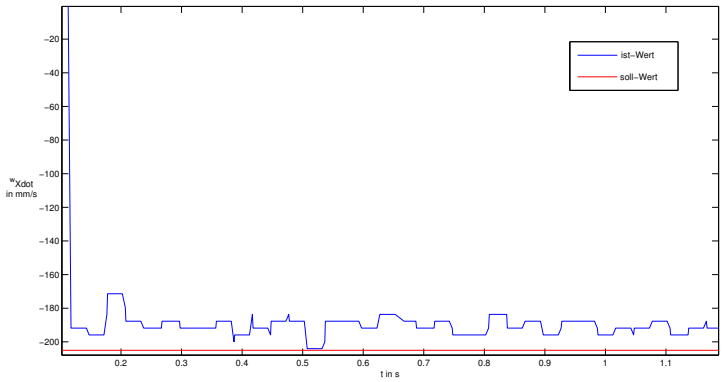
$$phidotdot = collect(expand(solve(phidotdot = vdotdot(3), phidotdot)), phidot);$$

$$phidotdot = \left(-\frac{3\ l^2\ i\ d}{-r^2 J_r + 3\ l^2 J_{GM} i} - \frac{3\ l^2 c_m c_e i^2}{R\left(-r^2 J_r + 3\ l^2 J_{GM} i\right)}\right) phidot + \frac{l\ c_m u2\ r}{R\left(-r^2 J_r + 3\ l^2 J_{GM} i\right)} + \frac{l\ c_m u3\ r}{R\left(-r^2 J_r + 3\ l^2 J_{GM} i\right)} + \frac{l\ c_m u1\ r}{R\left(-r^2 J_r + 3\ l^2 J_{GM} i\right)} \tag{11}$$

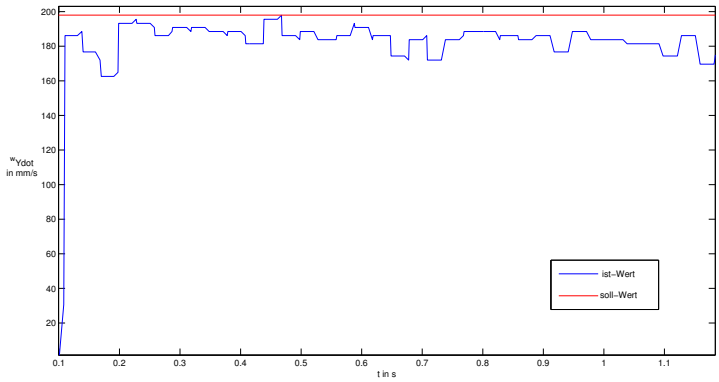
B Sprungantworten des Robotino®



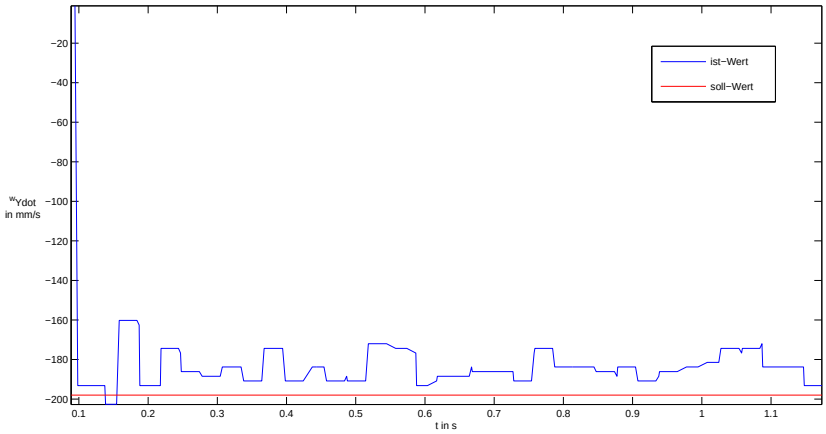
(a) positiver Sprung der x-Geschwindigkeit



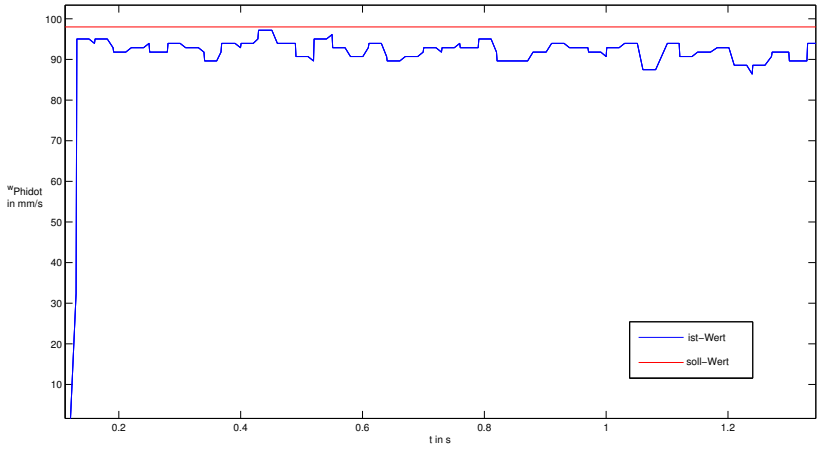
(b) negativer Sprung der x-Geschwindigkeit



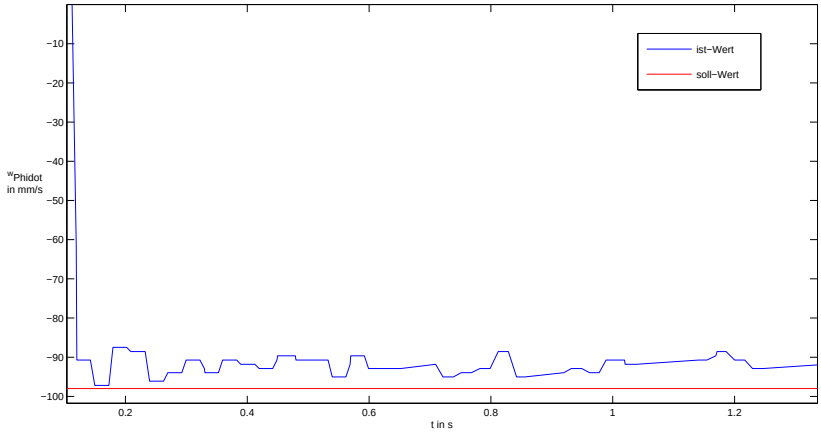
(c) positiver Sprung der y-Geschwindigkeit



(d) negativer Sprung der y-Geschwindigkeit



(e) positiver Sprung der phi-Geschwindigkeit



(f) negativer Sprung der phi-Geschwindigkeit

Abbildung B.1: Sprungantworten des Robotino

C Fehlerbestimmung

Teststrecke 1

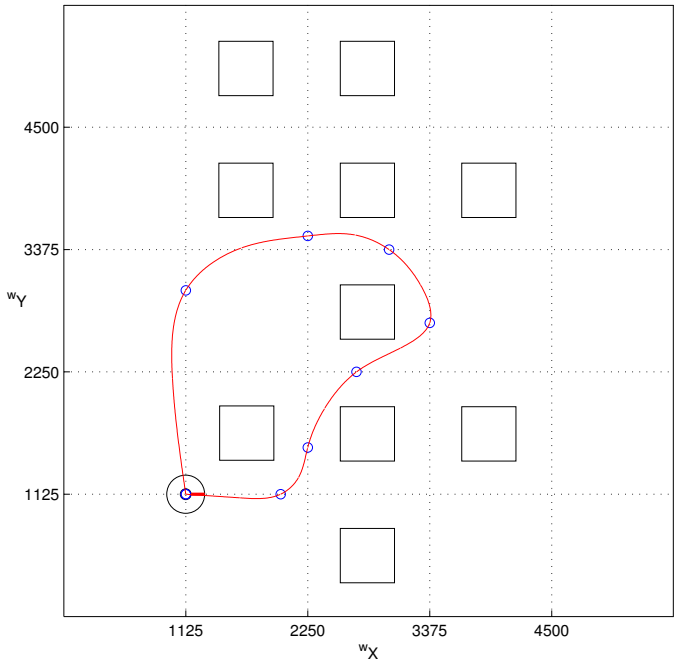


Abbildung C.1: Verlauf von Teststrecke 1

Tabelle C.1: Fehlerbestimmung für Bahn 1

	Lauf 1	Lauf 2	Lauf 3	Lauf 4	Lauf 5
mittlerer quadratischer Fehler	85.46	88.24	90.61	**107.89**	83.35
maximale Bahnabweichung in mm	17.73	18.56	19.11	17.67	**19.05**
Abweichung zum Zielpunkt in mm	0.75	0.53	0.76	0.90	0.40

	Lauf 6	Lauf 7	Lauf 8	Lauf 9	Lauf 10
mittlerer quadratischer Fehler	86.02	83.22	86.23	83.19	106.18
maximale Bahnabweichung in mm	16.58	18.65	18.92	17.56	17.94
Abweichung zum Zielpunkt in mm	0.56	0.68	0.81	0.35	**1.04**

Teststrecke 2

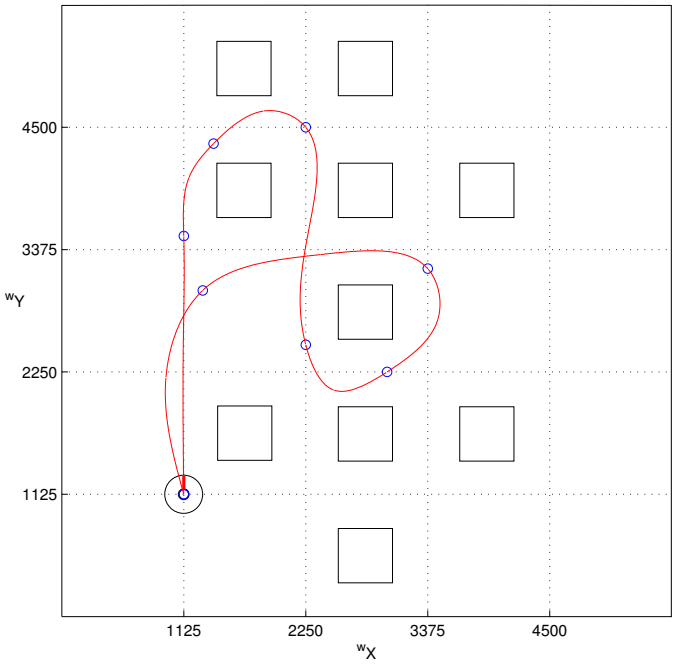


Abbildung C.2: Verlauf von Teststrecke 2

Tabelle C.2: Fehlerbestimmung für Teststrecke 2

	Lauf 1	Lauf 2	Lauf 3	Lauf 4	Lauf 5
mittlerer quadratischer Fehler	75.79	**107.82**	75.84	75.08	91.92
maximale Bahnabweichung in mm	16.31	**17.39**	15.51	15.11	17.11
Abweichung zum Zielpunkt in mm	0.40	**1.37**	0.27	0.64	0.45

	Lauf 6	Lauf 7	Lauf 8	Lauf 9	Lauf 10
mittlerer quadratischer Fehler	74.60	75.37	74.55	75.51	76.03
maximale Bahnabweichung in mm	14.79	15.82	14.73	14.58	15.40
Abweichung zum Zielpunkt in mm	0.53	0.53	0.26	0.77	0.59

Teststrecke 3

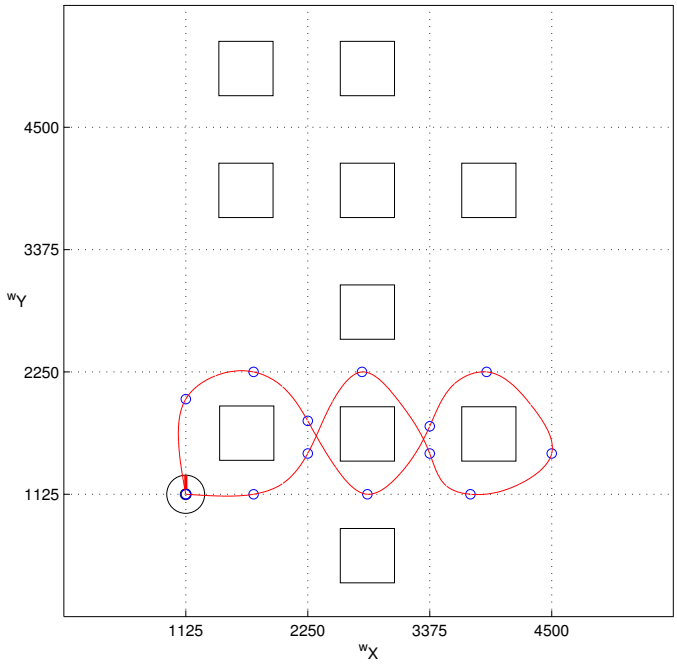


Abbildung C.3: Verlauf von Teststrecke 3

Tabelle C.3: Fehlerbestimmung für Teststrecke 3

	Lauf 1	Lauf 2	Lauf 3	Lauf 4	Lauf 5
mittlerer quadratischer Fehler	116.89	117.05	115.07	115.08	115.92
maximale Bahnabweichung in mm	19.50	19.24	19.28	**20.44**	19.10
Abweichung zum Zielpunkt in mm	0.74	0.93	0.83	0.82	0.50

	Lauf 6	Lauf 7	Lauf 8	Lauf 9	Lauf 10
mittlerer quadratischer Fehler	117.75	**118.07**	115.85	114.67	115.33
maximale Bahnabweichung in mm	20.43	20.07	20.11	19.49	18.80
Abweichung zum Zielpunkt in mm	**1.03**	0.73	0.52	0.43	0.77

D Compact Disc

Inhalt:

- MATLAB Quellcode der Algortihmen zur Bahnplanung und Bahnführung
- C++ Quellcode der Algorithmen zur Bahnplanung und Bahnführung
- Maple Quellcode zur Findung des dynamischen Systems
- PDF dieser Bachelorarbeit
- Rohdaten dieser Bachelorarbeit