Smart Mechatronics

Hochschule Bochum
Bochum University
of Applied Sciences

Bachelorarbeit

zur Erlangung des akademischen Grades
Bachelor of Engineering (B.Eng.)

Implementierung einer Schlupfregelung per Model-Based Design sowie einer SLAM-Kartografierung für ein autonomes Logistik-Fahrzeug

Autoren: Giuliano Montorio
giuliano.montorio@hs-bochum.de
Matrikelnummer: 015202887

Hannes Dittmann
hannes.dittmann@hs-bochum.de
Matrikelnummer: 015206942

Erstgutachter: Prof. Dr.-Ing. Arno Bergmann

Zweitgutachter: Dr.-Ing. Christoph Krimpmann

Abgabedatum: 20.02.2019

Eidesstattliche Erklärung

Eidesstattliche Erklärung zur Abschlussarbeit:

„Implementierung einer Schlupfregelung per Model-Based Design sowie einer SLAM-Kartografierung für ein autonomes Logistik-Fahrzeug“

Wir versichern, die von uns vorgelegte Arbeit selbstständig verfasst zu haben. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer entnommen sind, haben wir als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die wir für die Arbeit benutzt haben, sind angegeben. Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen.

Bochum, 20. Februar 2019	
Ort, Datum	Giuliano Montorio
Bochum, 20. Februar 2019	
Ort, Datum	Hannes Dittmann

Danksagung

Ein großes Dankeschön gilt all jenen Personen, die uns im Rahmen dieser Bachelorarbeit begleitet und geholfen haben. Ganz besonders möchten wir Herrn Prof. Dr.-Ing. Arno Bergmann, Herrn Dr.-Ing. Christoph Krimpmann und Herrn Stefan Beumler, M.Sc. danken, die unsere Arbeit durch ihre fachliche und persönliche Unterstützung begleitet haben. Auch beim Fachbereich Elektrotechnik und Informatik der Hochschule Bochum, insbesondere Herrn Dipl.-Ing. Thorsten Bartsch möchten wir uns bedanken. Für die Bereitstellung von Informationen und Dokumente sind wir Herrn Dennis Hotze, M.Sc. und der Smart Mechatronics GmbH sehr dankbar, ohne deren Hilfe und finanzielle Unterstützung dieses Projekt nicht möglich gewesen wäre.

Inhaltsverzeichnis

Abkürzungsverzeichnis

ALF	Autonomes Logistik Fahrzeug
BLDC	Brushless Direct Current
CAN	Controlled Area Network
CONSENS	Conceptual Design Specification Technique for the Engineering of Complex Systems
EPOS	Entwicklungsplattform Ortsfrequenzfilter-Sensor
FMEA	Failure Mode and Effects Analysis
LIDAR	Light Detection and Ranging
LTI	Linear, zeitinvariantes System
MCM	MotorController Module
RALF	Regelung eines Autonomen Logistik Fahrzeugs
ROS	Robot Operating System
RVIZ	ROS Visualization
SLAM	Simultaneous Localization and Mapping
TEB	Timed Elastic Band
TFEST	Transfer Function Estimation
URDF	Unified Robot Description Format
USBFS	Universal Serial Bus Filesystem

Symbolverzeichnis

Symbol	Bedeutung
D	Dämpfung der Übertragungsfunktion
G	Übertragungsfunktion
$\underline{H}$	Hilfsmatrix
K_P	Verstärkungsfaktor
K_s	Streckenverstärkung
$\mathcal{L}$	Laplace-Transformation
M	Momentanpol
$\mathcal{O}_\mathcal{T}$	Menge aller Odometriedaten
P_a	Skalierungsfaktor für manuellen Betrieb
P_m	Skalierungsfaktor für automatischen Betrieb
R_c	Circumscribed Radius
R_i	Inscribed Radius
T	Abklingzeitkonstante
T_g	Anstiegszeit
T_n	Nachstellzeit
T_u	Verzugszeit
T_v	Vorhaltezeit
U	Laplacetransformierte Eingangsgröße

Symbol	Bedeutung
$\mathcal{X}_\mathcal{T}$	Menge aller Positionsvektoren
$\mathcal{Z}_\mathcal{T}$	Menge aller Umgebungsmessungen
Y	Laplacetransformierte Ausgangsgröße
$\vec{a}$	Umrechnungsvektor
a_i	Koeffizienten der Differentialgleichung der Ausgangsgröße
b_j	Koeffizienten der Differentialgleichung der Eingangsgröße
$\vec{b}$	Allgemeines Bewegungsziel
c	Rotatorischer Bewegungsbefehl
d_i	Reelle Zahl
f	Cost Scaling Factor
g	Impulsantwort
h	Übergangsfunktion
$\vec{h}$	Hilfsvektor
i	imaginäre Einheit $i = \sqrt{-1}$
j	Komplexe Zahl $j = \sqrt{-1}$
k	Komplexe Zahl $k = \sqrt{-1}$
m	Karte der Umgebung
m_i	Landmarken
o	Odometrie
p	Wahrscheinlichkeitsfunktion
$\vec{p}$	Orientierungsvektor
q	Quaternion

Symbol	Bedeutung
r	Distanz
$\vec{r}_a$	Rotatorisches Bewegungsziel aus automatischen Betrieb
$\vec{r}_m$	Rotatorisches Bewegungsziel aus manuellen Betrieb
s	Komplexe Frequenz
t	Zeit
$\vec{t}_{ma}$	Translatorisches Bewegungsziel aus manuellen oder automatischen Betrieb
u	Eingangsgröße
$\hat{u}$	Sprunghöhe
$\vec{v}$	Geschwindigkeitsvektor
w	Führungsgröße
$\vec{x}_t$	Positionsvektor
$\vec{x}_0$	Startpositionsvektor
y	Ausgangsgröße eines Systems
z	Messwert der Umgebung
α	Fahrtwinkel
β	Posenwinkel
δ	Impulsfunktion
σ	Sprungfunktion

1 Einleitung

Die heutige Logistikbranche wird durch autonome und automatische Fahrzeuge im Straßen- und Schienenverkehr revolutioniert [1]. Bereits im Jahr 2017 beschleunigten 80000 Logistikroboter den Transportfluss bei der Firma *Amazon* [2]. Einer Studie von *Strategy*& zufolge können durch die Automatisierung von Logistikprozessen die Kosten um 47% gesenkt werden [3]. Die Hochschule Bochum, insbesondere das Institut für Systemtechnik forscht an einem autonomen Logistik-Fahrzeug (ALF), das derartige Prozesse bewältigen kann.

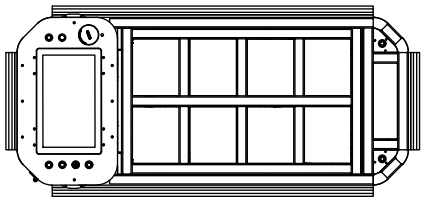
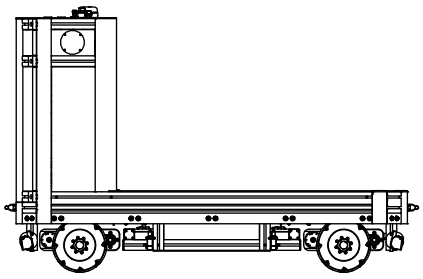
(a)  (b) 

Abbildung 1.1: (a) Darstellung des autonomen Logistik-Fahrzeugs aus der Draufsicht. Die acht benachbarten Rechtecke in der Mitte des Fahrzeugs stellen die Ladefläche des Fahrzeugs dar. (b) Darstellung aus der Seitenansicht. Oben links in der Abbildung ist der Schaltschrank zu sehen. Die Räder des Fahrzeugs befinden sich unten links und unten rechts in der Darstellung.

Das ALF aus Abbildung 1.1 und A.1 stellt die Grundlage für eine Entwicklung autonomer Fahrfunktionen. Dieses wurde im Rahmen der Masterarbeit „Entwicklung und Verifikation eines autonomen Logistik-Fahrzeugs“ von Dennis Hotze und Dominik Eickmann an dem Institut für Systemtechnik entwickelt. Das Fahrzeug dient als Plattform für aufeinander aufbauende Projekte, so auch für diese Bachelorarbeit. Die Grundidee besteht darin, Logistikprozesse an der Hochschule Bochum durch das ALF zu unterstützen. Die Motivation dieser Bachelorarbeit ist die Erforschung des autonomen Fahrens an der Hochschule Bochum.

Das ALF wurde so konzipiert, dass für die Bewegung des Fahrzeugs keine Veränderung an der Umgebung vorgenommen werden muss. Für die Erkennung der Umgebung ist das ALF mit entsprechender Sensorik ausgestattet, die in Abschnitt 2.5 erläutert ist. Ein berührungsempfindlicher Bildschirm auf der Oberseite des Fahrzeugs ermöglicht eine Interaktion mit einem verbauten Rechner, auf dem das Betriebssystem *Linux* installiert ist. Durch das vorangegangene Projekt ist die Steuerung des Fahrzeugs durch einen Joystick möglich. In Abbildung 1.1 sind im Zentrum der Darstellung acht Rechtecke zu sehen. Diese stellen die Ladefläche dar, die mit einem Gewicht von bis zu 370 kg beaufschlagt werden kann. Es sind vier bürstenlose Gleichstrommotoren (BLDC) in das Fahrzeug integriert, die jeweils ein Mecanumrad antreiben. Deren Funktionsweise wird in Abschnitt 2.3 erläutert. Mithilfe der sogenannten *MotorController Module* (MCM) werden jeweils zwei der vier BLDC-Motoren angesteuert. Die Module wurden an der Hochschule Bochum entwickelt und dienten für den Einsatz in einem elektronisch betriebenen Skateboard [4].

Im Kontext dieser Bachelorarbeit wird das Fahrverhalten des Fahrzeugs und seine Fähigkeit, autonom zu fahren, weiterentwickelt. Die daraus resultierenden Anforderungen an das Fahrzeug werden in dem Anhang A.1.3 in Form eines Lastenhefts festgehalten. Für die Realisierung der beschriebenen Ziele durch *Model-Based Design* (MBD), wird in dieser Bachelorarbeit das Programm *Matlab/Simulink* verwendet. Der MBD-Ansatz spiegelt sich über die Verwendung von *Simulink* hinaus in der Konzeptionierung, die in Kapitel 3 beschrieben ist, wieder. Es traten unerwünschte Rotationen während der Ausführung von Fahrmanövern des Fahrzeugs auf und ruften unvorhersehbare Bewegungen hervor. Mit Hilfe einer Lageregelung und einer untergeordneten Drehzahlregelung werden diese unterbunden. Dies ermöglicht eine Integration von autonomen Fahrfunktionen. Im Rahmen dieses Projekts ist die eigenständige Bewegung des ALFs bei gleichzeitiger Kartografierung der Umgebung zu entwickeln.

2 Grundlagen

Im Folgenden werden die Grundlagen behandelt, die für die Umsetzung des Projekts nötig sind und für ein besseres Verständnis der in Kapitel 4 erläuterten Funktionen dienen. Die Anbindung der verwendeten Sensorik im *Robot Operating System* (ROS) ist notwendig, um die im Anhang A.1.3 beschriebenen Anforderungen zu erfüllen.

2.1 Grundprinzip des Robot Operating System

Im Kontext dieser Bachelorarbeit wird ROS für die Kommunikation zwischen der verbauten Sensorik und der verwendeten Software, sowie der Integration autonomer Fahrfunktionen genutzt. Das Robot Operating System ist ein Softwareframework, das Entwicklern die Integration von komplexen Roboteranwendungen erleichtert und ist prinzipiell in Abbildung 2.1 dargestellt. [5]

Der Datenaustausch in dem ROS-Netzwerk findet zwischen sogenannten Knoten statt. Diese repräsentieren bestimmte Softwareanwendungen, wie zum Beispiel das Erfassen und Umrechnen von Sensorrohdaten. Die Knoten veröffentlichen und abonnieren sogenannte Topics, wodurch ein Datenaustausch stattfindet. Die Datenübertragung wird durch in ROS vordefinierte oder selbst erstellte Nachrichtentypen realisiert. [6]

Im vorangegangenen Projekt wurde ROS bereits in das ALF eingebunden und ist durch das Lastenheft für den Einsatz in dieser Bachelorarbeit vorgesehen [4].

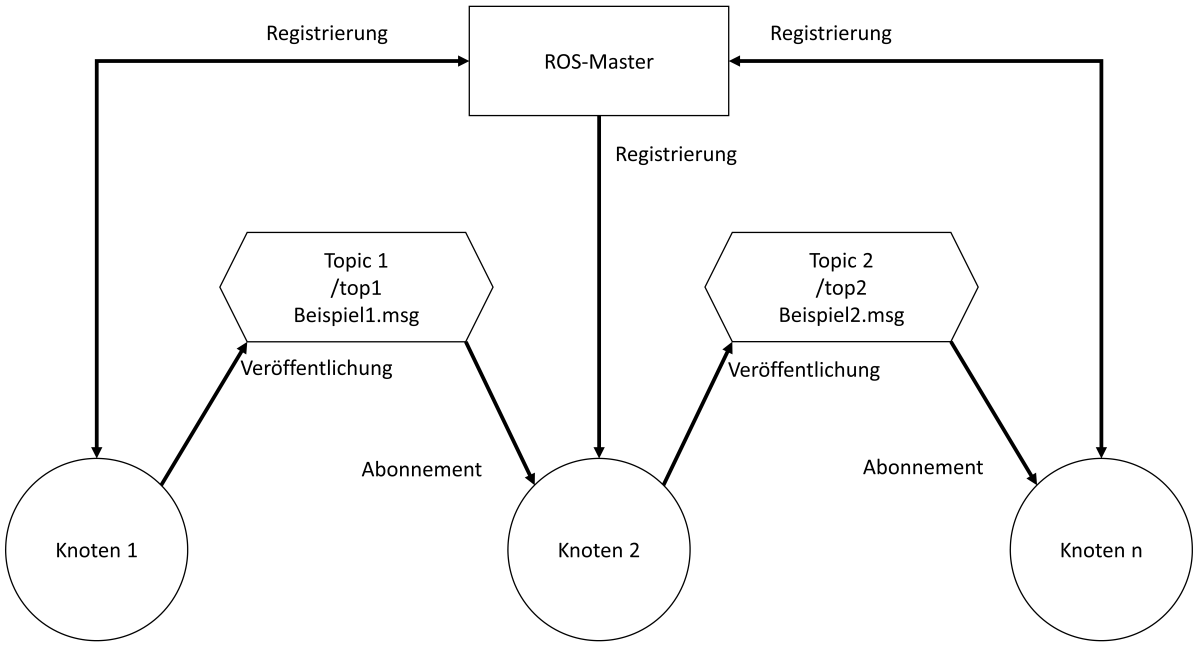


Abbildung 2.1: Funktionsprinzip des Robot Operating Systems. Die Pfeilrichtungen symbolisieren in dieser Darstellung den Informations- und Datenfluss. Knoten sind in dieser Abbildung als Kreis, Topics als Hexagon und der ROS-Master als Rechteck ausgeführt.

Des Weiteren können Einstellungen für Knoten auf dem Parameterserver hinterlegt werden, zum Beispiel die Abtastraten der in Kapitel 4.2.3 beschriebenen *Kinect*-Sensoren. Diese Informationen sind global in dem hier veranschaulichten Netzwerk sichtbar, sodass jeder Knoten auf alle Parameter zugreifen kann [7]. Der Parameterserver befindet sich auf der zentralen Einheit eines ROS-Netzwerks, den sogenannten Master [7]. In den meisten Fällen sind dies Rechner oder Steuereinheiten, die für die Namensgebung und Kommunikation der Knoten zuständig sind [8, 6].

2.2 Eigenschaften von Regelsystemen

Ziel der in Kapitel 1 beschriebenen Regeleinrichtung ist es, die im Anhang A.1.3 erläuterten Anforderung zu erfüllen. Aus diesen geht hervor, eine Lage- und Drehzahlregelung zu integrieren, um den in Kapitel 1 beschriebenen Rotationen entgegenzuwirken.

Grundlage für den Reglerentwurf ist die Kenntnis des Übertragungsverhaltens der zu regelnden Antriebe des ALFs. Lässt sich das Verhalten dieser mithilfe von physikalischen Gesetzmäßigkeiten oder Messungen erfassen, so entspricht die aufgestellte Gleichung einem mathematischen Modell. Dies dient als Ausgangspunkt für Analysen, Synthesen oder Simulationen von Regelsystemen. Zwei Möglichkeiten, das Übertragungsverhalten eines Systems zu ermitteln, sind die Sprungantwort und die Impulsantwort. Die Sprungantwort $y(t)$ ist definiert als die Reaktion des Systems auf eine sprungförmige Eingangsgröße $u(t)$ aus Gleichung (2.1). [9]

$$u(t) = \hat{u}\,\sigma(t) \text{ mit } \hat{u} = \text{ const. und } \sigma(t) = \begin{cases} 1 & \text{für } t > 0 \\ 0 & \text{für } t < 0 \end{cases} \tag{2.1}$$

Die Übergangsfunktion stellt die auf die Sprunghöhe $\hat{u}$ bezogene Sprungantwort $h(t) = \frac{y(t)}{\hat{u}}$ dar, die bei einem kausalen System die Eigenschaft $h(t) = 0$ für $t < 0$ besitzt. Die Impulsantwort $g(t)$ ist definiert als die Antwort des Systems auf die Impulsfunktion. Diese wird als Einheitsimpuls oder Dirac-„Stoß“ bezeichnet und wird durch die Gleichung (2.2) beschrieben. [9]

$$\delta(t) = \begin{cases} 0 & \text{für } t < 0 \text{ und } t > 0 \\ \infty & \text{für } t = 0 \end{cases} \tag{2.2}$$

Die Impulsfunktion $\delta(t)$ wird symbolisch als Pfeil mit der Länge 1 dargestellt und besitzt die Eigenschaft $\int_{-\infty}^{\infty} \delta(t)\mathrm{d}t = 1$. Es ist zu beachten, dass für die Höhe des Impulses weiterhin $\delta(0) = \infty$ gilt. Abbildung 2.2 zeigt die Sprung- und Impulsfunktion. [9]

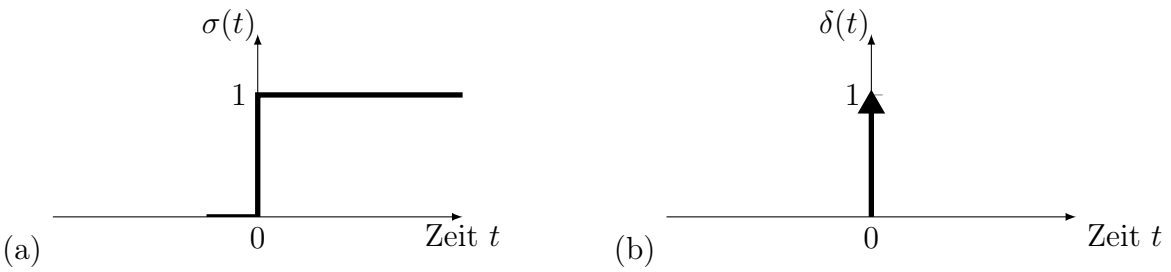


Abbildung 2.2: (a) Darstellung der Sprungfunktion $\sigma(t)$. (b) Die Impulsfunktion $\delta(t)$ wird symbolisch durch einen Pfeil der Länge 1 abgebildet [9]. Die Länge 1 wird auch als Impulsstärke bezeichnet, wobei für die Höhe des Impulses weiterhin $\delta(0) \to \infty$ aus Gleichung (2.2) gilt [9]. Zwischen den Testfunktionen besteht der Zusammenhang $\delta(t) = \frac{\mathrm{d}}{\mathrm{d}t}\,\sigma(t)$ [9].

Werden die beiden Funktionen $u(t)$ und $\delta(t)$ als Eingangssignal eines linearen Systems verwendet, so erhält man als Ausgangsgröße die Übergangsfunktion $h(t)$ und die Impulsantwort $g(t)$. Ein System heißt dann linear, wenn sich die Wirkung zweier linear überlagerter Eingangssignale in gleicher Weise am Ausgang überlagern [10]. Abbildung 2.3 zeigt beispielhaft die Übergangsfunktion und Impulsantwort für ein Verzögerungsglied erster Ordnung.

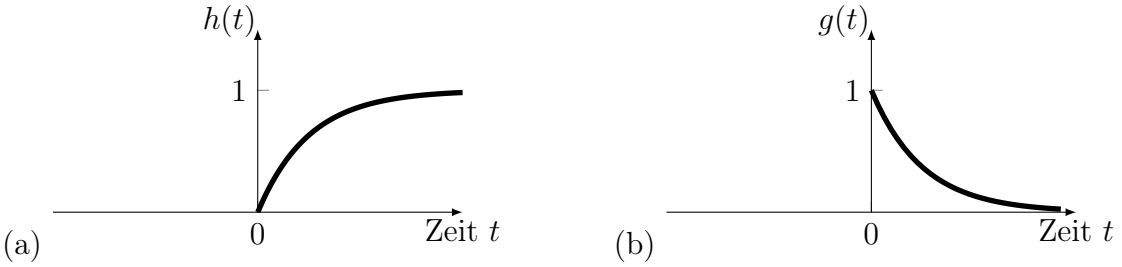


Abbildung 2.3: (a) Die Abbildung zeigt die Übergangsfunktion $h(t)$ eines Verzögerungsglieds erster Ordnung als Antwort auf die sprungförmige Eingangsgröße $u(t)$ mit $\hat{u} = 1$. (b) Die Impulsantwort auf das Eingangssignal $\delta(t)$. Zwischen den Funktionen $h(t)$ und $g(t)$ besteht der Zusammenhang $g(t) = \frac{\mathrm{d}}{\mathrm{d}t}\,h(t)$.

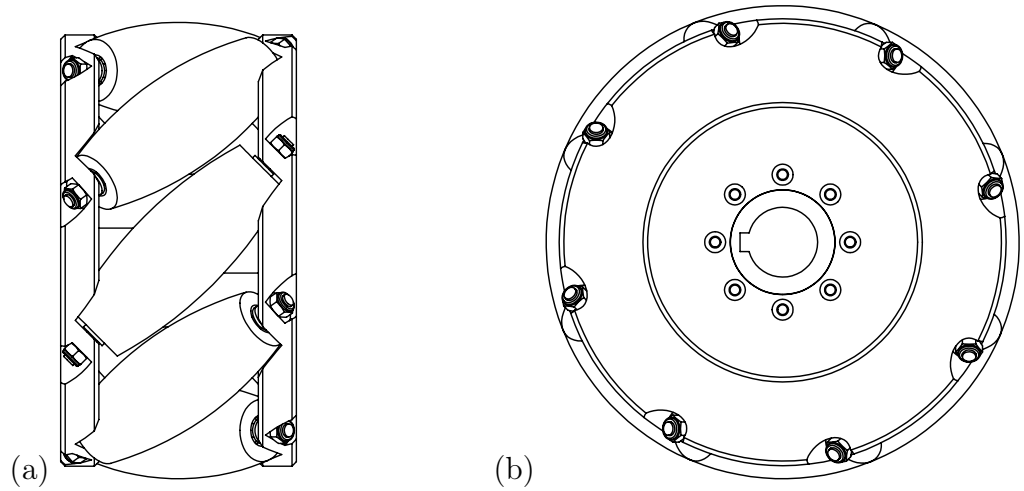
Mit Hilfe der Übertragungsfunktion kann bei bekannter Eingangsgröße $u(t)$ beziehungsweise $U(s)$ unmittelbar die Laplace-Transformierte der Ausgangsgröße $Y(s)$ ermittelt werden. Die Übertragungsfunktion $G(s)$ ist definiert als die Laplace Transformierte der Impulsantwort $g(t)$ und wird in Gleichung (2.3) abgebildet. [9]

$$\text{Übertragungsfunktion: } \mathcal{L}\{g(t)\} = G(s) = \frac{Y(s)}{U(s)} \tag{2.3}$$

Die Gleichung (2.3) gilt für lineare, kontinuierliche, zeitinvariante Systeme (LTI) ohne Berücksichtigung einer Totzeit. Deren Verhalten wird durch die gewöhnliche lineare Differentialgleichung $n-ter$ Ordnung der Form aus Gleichung (2.4) beschrieben [9]. Systeme mit Totzeit können nicht durch eine gewöhnliche Differentialgleichung, sondern nur durch eine partielle Differentialgleichung beschrieben werden [11].

$$\sum_{i=0}^{n} a_i \frac{\mathrm{d}^i y(t)}{\mathrm{d}t^i} = \sum_{j=0}^{n} b_j \frac{\mathrm{d}^j u(t)}{\mathrm{d}t^j} \tag{2.4}$$

2.3 Funktionsprinzip der Mecanumräder



(a) (b)

Abbildung 2.4: (a) Abbildung eines der in den ALF verbauten Mecanumräder aus der Draufsicht. Die in einem Winkel von 45° angeordneten, ovalen Körper stellen Rollen dar, die am Rad angebracht sind. Diese erlauben eine omnidirektionale Fahrweise, die in Kapitel 2.4.1 erläutert ist. Links und rechts in der Darstellung sind mit den Rollen verbundene Befestigungsplatten zu sehen. (b) Darstellung eines der in den ALF montierten Mecanumräder aus der Seitenansicht. Man erkennt an der kreisförmigen Abbildung des Rades, wie eine runde Form durch die ovalen Rollen zustande kommt. [12]

Für die Übertragung des von den Motoren bereitgestellten Drehmoments und für die Realisierung einer Lenkung sind am ALF durch die vorangegangene Masterarbeit vier Mecanumräder verbaut [4]. Beispielhaft ist in Abbildung 2.4 ein solches Mecanumrad dargestellt. Entlang des Umfangs der Mecanumräder sind freilaufende Rollen angebracht, dadurch ist eine Bewegung in beliebige Richtungen möglich [13]. Derartige Fahrmanöver werden als omnidirektional bezeichnet und in Kapitel 2.4.1 erklärt [14]. Damit das Rad eine runde Form beibehält ist das Profil der in Abbildung 2.4 dargestellten Rollen oval konstruiert [4]. Durch die Rollen der Mecanumräder und einer unebenen Fahrbahn wird ein schlupfbehaftetes Abrollen hervorgerufen. Es kommt zu unerwünschten Rotationen des Fahrzeugs, wodurch die in Kapitel 1 geschilderten Folgen hervorgerufen werden [14, 4].

2.4 Unterscheidung von Lenkungsprinzipien des autonomen Logistik Fahrzeugs

Das ALF ist für den Einsatz auf dem Gelände der Hochschule Bochum konzipiert. In diesem Arbeitsumfeld kommt es zu engen Passagen und Kurven. Diese können mit einer Omnidirektionalen Bewegung ohne Änderung der Fahrzeugorientierung nicht durchfahren werden. Aus diesem Grund ist es in dieser Bachelorarbeit vorgesehen die Lenkung des Roboters, nicht nur nach der omnidirektionalen Verfahrweise, sondern auch nach dem Prinzip der Achsschenkellenkung auszulegen. In den folgenden Abschnitten 2.4.1 und 2.4.2 werden die beiden Lenkungstechniken erläutert.

2.4.1 Omnidirektionaler Antrieb

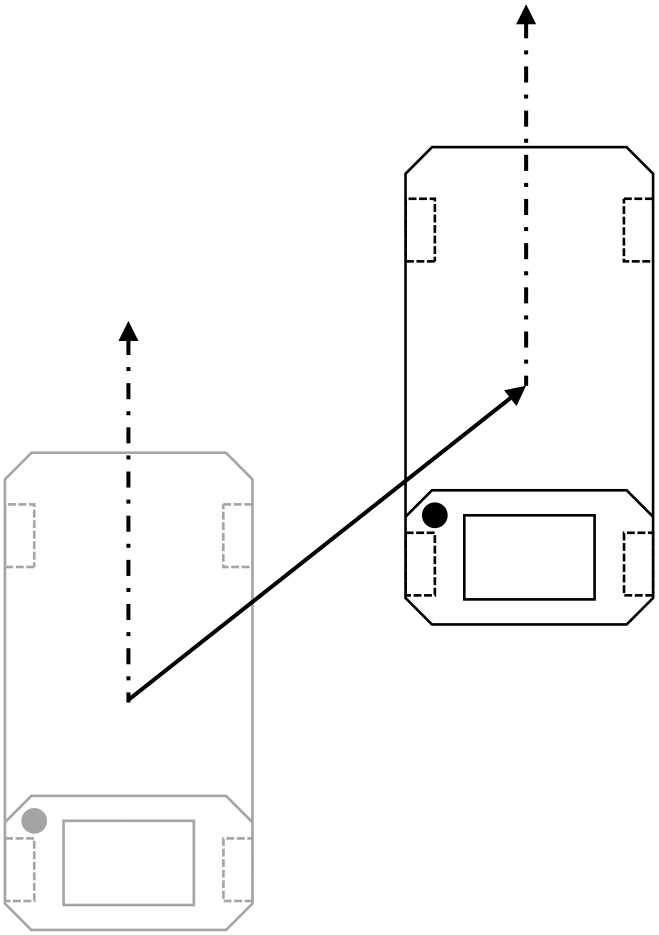


Abbildung 2.5: Beispiel einer Bewegung des Fahrzeugs ohne Änderung seiner Orientierung aus der Draufsicht. Der ALF wird oben rechts und unten links in zwei Positionen als Oktagon dargestellt. Der als Volllinie ausgeführte Pfeil zeigt die Bewegungsrichtung und der als Strichpunktlinie ausgeführte Pfeil die Orientierung des Fahrzeugs.

Ein omnidirektionaler Antrieb ermöglicht Fahrzeugen, bei gleichbleibender Fahrzeugorientierung, in beliebige Richtungen zu manövrieren [15]. Dies ist in Abbildung 2.5 dargestellt. Für diese Bachelorarbeit wird eine Fahrt in Richtung des als Strichpunktlinie dargestellten Pfeils als vorwärts angenommen. Wenn es aufgrund der räumlichen Gegebenheiten und Kurvenfahrten zu einer Änderung der Fahrzeugorientierung kommen muss, kann dies durch die in Abbildung 2.6 gezeigte Ansteuerung der Mecanumräder realisiert werden.

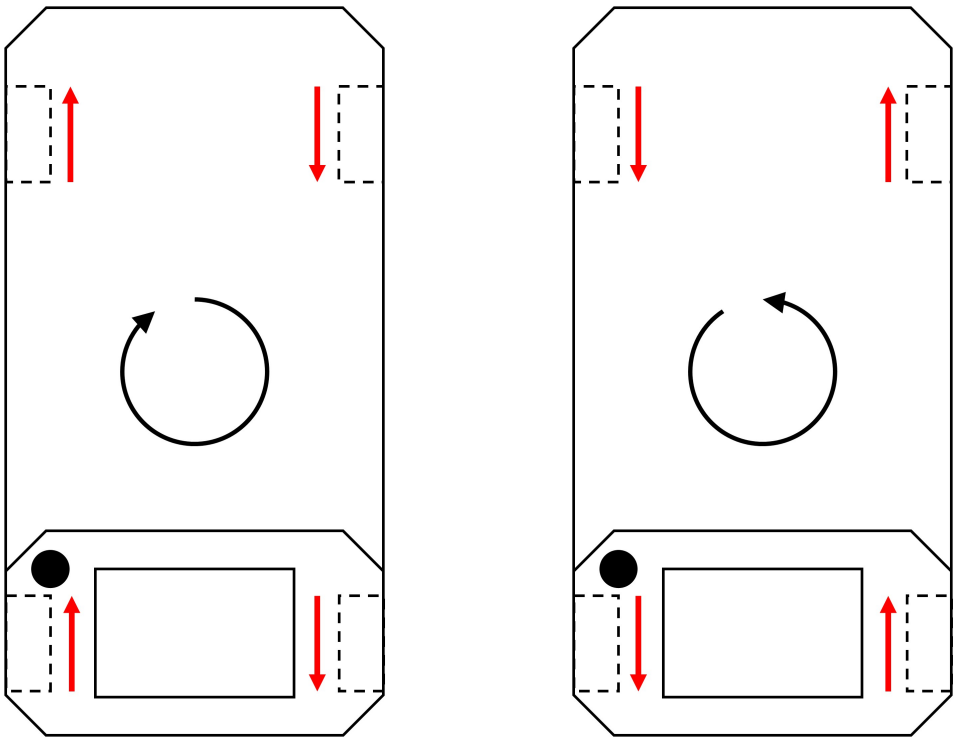


Abbildung 2.6: Darstellung einer Rotation im und gegen den Uhrzeigersinn des Fahrzeugs ohne Änderung der Position von oben betrachtet. Gestrichelte Rechtecke stellen in hier die Räder des ALFs dar. Die roten, geraden Pfeile zeigen symbolisch den Betrag der Winkelgeschwindigkeit an den jeweiligen Rädern. Runde Pfeile zeigen den Drehsinn für die entsprechende Ansteuerung der Räder.

In Abbildung 2.6 sind neben der als gestrichelten Viereck dargestellten Mecanumräder rote Pfeile zu sehen. Diese stellen symbolisch den Betrag und die Drehrichtung der entsprechenden Räder dar. Als runder Pfeil ist die Drehrichtung des ALFs für eine Drehung mit und gegen den Uhrzeigersinn abgebildet. Durch die in Abbildung 2.6 dargestellte Ansteuerung befindet sich die Drehachse in dem Zentrum des Fahrzeugs, sodass eine ausschließlich rotatorische Bewegung zustande kommt [15].

2.4.2 Achsschenkellenkung

Das Prinzip der Achsschenkellenkung setzt voraus, dass alle vier Räder wie in Abbildung 2.7 tangential auf Kreisbahnen verlaufen [16]. Diese besitzen denselben Mittelpunkt, den Momentanpol M des Fahrzeugs [16]. Daraus resultiert eine Kurvenfahrt mit konstantem Radius, bei der das Fahrzeug immer gleich zu dem Rotationszentrum ausgerichtet ist. Eine Geradeausfahrt wird idealisiert als Kurvenfahrt mit unendlich großem Radius betrachtet [17].

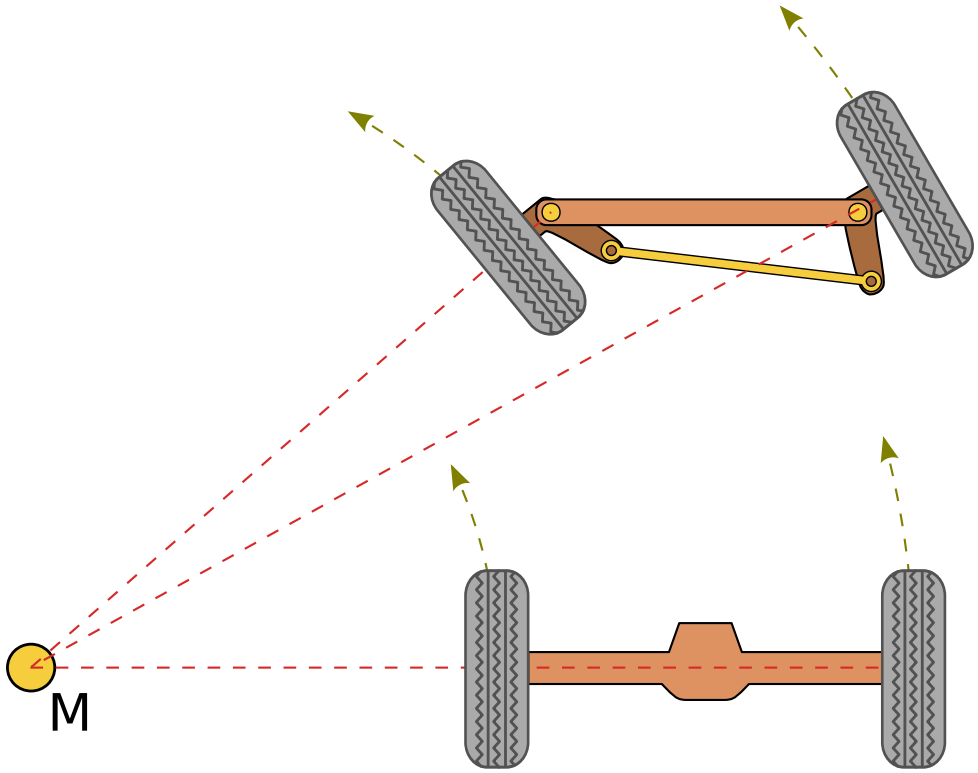


Abbildung 2.7: Darstellung einer Achsschenkellenkung aus der Draufsicht. Unten rechts in der Abbildung ist eine starre, nicht lenkbare Achse zu sehen. Die Räder der oben rechts dargestellten Lenkachse und der starren Achse rotieren auf Kreisbahnen um den Momentanpol M. [18]

Bei dem ALF ist dies nicht möglich, da die Räder nicht über eigene Schwenkachsen verfügen und in dieselbe Richtung zeigen. Daher wurde als weiteres Lenkungsprinzip bei dem ALF ein zur Achsschenkellenkung analoges Lenkverhalten entwickelt.

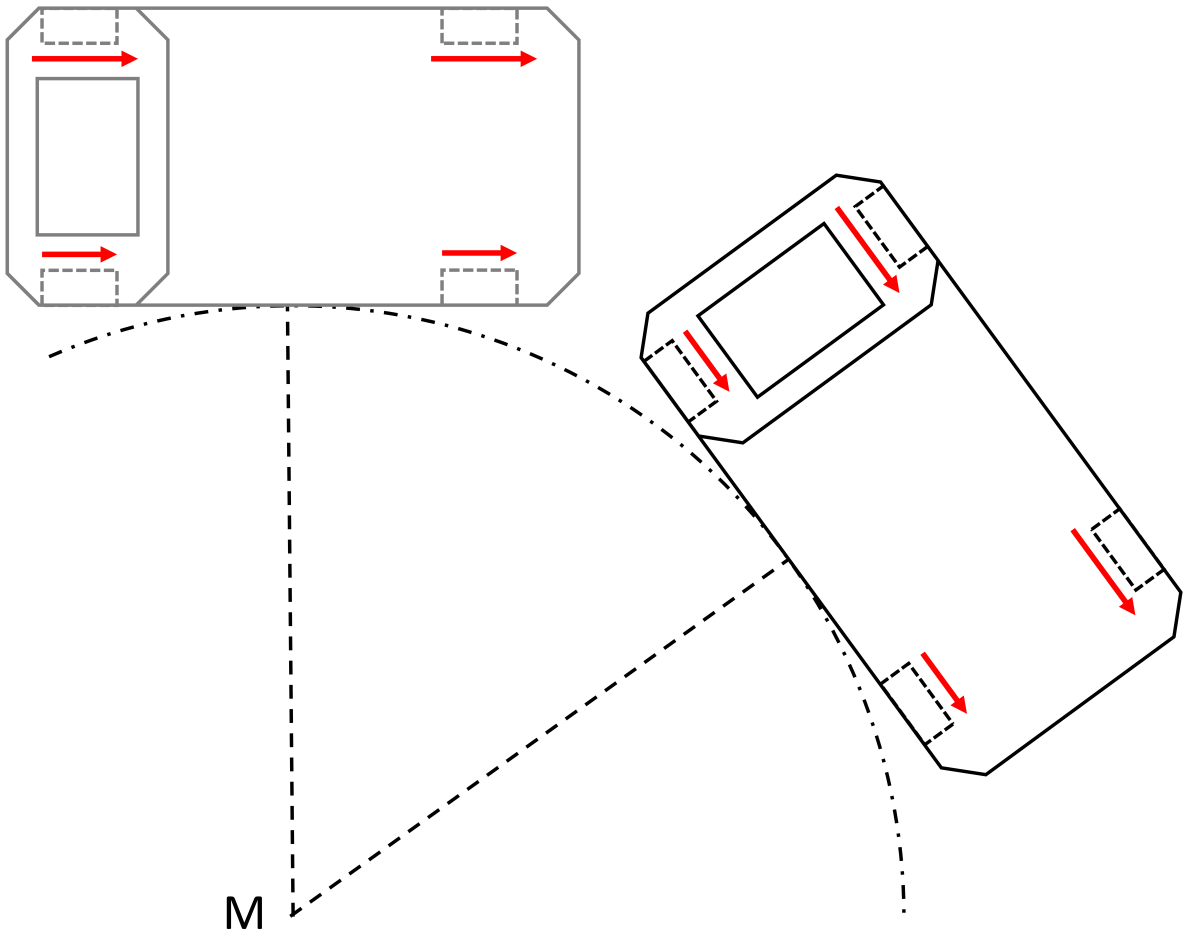


Abbildung 2.8: Realisierung einer Kurvenfahrt durch eine zur Achsschenkellenkung analogen Fahrweise. Zu sehen ist das ALF aus der Draufsicht in zwei Positionen. Durch die grau dargestellte Draufsicht ist das Fahrzeug in Position 1 dargestellt. Dabei ist das ALF gleich zum Momentanpol M ausgerichtet, wie in der schwarzen Darstellung. Diese kennzeichnet Position 2. Der mit einer Strichpunktlinie dargestellte Kreis suggeriert die Kreisbahnen aus Abbildung 2.7. Die Ansteuerung der Räder wird durch die Pfeile symbolisiert, die die Winkelgeschwindigkeiten dieser zeigen. Hierbei weisen die vom Rotationszentrum weiter entfernten Räder eine größere Winkelgeschwindigkeit auf.

Durch die in Abbildung 2.8 dargestellte Ansteuerung der Mecanumräder, kann eine zur Achsschenkellenkung analoge Fahrweise hervorgerufen werden.

2.5 Sensorik zur Erkennung der Umgebung

Abbildung 2.9: Abbildung des in dieser Bachelorarbeit verwendeten *RPLIDAR A2*. Bei Inbetriebnahme rotiert die obere Hälfte des Sensors, die durch die untere rote Linie im Zentrum der Abbildung markiert ist. [19]

Für die Erkennung von Objekten werden drei Sensoren verwendet. Neben zwei Kinect-Sensoren ist ein rotierender 360°-Laserscanner namens *RPLIDAR A2* der Firma *Slamtec* verbaut. Der Name Lidar kommt aus dem Englischen und ist eine Abkürzung für „light detection and ranging“. Durch die Rotation des in Abbildung 2.9 dargestellten Sensors, werden Gegenständen auf dessen Montagehöhe erfasst. [19]

Abbildung 2.10: Abbildung des in dieser Bachelorarbeit verwendeten *Kinect*-Sensors. [20]

Der beschriebene Lidar-Sensor ist auf einer Höhe von circa einem Meter montiert, somit werden Objekte unter dieser Höhe nicht erkannt. Die Auswertung der *Kinect*-Sensoren ist durch die Erweiterung des Sichtfeldes begründet. Diese befinden sich an der Schaltschranktür und am Rücken des Schaltschranks. Neben einer üblichen Kamerafunktion sind Tiefensensoren in die Geräte integriert. Im Rahmen dieser Bachelorarbeit werden die Kinect Kameras aus Abbildung 2.10 als weitere Quelle für Umgebungsinformationen genutzt. Dadurch können Gegenstände unter der Montagehöhe des Lidar-Sensors erkannt werden.

2.6 Funktionsweise von Simultaneous Localization and Mapping

Simultaneous Localization and Mapping (SLAM) beschreibt das Problem in der Robotik, bei dem ein mobiler Roboter eine Karte seiner Umgebung erstellt und gleichzeitig seine Position in dieser schätzt [21]. Zusätzlich zur Schwierigkeit der Positionsschätzung, besteht für mobile Roboter ein Datenassoziationsproblem. Dadurch kommt es zu Überlappungen von neuen und bereits in einer Karte vorhandenen Messwerten [21]. Eine für das Projekt relevante Lösung von SLAM wird im Abschnitt 2.6.2 näher erläutert.

2.6.1 Mathematische Beschreibung des SLAM Problems

Die Zeit wird mit t und die Roboterposition zum Zeitpunkt t mit $\vec{x}_t$ definiert. Für mobile Roboter in der Ebene ist $\vec{x_t}$ ein dreidimensionaler Vektor, wobei zwei Koordinaten die Lage in der Ebene und ein Wert die Orientierung des Roboters beschreibt. Die Abfolge der Bewegung oder eine Trajektorie wird mit der bekannten Ausgangsposition $\vec{x}_0$ und der Gleichung (2.5) beschrieben. [21]

$$\mathcal{X}_T = \{\vec{x}_0, \vec{x}_2, \vec{x}_2, ..., \vec{x}_T\} \text{ mit } T \to \infty \tag{2.5}$$

Informationen zwischen zwei aufeinanderfolgenden Positionen des Roboters $\vec{x}_{t-1}$ und $\vec{x}_t$ werden als Odometrie o bezeichnet und mit der Gleichung (2.6) dargestellt. Odometriedaten können zum Beispiel aus Inkremental-, Absolutwertgebern oder Beschleunigungssensoren gewonnen werden. [21, 22]

$$\mathcal{O}_T = \{o_0, o_1, o_2, ..., o_T\} \text{ mit } T \to \infty \tag{2.6}$$

Für eine Erkennung der Umgebung, muss das ALF mit entsprechender Sensorik ausgestattet sein. Derartige Sensoren wurden in Abschnitt 2.5 behandelt. Die reale Karte der Umgebung wird durch m beschrieben und beinhaltet die Positionen von auffälligen Punkten der Umgebung, sogenannten Landmarken. Im gegebenen Anwendungsfall sind dies statische Objekte. Die Sensorik zur Erkennung der Umgebung gibt Informationen zur aktuellen Position $\vec{x}_t$ und der Umgebung m. Wird angenommen, dass diese zu jedem Zeitpunkt einen Messwert z aufnimmt, so wird die Reihe von Messwerten durch Gleichung (2.7) definiert. [21, 23]

$$\mathcal{Z}_T = \{z_1, z_2, z_3, ...,z_T\} \text{ mit } T \to \infty \tag{2.7}$$

Alle Roboterpositionen, Landmarken und Messwerte stehen als Zufallsvariablen in Abhängigkeit zueinander. Die Lösung der SLAM-Problematik zielt darauf ab, die Wahrscheinlichkeitsverteilung von Landmarken und Roboterpositionen abhängig von den Messwerten zu schätzen. Durch diese Schätzung wird die wahrscheinlichste Robotertrajektorie und Umgebungskarte bestimmt. Bei der SLAM Problematik wird zwischen zwei Fällen unterschieden: dem Full-SLAM und Online-SLAM Problem. Bei dem Full-SLAM wird die gesamte Trajektorie und die Umgebung auf Grundlage der Messwerte $\mathcal{Z}_T$ und Odometrie Daten $\mathcal{O}_T$ geschätzt. In Abbildung 2.11 ist dies graphisch dargestellt. [21, 23]

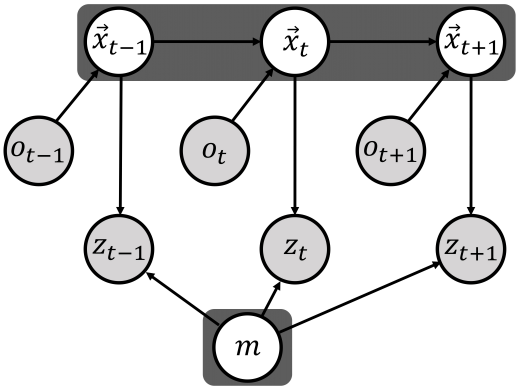


Abbildung 2.11: Graphisches Modell des Full-SLAM Problems. Die Pfeile zeigen direkte Abhängigkeiten der eingeführten Variablen. Die grau hinterlegten Variablen sind nicht messbare Größen, die mit den messbaren Mengen $\mathcal{O}_T$ und $\mathcal{Z}_T$ geschätzt werden. [24]

Algorithmen zur Lösung des Full-SLAM Problems verarbeiten alle aufgenommenen Daten nach der Fahrt, bei der die Umgebung aufgenommen wurde. Mathematisch wird das Full-SLAM Problem in Gleichung (2.8) definiert.

$$p(\mathcal{X}_T,\, m \mid \mathcal{Z}_T,\, \mathcal{O}_T) \tag{2.8}$$

Die Gleichung (2.8) drückt die Wahrscheinlichkeit für die gesamte Trajektorie $\mathcal{X}_T$ und Karte m für die gegebenen Daten $\mathcal{Z}_T$ und $\mathcal{O}_T$ aus. Online-SLAM hingegen beschreibt nur die Schätzung der aktuelle Position x_t und Karte m auf Basis der Messwerte $\mathcal{Z}_T$ und Odometriedaten $\mathcal{O}_T$ während der Fahrt. In Gleichung (2.9) wird dies mathematisch ausgedrückt. Algorithmen, die das Online-SLAM Problem lösen, verarbeiten die Datenelemente während der Fahrt. Dabei wird die Wahrscheinlichkeit für die aktuelle Roboterposition und Karte unter der Voraussetzung der gegebenen Messwerte und Odometriedaten geschätzt. In Abbildung 2.12 ist das graphische Modell des Online-SLAM Problems zu sehen. [21, 23]

$$p(\vec{x}_t,\, m \mid \mathcal{Z}_T,\, \mathcal{O}_T) \tag{2.9}$$

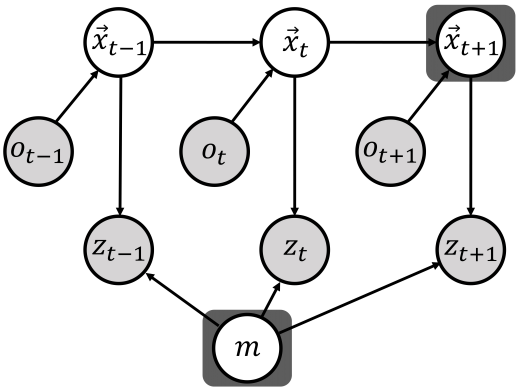


Abbildung 2.12: Graphisches Modell des Online-SLAM Problems. [24]

2.6.2 Graph-basierende Techniken

Bei graphbasierten Techniken werden die Wahrscheinlichkeitsverteilungen des SLAM-Problems als sogenannte Graphen dargestellt. Dieser wird in Abbildung 2.13 beispielhaft präsentiert. In den Graphen werden Knoten eingetragen, die den Landmarken m_i mit $i \in \mathbb{R}$ der Umgebung und Positionen des Roboters $\vec{x}_t$ entsprechen. Die Knoten werden durch Kanten verbunden, die einen stochastischen Zusammenhang zwischen den Variablen abbilden. Beobachtet der Roboter eine Landmarke, muss demzufolge eine Kante zwischen der Position des Roboters und der beobachteten Landmarke eingetragen werden. Des Weiteren besteht eine Abhängigkeit zwischen den einzelnen Positionen des Roboters. Somit muss auch eine Kante zwischen den Positionen $\vec{x}_t$ und $\vec{x}_{t-1}$ eingetragen werden. Diese entspricht der in Abschnitt 2.6.1 beschriebenen Odometrie. Werden Wahrscheinlichkeitswerte über die Anwesenheit von beobachteten Landmarken den Zellen einer Matrix zugewiesen, so entsteht in Kombination mit den Messungen Z_T die sogenannte *Occupancy Grid*-Karte oder auch Belegtheitskarte. [21, 23, 25]

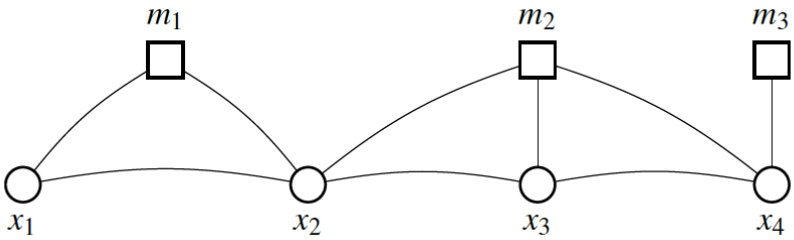


Abbildung 2.13: Beispiel einer graphischen Darstellung mit Landmarken, die als eckige Knoten abgebildet sind. Positionen sind hier als runde Knoten und Kanten als Verbindungen zwischen Knoten eingetragen. [23]

In dieser Bachelorarbeit werden nicht einzelne Beobachtungen von Landmarken verarbeitet sondern ganze Aufnahmen der Umgebung, zum Beispiel aus dem in Kapitel 2.5 beschriebenen Lidar Sensor. Informationen über die relativen Positionen von Aufnahmen zueinander werden in einer Graphstruktur wie aus Abbildung 2.13 gespeichert und anschließend ausgerichtet, wodurch das Datenassoziationsproblem gelöst wird. Eine solche Vorgehensweise wird als *Scan Matching* bezeichnet. [21, 23]

2.7 Darstellungsarten der Orientierungen

In dieser Bachelorarbeit sind zwei Darstellungsformen von Rotationen relevant, die Darstellung durch Roll-Nick-Gier-Winkel, sogenannte Euler-Winkel, und die durch Quaternionen. Für die Koordinatentransformationen und die Positionsschätzung des verwendeten SLAM-Algorithmus in Abschnitt 4.3 werden diese verwendet.

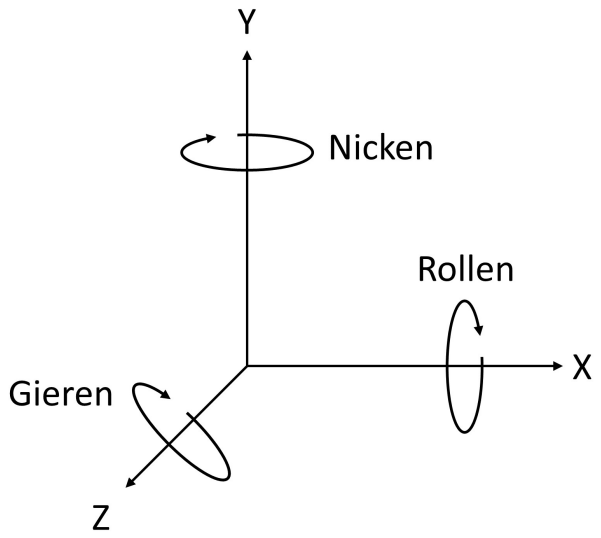


Abbildung 2.14: Darstellung von Rollen, Nicken und Gieren.

In Abbildung 2.14 werden die Begriffe Rollen, Nicken und Gieren veranschaulicht. Das Rollen typisiert eine Rotation um die X-Achse, das Nicken eine um die Hoch- beziehungsweise Y-Achse und das Gieren veranschaulicht eine Drehung um die Z-Achse [21]. Bei dieser Darstellungsweise können mehrdeutige Zustände erreicht werden. Wird beispielsweise eine beliebige Drehung um die Z-Achse und eine darauffolgende 90°-Drehung um die Y-Achse eines dreidimensionalen Körpers betrachtet, so ist eine dritte Rotation um die X-Achse redundant [26]. Diese Problematik wird als Singularität bezeichnet und tritt bei der Darstellung durch Euler-Winkeln auf, wenn zwei oder mehr Rotationsachsen kollinear zueinander sind [27].

Um diese Problematik zu umgehen gibt es die Möglichkeit, Rotationen alternativ als Quaternionen zu beschreiben. Dieser Zahlenbereich wird für die Darstellung von Orientierungen benutzt, da bei dieser keine Singularität auftritt. [21]

$$q = d_1 + d_2 \cdot i + d_3 \cdot j + d_4 \cdot k \text{ mit } i^2 = j^2 = k^2 = -1 \tag{2.10}$$

$$\vec{p} = \begin{pmatrix} \frac{d_2}{\sin(\frac{\omega}{2})} \\ \frac{d_3}{\sin(\frac{\omega}{2})} \\ \frac{d_4}{\sin(\frac{\omega}{2})} \end{pmatrix} \text{ mit } \omega = 2 \cdot \arccos(d_1) \tag{2.11}$$

Aus Gleichung (2.10) lässt sich erkennen, dass Quaternionen aus einem Realteil und einem Imaginärteil, bestehend aus drei Komponenten, zusammengesetzt sind.

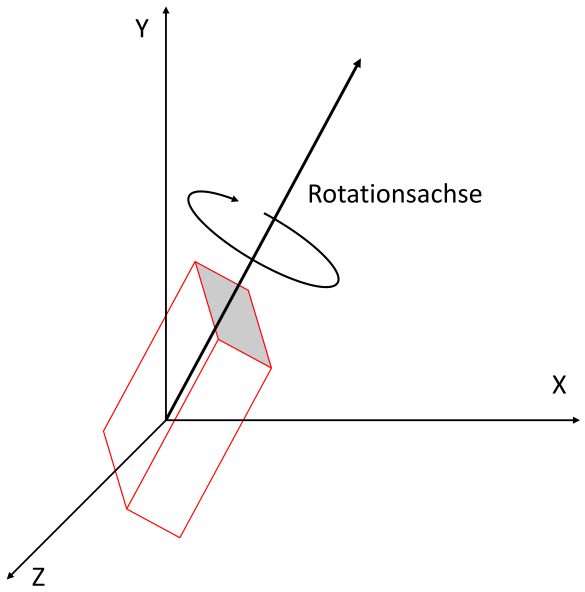


Abbildung 2.15: Darstellung einer Rotation durch Quaternionen. Der zu rotierende Körper liegt im Koordinatenursprung. Die Rotationsachse erfährt eine Drehung um den Winkel ω

Mithilfe des Realteils einer Quaternion lässt sich nach Gleichung (2.11) der Drehwinkel um die in Abbildung 2.15 dargestellte Rotationsachse berechnen. Die Lage des Vektors wird anhand der drei Komponenten des Imaginärteils aus Gleichung (2.11) ermittelt. Anhand der Nutzung von Quaternionen lässt sich die Orientierung eines Körpers im dreidimensionalen Raum eindeutig darstellen. [21, 27]

3 Konzeptionierung

Die Anforderungen an das ALF werden unter anderem mit Hilfe der Umfeldmodellierung aus der „Conceptual design specification technique for the engineering of complex Systems“ (CONSENS) erhoben und als Anforderungsliste im Lastenheft festgehalten. Die CONSENS Methode wird zur systematischen Spezifikation von komplexen Systemen angewendet und an der Hochschule Bochum am Institut für Systemtechnik geschult. Im Umfeldmodell der Bachelorarbeit, das in Abbildung A.3 zu sehen ist, werden die Komponenten für die Umsetzung der Schlupfregelung und SLAM-Kartografierung abgebildet.

Innerhalb der *MotorController Module* ist eine Drehmomentregelung implementiert. Die vier verbauten Motoren und deren Antriebseinheit weisen jedoch ein unterschiedliches Anlaufverhalten auf [4]. Dies hat zur Folge, dass die vier Räder bei gleicher Drehmomentvorgabe unterschiedlich schnell drehen und sich das Fahrzeug dadurch unvorhersehbar dreht [4]. Um diesen Schlupf entgegenzuwirken, wird eine übergeordnete Drehzahl- und Lageregelung implementiert. Abbildung A.4 zeigt die entwickelte Wirkstruktur der Schlupfregelung. Dieser Struktur ist die funktionelle Planung der Regelung zu entnehmen. Es behandelt die CAN-Bus und ROS Schnittstellen sowie die übergeordnete Bewegungsplanung für das autonome Logistik-Fahrzeug. Bisher konnte das ALF ausschließlich mit einem Joystick gesteuert werden. Mithilfe einer Modusumschaltung an der Fernbedienung wird zwischen „manuellen Betrieb“ und „automatisierten Betrieb“ unterschieden. Der „manuelle Betrieb“ beschreibt eine Steuerung des Fahrzeugs durch den Joystick. Während des „automatisierten Betriebs“ kann das ALF manuell oder durch eine automatisierte Anwendung eingegebene Posen anfahren. Eine Pose beschreibt die Position und Orientierung eines Objekts [15]. Die Lageregelung generiert aus dem von der Bewegungsplanung vorgegebenen Bewegungsvektor vier Soll-Drehzahlen. Es besteht die Möglichkeit, die Ist-Lage über den Beschleunigungssensor des *Raspberry Pis* oder aus dem verwendeten SLAM-Algorithmus zu verwenden.

Abbildung A.5 zeigt die entwickelte Wirkstruktur für die Navigation mithilfe von ROS. In dieser Struktur werden die Daten der Sensorik zur Erkennung der Umgebung aus Abschnitt 2.5 verarbeitet. Der ROS-Knoten *Hector Slam*, der in Kapitel 4.3 behandelt wird, generiert durch die Lösung des SLAM-Problems eine geschätzte Ist-Pose. Diese wird für die übergeordnete Lageregelung genutzt. Zudem liefert der Knoten *Move Base* eine Trajektorie für die in Kapitel 4.3 beschriebene Bewegungsplanung. Zur Visualisierung der aktuellen Karte, Ziel-Posen, Trajektorien und weiterer kompatibler Topics wird der Knoten *Rviz* verwendet.

4 Integration in das vorhandene System

Um die in Kapitel 3 beschriebenen Probleme des Anlaufverhaltens entgegenzuwirken, wird zur bereits bestehenden Drehmomentregelung eine übergeordnete Drehzahl- und Lageregelung implementiert. Grundlage für den Reglerentwurf bilden die Übertragungsfunktionen der Antriebe. Diese werden mit den in Abschnitt 2.2 beschriebenen Testfunktionen und anschließender Annäherung ermittelt. Die ungewollten Rotationen werden durch die übergeordneten Regelkreise ausgeregelt. Dieses ist notwendig um definierte Bewegungsbefehle mit der in dem Lastenheft geforderten Präzision abzufahren.

4.1 Analyse und Regelung des Fahrverhaltens

Um das Systemverhalten der vorhandenen Antriebe zu analysieren, wurde eine Sprungantwort aufgenommen. Dafür wurde der maximale Drehmomentsollwert an die Motorcontroller gesendet und die resultierenden Drehzahlen der Motoren aufgezeichnet. Die Sprungantworten wurden gleichzeitig unter Last bei einer Vorwärtsfahrt in dem Labor des Instituts für Systemtechnik aufgenommen. Die Last bestand lediglich aus dem Eigengewicht des Fahrzeugs. Die Abbildungen 4.1, 4.2, 4.3 und 4.4 zeigen die aufgenommenen Sprungantworten der einzelnen Antriebe.

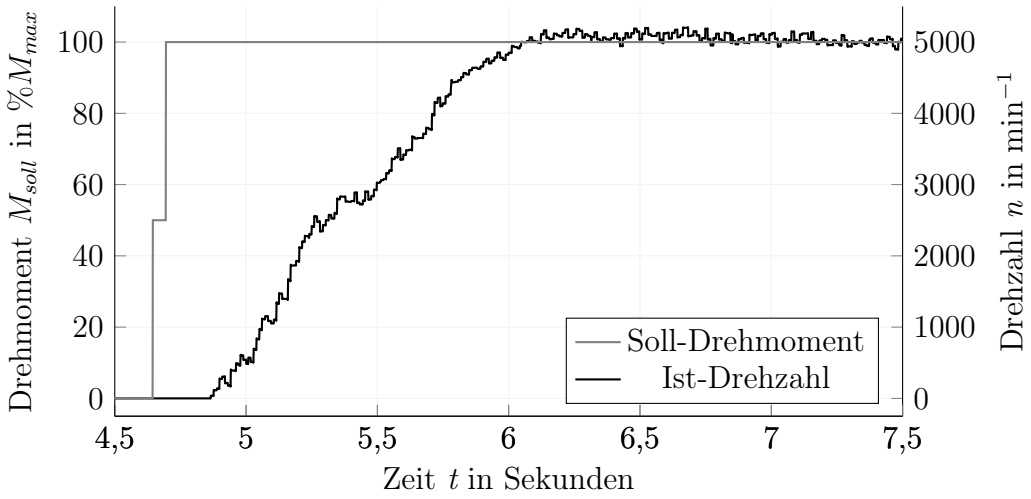


Abbildung 4.1: Sprungantwort des Motors vorne rechts und maximale, sprungförmige Drehmomentvorgabe. Bei der Sollwertvorgabe handelt es sich um einen prozentualen Anteil des maximal möglichen Drehmoments. Die Stufe an dem sprungförmigen Eingangssignal entsteht durch die Eingabe des Sollwerts über den Joystick. Mit diesem Eingabegerät ist es nicht möglich einen idealen Einheitssprung zu erzeugen, wie die sprungförmige Eingangsgröße $u(t)$ aus Abschnitt 2.2 vorsieht. Für weitere Analysen wird diese Abweichung vernachlässigt.

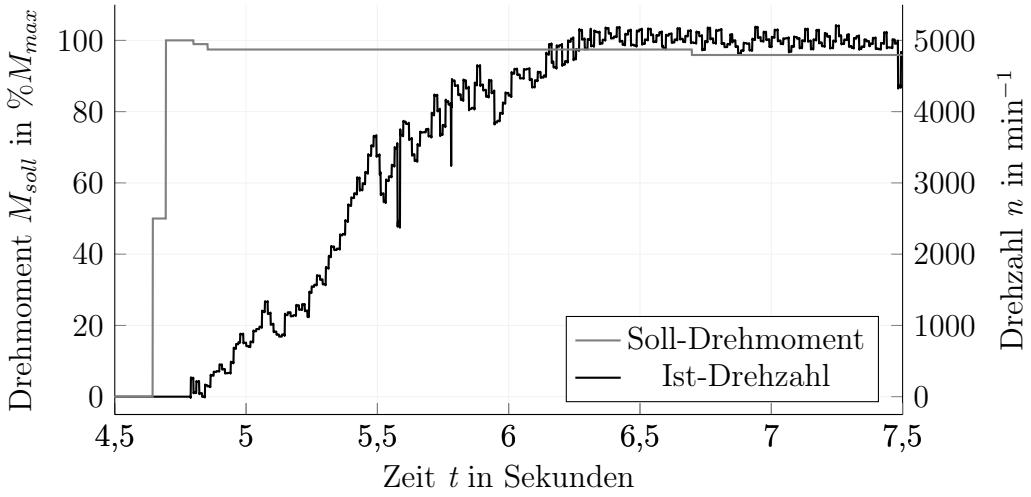


Abbildung 4.2: Sprungantwort des Motors hinten links und maximale, sprungförmige Drehmomentvorgabe.

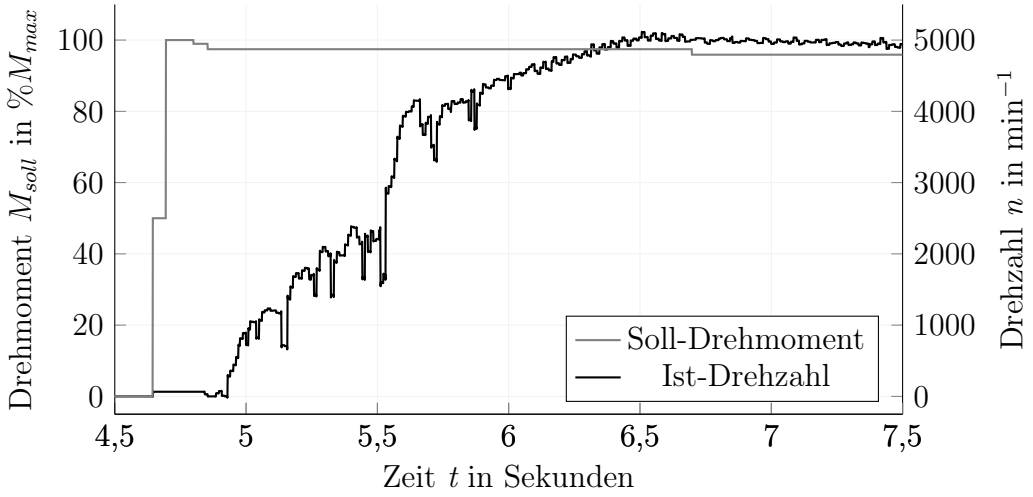


Abbildung 4.3: Sprungantwort des Motors vorne rechts und maximale sprungförmige Drehmomentvorgabe.

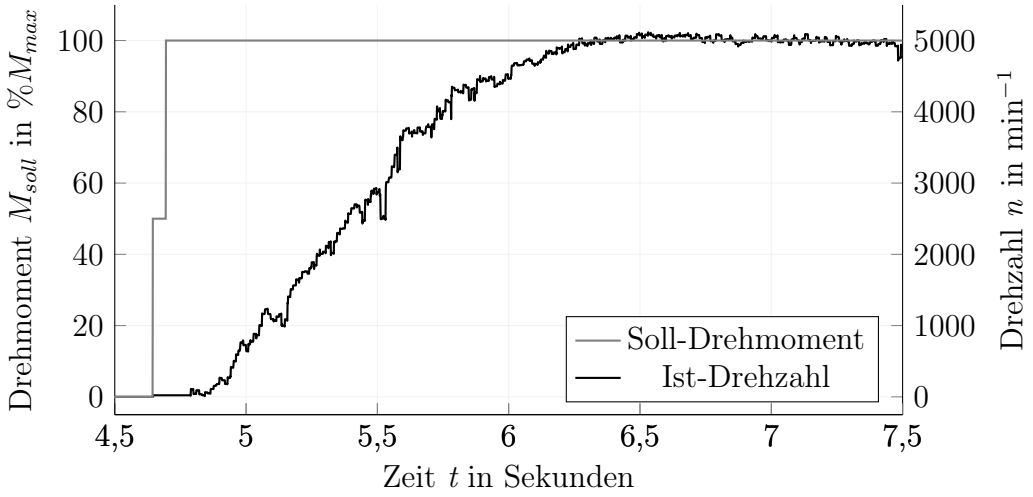


Abbildung 4.4: Sprungantwort des Motors vorne links und maximale sprungförmige Drehmomentvorgabe.

Anhand der Sprungantworten aus den Abbildungen 4.1, 4.2, 4.3 und 4.4 und der Vernachlässigung von Totzeiten, lässt sich das Verhalten von Verzögerungsgliedern 2. Ordnung identifizieren. Deren Übertragungsfunktion mit der allgemeinen Form

$$G(s) = \frac{K_s}{T^2 s^2 + 2DTs + 1} \tag{4.1}$$

beschrieben wird [9, 10]. Eine Berechnung der Fehlerleistung des angenäherten Signals im Verhältnis zu der gemessenen Ist-Drehzahl ergab für die vier Antriebe einen Wert von 0,7%. Da die im Lastenheft beschriebenen Anforderungen auch mit Vernachlässigung der Totzeiten erreicht wurden, ist eine Optimierung der Annäherung nicht notwendig. Durch diese Identifikation werden die Übertragungsfunktionen für die Sprungantworten aus den Abbildungen 4.1, 4.2, 4.3 und 4.4 geschätzt. Hierzu sind Kenntnisse über die Anzahl der Pol- und Nullstellen der Übertragungsfunktion erforderlich. Bei einem Verzögerungsglied 2. Ordnung treten aufgrund der gebrochen-rationalen Funktion aus Gleichung (4.1) zwei Polstellen und keine Nullstelle auf [9]. Durch Annähern mit *Matlab* erhält man die Übertragungsfunktionen (4.2), (4.3), (4.4) und (4.5):

$$G_1(s) = \frac{248{,}4}{s^2 + 3{,}057s + 5{,}104} \tag{4.2}$$

$$G_2(s) = \frac{232{,}4}{s^2 + 3{,}005s + 4{,}702} \tag{4.3}$$

$$G_3(s) = \frac{314}{s^2 + 3{,}816s + 6{,}208} \tag{4.4}$$

$$G_4(s) = \frac{251{,}6}{s^2 + 3{,}175s + 5{,}196} \tag{4.5}$$

Zur Verifikation der approximierten Übertragungsfunktionen in *Simulink*, wurde eine Simulation mit den gleichen Simulationsparametern wie für die Aufnahme der Sprungantwort durchgeführt. Es zeigte sich ein äquivalentes Übertragungsverhalten der geschätzten und experimentell ermittelten Sprungantworten. Diese sind in den Abbildungen 4.5, 4.6, 4.7 und 4.8 zu sehen.

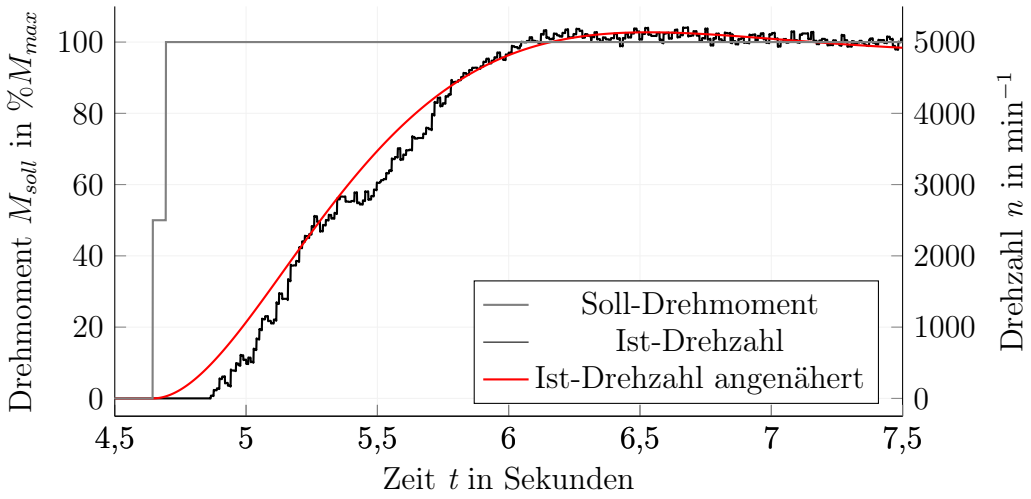


Abbildung 4.5: Sprungantwort des Motors hinten rechts mit der angenäherten Übertragungsfunktion $G_1(s)$ und dem Eingangssignal aus der experimentell ermittelten Sprungantwort. Der zeitliche Verlauf der Übertragungsfunktion wurde mit Hilfe von *Simulink* ermittelt. Dafür wurde die Simulationszeit, Schrittweite und der Start des Eingangssignals der originalen Sprungantwort als Simulationsparameter in *Simulink* eingegeben.

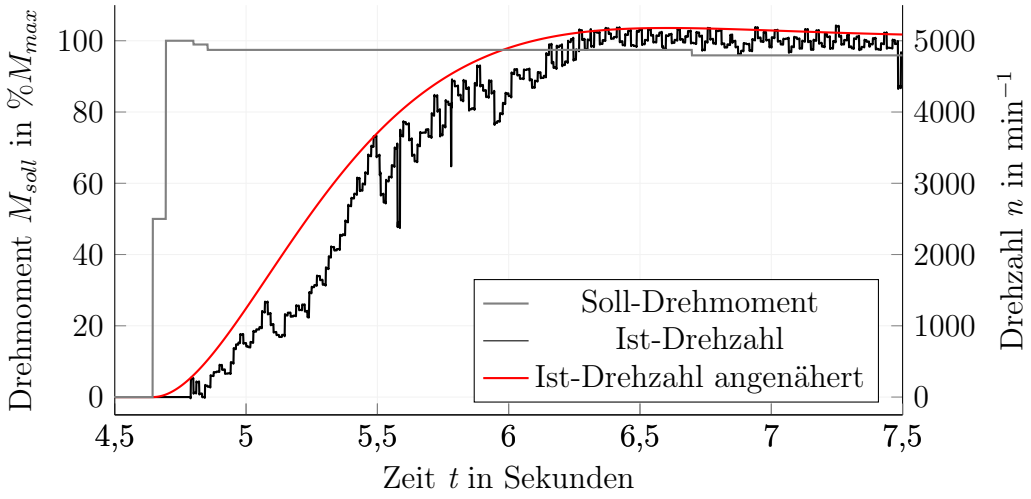


Abbildung 4.6: Sprungantwort des Motors hinten links mit der angenäherten Übertragungsfunktion $G_2(s)$ und dem Eingangssignal aus der experimentell ermittelten Sprungantwort.

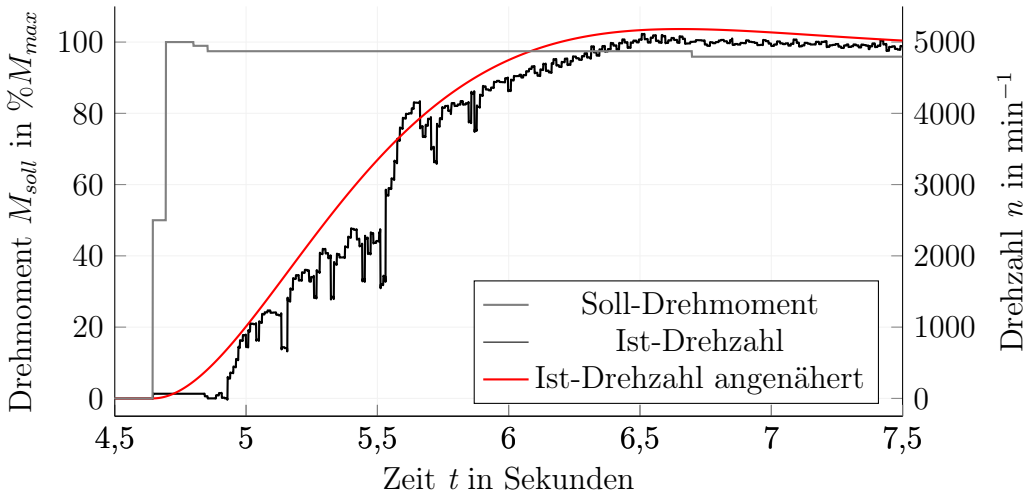


Abbildung 4.7: Sprungantwort des Motors vorne rechts mit der angenäherten Übertragungsfunktion $G_3(s)$ und dem Eingangssignal aus der experimentell ermittelten Sprungantwort.

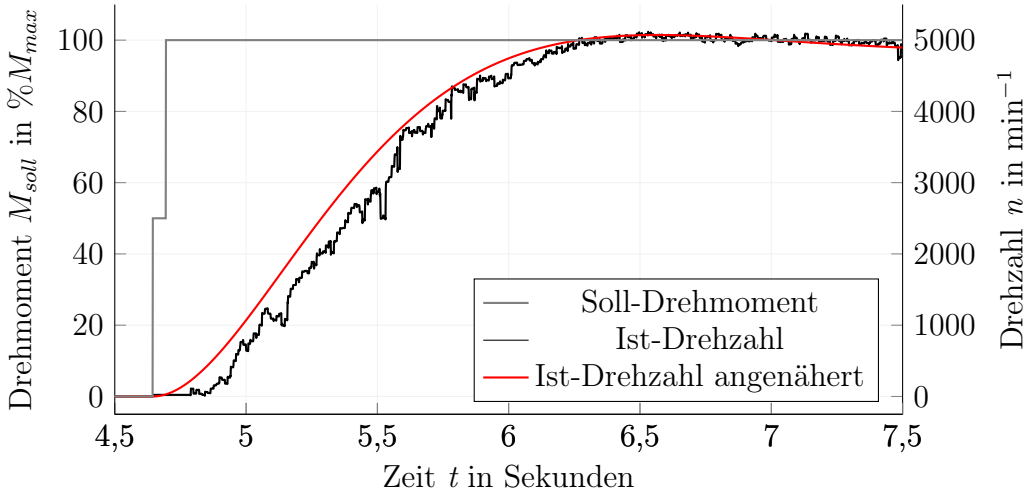


Abbildung 4.8: Sprungantwort des Motors vorne links mit der angenäherten Übertragungsfunktion $G_4(s)$ und dem Eingangssignal aus der experimentell ermittelten Sprungantwort.

Die Übertragungsfunktionen werden in die Form aus Gleichung (4.1) umgestellt, sodass die Zeitkonstante, Dämpfung und Streckenverstärkung bestimmt werden kann.

$$G_1(s) = \frac{48{,}6677}{0{,}1959s^2 + 0{,}5989s + 1} \rightarrow D_1 = 0{,}7071 \text{ und } T_1 = 0{,}4261 \text{ s} \tag{4.6}$$

$$G_2(s) = \frac{49{,}4258}{0{,}2127s^2 + 0{,}6391s + 1} \rightarrow D_2 = 0{,}6929 \text{ und } T_2 = 0{,}4612 \text{ s} \tag{4.7}$$

$$G_3(s) = \frac{50{,}5799}{0{,}1611s^2 + 0{,}6147s + 1} \rightarrow D_3 = 0{,}7691 \text{ und } T_3 = 0{,}4014 \text{ s} \tag{4.8}$$

$$G_4(s) = \frac{48{,}4219}{0{,}1925s^2 + 0{,}6110s + 1} \rightarrow D_4 = 0{,}6965 \text{ und } T_4 = 0{,}4387 \text{ s} \tag{4.9}$$

Die nun vorhandenen mathematischen Modelle der Übertragungsfunktionen werden für die Simulation verschiedener Regler in *Matlab/Simulink* und die anschließende Verwendung im ALF genutzt. Die Werte der Dämpfung D liegen zwischen null und eins. Daraus resultiert ein schwingendes Verhalten der Übertragungsfunktion. Das Abklingen der Schwingungsverläufe wird durch die Größe T beschrieben und wird auch als Abklingzeitkonstante bezeichnet [9].

4.1.1 Regelung der Drehzahl

Die Drehzahlregelung wird mithilfe eines PI-Reglers realisiert, da dieser die Vorteile eines P- und I-Reglers vereint [28]. Aufgrund des P-Anteils reagiert der Regler unmittelbar und hat durch die zeitliche Integration der Regelabweichung keine bleibende Regeldifferenz. Durch die vorhandenen mathematischen Modelle können verschiedene PI-Regler an den Regelstrecken simuliert werden. Es werden PI-Regler mit den Einstellregeln nach Chien, Hrones und Reswick aus Tabelle 4.1 und empirisch am Fahrverhalten ausgelegt.

Regler		Aperiodischer Regelverlauf		Regelverlauf mit 20% Überschwingen	
		Störung	Führung	Störungen	Führung
P	K_p	$0{,}3\frac{T_g}{T_u K_s}$	$0{,}3\frac{T_g}{T_u K_s}$	$0{,}7\frac{T_g}{T_u K_s}$	$0{,}7\frac{T_g}{T_u K_s}$
PI	K_p	$0{,}6\frac{T_g}{T_u K_s}$	$0{,}35\frac{T_g}{T_u K_s}$	$0{,}7\frac{T_g}{T_u K_s}$	$0{,}6\frac{T_g}{T_u K_s}$
	T_n	$4T_u$	$1{,}2T_g$	$2{,}3T_u$	T_g
PID	K_p	$0{,}95\frac{T_g}{T_u K_s}$	$0{,}6\frac{T_g}{T_u K_s}$	$1{,}2\frac{T_g}{T_u K_s}$	$0{,}95\frac{T_g}{T_u K_s}$
	T_n	$2{,}4T_u$	T_g	$2T_u$	$1{,}35T_g$
	T_v	$0{,}42T_u$	$0{,}5T_u$	$0{,}42T_u$	$0{,}47T_u$

Tabelle 4.1: Die Einstellregeln wurden von Chien, Hrones und Reswick für verschiedene Übertragungsfunktionen ermittelt. Die Einstellwerte sind für Führungs- und Störverhalten angegeben. Dabei wird unterschieden, ob eine Störung möglichst schnell ausgeglichen wird oder ob der Regler einem zeitlich veränderlichen Sollwert gut folgen soll. [28]

Für die Einstellregeln sind Informationen über die Verzugszeit T_u und die Anstiegszeit T_g notwendig [9, 28]. Diese Größen werden durch die Anwendung des Wendetangentenverfahrens an der approximierten Regelstrecken gewonnen. Das Verfahren wird in dieser Bachelorarbeit für eine Übertragungsfunktion gezeigt. Das Vorgehen für die verbliebenen drei Regelstrecken ist analog.

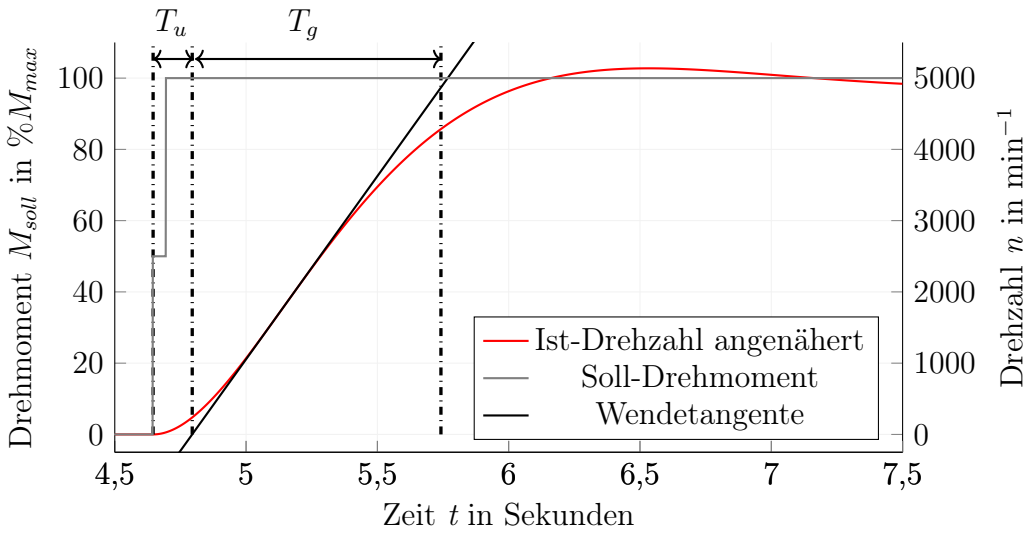


Abbildung 4.9: Dargestellt ist die angenäherte Übertragungsfunktion des Antriebs hinten rechts mit Soll-Drehmoment und Wendetangente. Durch Bestimmen und Einzeichnen der Wendetangente wird die Verzugs- und Anstiegszeit ermittelt. In Form einer Strichpunktlinie sind Hilfslinien eingezeichnet, mit denen die Zeiten von der Zeitachse abgelesen werden. Für die Übertragungsfunktion $G_1(s)$ wird eine Anstiegszeit $T_g = 0,9469$ s und eine Verzugszeit $T_u = 0,1490$ s bestimmt.

Mithilfe der in Abbildung 4.9 eingezeichneten Wendetangente können die Zeitkonstanten bestimmt werden, die nötig sind, um die Nachstellzeit T_n und Verstärkung K_P für den PI-Regler zu bestimmen. Durch eine Aktualisierung des Bewegungsziels, wie in Abschnitt 4.1.2 erklärt, wird die Führungsgröße laufend geändert. Aufgrund der sich ändernden Führungsgröße wurden die Einstellregeln, gemäß der Tabelle 4.1, für Führungsverhalten mit aperiodischen und 20% überschwingenden Regelgrößenverlauf gewählt [28]. Die Gleichungen (4.10), (4.11) und (4.12) zeigen die ermittelten Regler sowie den empirisch, durch Beobachtung des Fahrverhaltens, entwickelten Regler.

$$G_{pi1}(s) = \frac{0{,}07419s + 0{,}07835}{0{,}9469s} \tag{4.10}$$

$$G_{pi2}(s) = \frac{0{,}05193s + 0{,}0457}{1{,}136s} \tag{4.11}$$

$$G_{pi3}(s) = \frac{0{,}1s + 0{,}005}{s} \tag{4.12}$$

Die Simulation mit den drei Reglern $G_{pi1}(s)$, $G_{pi2}(s)$ und $G_{pi3}(s)$ an der approximierten Regelstrecke aus Gleichung (4.6), ergibt die in Abbildung 4.10 dargestellten Regelgrößenverläufe der drei geschlossenen Regelkreise. Dabei ist eine deutliche Regeldifferenz bei dem empirisch ermittelten Regler $G_{pi3}(s)$ zu beobachten. Da dieses Verhalten nicht den im Lastenheft erhobenen Anforderungen an die Regelung entspricht, ist dieser Regler nicht für eine Integration in das ALF geeignet. Der Regelgrößenverlauf mit dem Regler $G_{pi1}(s)$ zeigt ein Überschwingen auf bis zu 2000 min^{-1} bei einem Sollwert von 1500 min^{-1}. Dieses Verhalten genügt nicht den Anforderungen aus dem Lastenheft A.1.3, weshalb die Integration des Reglers nicht vorgesehen ist. Der Regler $G_{pi2}(s)$ zeigt das gewünscht Verhalten und wird im ALF verwendet.

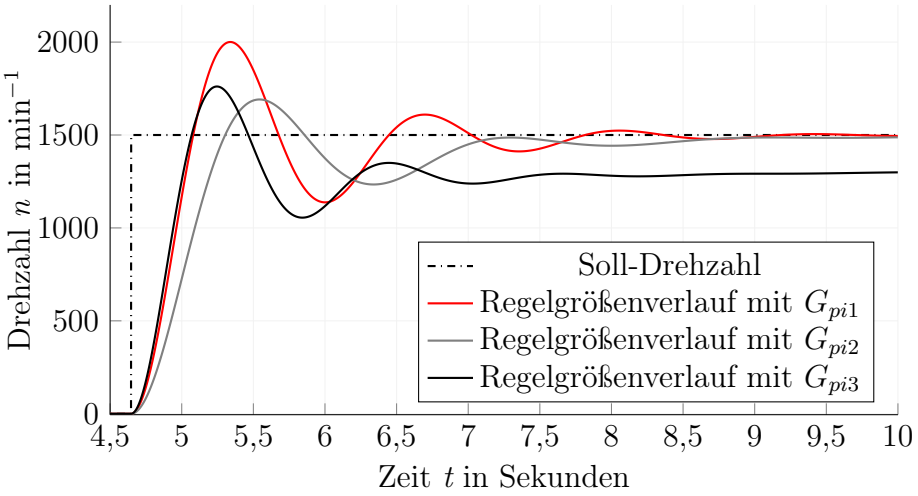


Abbildung 4.10: Regelgrößenverlauf des Motors hinten rechts an der geschätzten Übertragungsfunktion $G_1(s)$ und dem sprungförmigen Eingangssignal mit der Soll-Drehzahl $n = 1500$ min^{-1}. Zu sehen sind drei Regelgrößenverläufe des geschlossenen Regelkreises mit den Reglern (4.10), (4.11) und (4.12).

Die Implementierung des Reglers im ALF erfolgte mit einem LTI-System aus der *Control System Toolbox* von *Matlab/Simulink*, wobei die übergeordnete Lageregelung aus Abschnitt 4.1.2 vier Soll-Drehzahlen als Führungsgröße vorgibt. Stell- und Regelgröße werden in dem *Simulink*-Modell in den CAN-Bus eingegeben beziehungsweise ausgelesen, wodurch der Regelkreis geschlossen ist.

Abbildung 4.11 zeigt den Regelgrößenverlauf nach Implementierung des Reglers $G_{pi2}(s)$ am ALF. Zum Zeitpunkt $t = 0{,}8$ s ist ein Überschwingen der Regelgröße zu beobachten. Dies spiegelt den gewünschten Regelgrößenverlauf nach den Einstellregeln wieder und ist in den an der approximierten Regelstrecke simulierten Regelgrößenverläufe aus Abbildung 4.10 zu erkennen. Der Sollwert schwankt, da die übergeordnete Lageregelung bereits die Führungsgröße vorgibt. Die Eingangsgröße ist somit nicht mehr konstant.

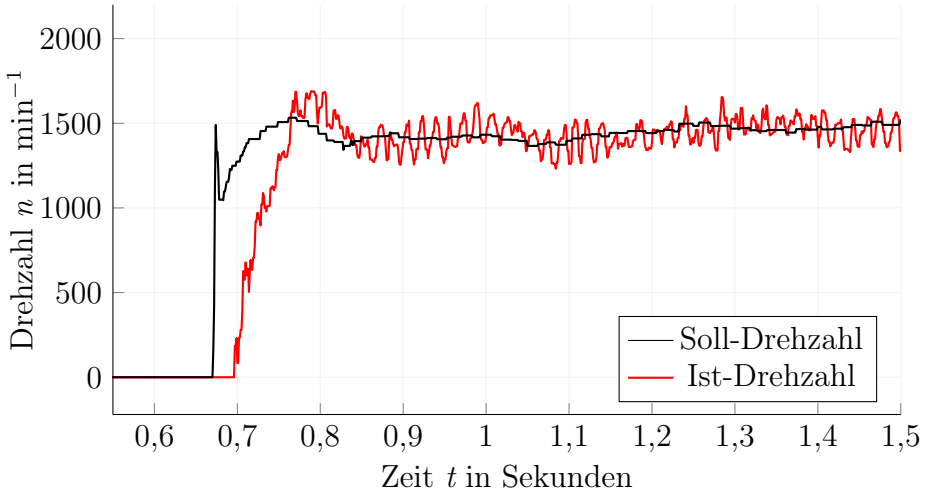


Abbildung 4.11: Regelgrößen- und Sollwertverlauf nach Integration des Reglers $G_{pi2}(s)$. Der Sollwert wird bereits aus der übergeordneten Lageregelung generiert. Dies hat zur Folge, dass die Führungsgröße nicht konstant ist.

4.1.2 Umsetzung der Lageregelung

Für die Umsetzung der verschiedenen Lenkungsprinzipien aus Kapitel 2.4 wird für dieses Projekt die Bewegung des Fahrzeugs in zwei Komponenten unterteilt. Für die Beschreibung des Rotationswinkels, zwischen dem globalen Koordinatensystem der Karte und dem lokalen, wird der Posenwinkel β eingeführt. In Abbildung 4.12 ist dieser unten rechts dargestellt. Der Fahrtwinkel α stellt die Fahrtrichtung beziehungsweise die Ausrichtung des resultierenden Geschwindigkeitsvektors der Bewegung im Verhältnis zum lokalen Koordinatensystem des Fahrzeugs dar.

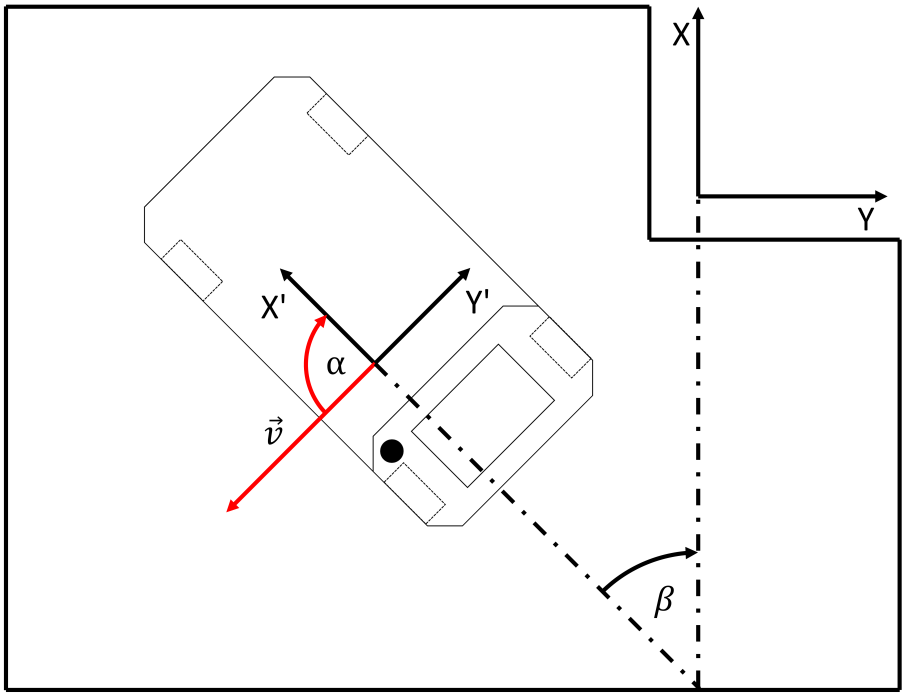


Abbildung 4.12: Darstellung des Posen- und Fahrtwinkels aus der Draufsicht. Im Zentrum der Abbildung ist das ALF mit dem lokalen Koordinatensystem des Fahrzeugs zu sehen. Der Fahrtwinkel α liegt zwischen der X-Achse des lokalen Koordinatensystems und dem als roten Pfeil dargestellten Geschwindigkeitsvektor des Fahrzeugs. Oben rechts in der Darstellung befindet sich das globale Koordinatensystem. Der Posenwinkel ist hier mit β gekennzeichnet.

Mit dieser Unterteilung wird ein translatorisches Bewegungsziel, mithilfe des Fahrtwinkels, und ein rotatorisches durch den Posenwinkel vorgegeben. Das translatorische Bewegungsziel kann durch zwei Quellen generiert werden. Die Daten des Joysticks und die des Knotens *Move Base*, der in Kapitel 4.4 erklärt wird, werden hierfür verwendet. Durch die Betriebsmodi aus Kapitel 3 werden X- und Y-Richtungskomponenten aus den Daten des Joysticks oder des Knotens generiert. Mithilfe dieser Komponenten und des Simulinkblocks *Polar to Cartesian* wird der Fahrtwinkel α bestimmt. Dieser wird durch die Multiplikation des Vektors $\vec{a}$ und der Hilfsmatrix $\underline{H}$, wie in Gleichung (4.13), in ein translatorisches Bewegungsziel $\vec{t}_{ma}$ überführt. Die Hilfsmatrix $\underline{H}$ wird für die Umrechnung in eine in der Praxis anwendbare mathematische Form verwendet. Dessen Zeilen entsprechen jeweils einem Rad des Fahrzeugs in der Reihenfolge von oben nach unten vorne rechts, vorne links, hinten rechts und hinten links. Die Spalten enthalten die Multiplikatoren für die Komponenten in Vektor $\vec{a}$.

$$\vec{t}_{ma} = \underline{H} \cdot \vec{a} \quad \text{mit} \quad \underline{H} = \begin{pmatrix} 1 & -1 \\ 1 & 1 \\ 1 & 1 \\ 1 & -1 \end{pmatrix} \quad \text{und} \quad \vec{a} = \begin{pmatrix} \cos(\alpha) \\ \sin(\alpha) \end{pmatrix} \tag{4.13}$$

Das rotatorische Bewegungsziel wird ebenfalls manuell über den Joystick oder durch den Knoten *Move Base* erzeugt. Das durch die manuelle Steuerung erzeugte rotatorische Bewegungsziel wird als Vektor $\vec{r}_m$ gekennzeichnet. Hierbei wird der Soll-Posenwinkel durch Drücken der entsprechenden Tasten am Joystick in *Simulink* inkrementiert beziehungsweise dekrementiert. Der Ist-Posenwinkel wird aus dem ROS-Knoten *Hector Slam* gewonnen, der in Kapitel 4.4 beschrieben wird. Durch eine Multiplikation der Differenz aus Soll- und Ist-Posenwinkel mit einem ermittelten Faktors P_m und des Hilfsvektors $\vec{h}$ entsteht das Bewegungsziel $\vec{r}_m$ aus Gleichung (4.14). Der Faktor P_m ist durch die Winkeldifferenz $\Delta\beta$ zwischen Soll- und Ist-Posenwinkel begründet. Gemäß des Anhangs A.1.3 darf dieser den Wert von 10° nicht übersteigen. Mit der Differenz nimmt die Dominanz des rotatorischen Bewegungsziels zu beziehungsweise ab. Ab einem Wert von maximal 10° wird nur das rotatorische Bewegungsziel umgesetzt.

$$\vec{r}_m = \Delta\beta \cdot P_m \cdot \vec{h} \quad \text{mit} \quad P_m = \frac{1}{10^\circ} \quad \text{und} \quad \vec{h} = \begin{pmatrix} -1 \\ 1 \\ -1 \\ 1 \end{pmatrix} \tag{4.14}$$

Für die automatische Steuerung des Fahrzeugs wird mithilfe des Knotens *Move Base* eine Trajektorie geplant. Die Erzeugung einer solchen Trajektorie wird in Kapitel 4.4 erläutert. Aus dem Topic *cmd_ vel* des Knotens lässt sich ein rotatorischer Bewegungsbefehl c für die Verfolgung einer Trajektorie auslesen. Analog zur Winkeldifferenz wird dieser Wert mit einem ermittelten Faktor P_a skaliert und mit einem Hilfsvektor $\vec{h}$ multipliziert. Das Ergebnis ist das daraus folgende Bewegungsziel $\vec{r}_a$ und wird in Gleichung (4.15) gezeigt. Der Betrag des Faktors P_a wurde nach dem Rotationsbefehl aus dem Topic *cmd_ vel* ausgelegt, das in Kapitel 4.4.1 erläutert wird. In den Parametern von *Move Base* ist für den Rotationsbefehl ein Höchstwert von 0,5 hinterlegt. Aus der Dokumentation des entsprechenden Knotens ist nicht ersichtlich, wann der Höchstwert eintritt [29]. Fordert das Topic diesen Wert als rotatorischen Bewegungsbefehl c, wird es aufgrund des Faktors P_a vorrangig abgearbeitet.

$$\vec{r}_a = c \cdot P_a \cdot \vec{h} \quad \text{mit} \quad P_a = -2 \quad \text{und} \quad \vec{h} = \begin{pmatrix} -1 \\ 1 \\ -1 \\ 1 \end{pmatrix} \tag{4.15}$$

Die drei Ziele können mathematisch durch eine Addition der Vektoren vereint werden und ergeben ein neues, gemeinsames Bewegungsziel $\vec{b}$.

$$\vec{t}_{ma} + \vec{r}_m + \vec{r}_a = \vec{b} \tag{4.16}$$

Wird der Vektor des resultierende Bewegungsziel $\vec{b}$ mit der geforderten Drehzahl multipliziert erhält man die Soll-Drehzahl aller Räder. Diese werden in der Drehzahlregelung, wie in Abschnitt 4.1.1 beschrieben, als Eingangsgröße verwendet.

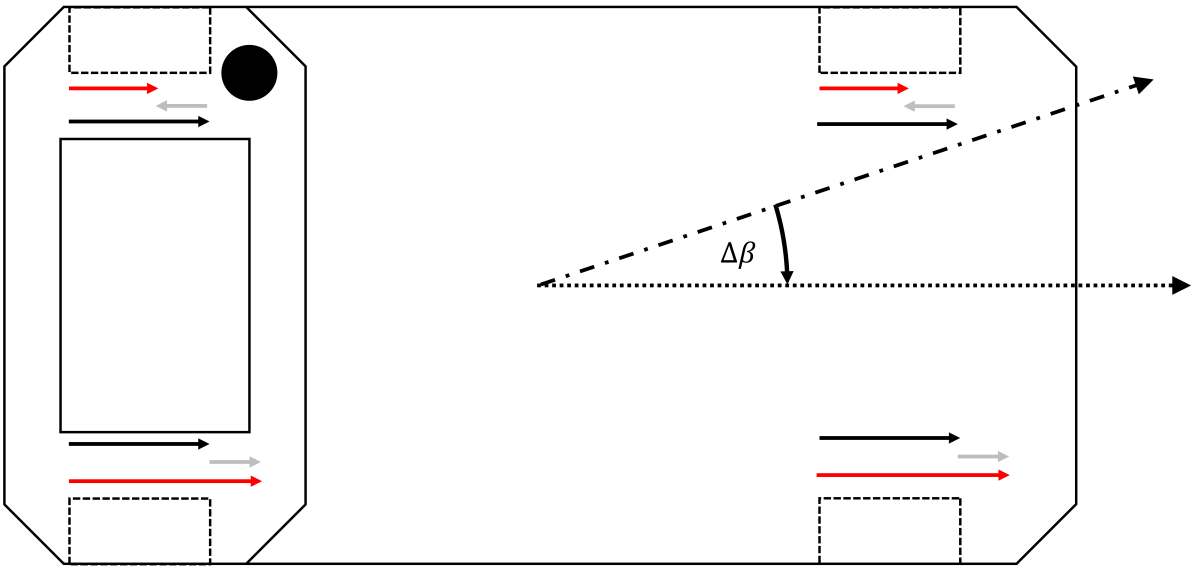


Abbildung 4.13: Darstellung des Funktionsprinzips der Lageregelung aus der Draufsicht des ALFs. Der schwarze, gepunktete Vektor zeigt die momentane Ausrichtung der Pose. Der als Strichpunktlinie ausgeführte Pfeil ist als zu erreichende Pose zu deuten. Der dargestellte Winkel $\Delta\beta$ zeigt die Posenwinkeldifferenz. Die gestrichelten Vierecke in den Ecken des Fahrzeugs stellen die Räder dar. Für jedes Rad sind Pfeile in die Abbildung eingetragen, die jeweils symbolisch eine Winkelgeschwindigkeit der entsprechenden Räder repräsentieren. Die schwarzen Pfeile stellen das translatorische Bewegungsziel und die grauen ein rotatorisches dar. Resultierend aus Gleichung (4.16) beschreiben die roten Pfeile die Einträge des Vektors $\vec{b}$.

In Abbildung 4.13 ist die Funktionsweise der Lageregelung prinzipiell dargestellt. Das Fahrzeug soll sich hier beispielhaft mit einem Fahrtwinkel von 0° entlang des Pfeils, der als Strichpunktlinie dargestellt ist, ausrichten und geradeaus fahren. Der Winkel $\Delta\beta$ zeigt die Differenz zwischen Soll- und Ist-Pose. Die schwarzen Pfeile stellen den Betrag und die Richtung der Einträge des aus den Fahrtwinkel errechneten Vektors dar. Der Posenwinkel wird in diesem Beispiel verändert und erzeugt einen Vektor der hier mit roten Pfeilen symbolisch gezeigt wird. Mit der Berechnung nach Gleichung (4.16) ergibt sich der resultierende Vektor $\vec{b}$, dessen Einträge in Abbildung 4.13 durch graue Pfeile dargestellt werden.

4.2 Auswertung der Sensordaten

Die Verarbeitung der Sensorsignale ist für eine kollisionsfreie Bewegung des Fahrzeugs im automatischen Betrieb notwendig. Neben der Visualisierung dieser Signale ist im Folgenden die Integration in das System beschrieben.

4.2.1 Visualisierung der Sensordaten

ROS Visualization, oder auch *Rviz* genannt, ist ein 3D-Visualisierungswerkzeug und wird für die Anzeige von Sensordaten und Statusinformationen genutzt. Zur Visualisierung des Robotermodels in *Rviz* wird eine *Unified Robot Description Format* (URDF) Datei erstellt. In der URDF Datei werden vom Benutzer dreidimensionale Körper programmiert und ausgerichtet. In Abbildung 4.14 ist das mit der URDF-Datei erzeugte und in *Rviz* visualisierte ALF zu sehen.

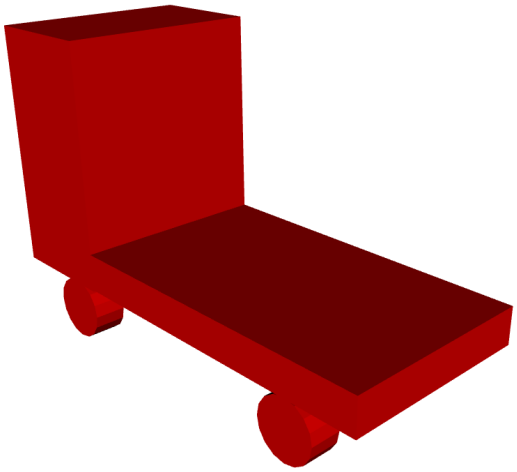


Abbildung 4.14: Visualisierung des in Rviz importierten Robotermodells von schräg oben betrachtet.

Wenn der Knoten Rviz gestartet wird öffnet sich eine Benutzeroberfläche, auf der die entsprechenden Topics abonniert werden können. Für eine bessere Übersicht lassen sich die visualisierten Sensordaten einfärben, um diese später voneinander unterscheiden zu können. Ebenfalls ist es möglich in Rviz eine 2D-Pose einzugeben, die als Navigationsziel für den in Kapitel 4.3 erklärten Trajektorieplaner dient. [30]

4.2.2 Einbindung des RPLIDAR A2

Die Integration der Messwerte des *RPLIDAR A2* in das ROS-Netzwerk wird durch den Knoten *Rplidar* umgesetzt [31]. Bei dem Aufruf des Knotens wird der Motor des 360°-Laserscanners gestartet und das Topic *Scan* mit dem Nachrichtentyp *LaserScan* veröffentlicht [31]. In Abbildung 4.15 sind Beobachtungen von Landmarken des Lidar-Sensors als schwarze Punkte zu sehen.

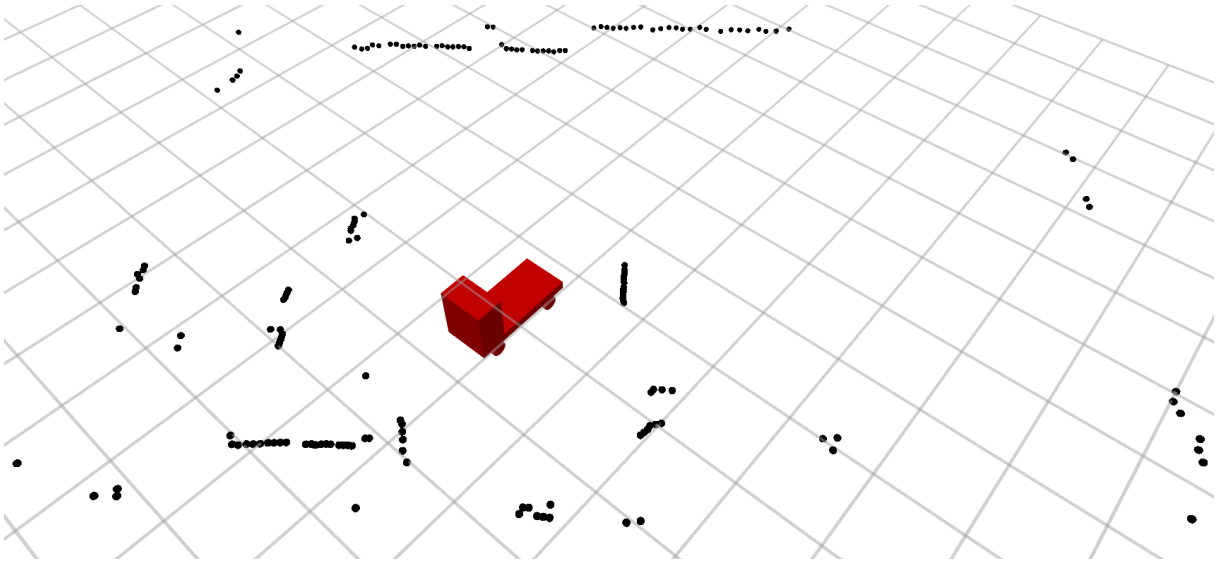


Abbildung 4.15: Die Abbildung zeigt die in *Rviz* visualisierten Sensorsignale des *RPLIDAR A2*. Die schwarzen Punkte stellen diese dar und repräsentieren die von dem Sensor erkannten Objekte. Mittig in der Darstellung ist das durch die URDF-Datei importierte, rote Robotermodell zu sehen.

4.2.3 Integration der Kinect-Sensoren

Für die Integration der von den *Kinect*-Sensoren aufgenommenen Bildinformationen in das ROS-Netzwerk wird in dieser Bachelorarbeit der Knoten *Kinect2 Bridge* verwendet. Das Softwarepaket enthält eine Kalibrierung, eine Registrierung, ein eigenständiges Anzeigewerkzeug für das Kamerabild und einige Knoten für die Veröffentlichung aller Kamerafunktionen als Topic.

Für die Vermeidung von Softwareabstürzen, die bei der Inbetriebnahme von zwei *Kinect*-Sensoren auftreten, wurde der Buffer des *Universal Serial Bus Filesystem* (USBFS) vergrößert [32]. Zudem wird die Bildrate auf maximal 5 Bilder pro Sekunde reduziert, anschließend konnten beide *Kinect*-Sensoren ohne Probleme betrieben werden.

Das vom Knoten *Kinect2 Bridge* veröffentlichte Topic *kinect2/hd/image_depth_rect* ist vom Nachrichtentyp *Image*. Dieses kann von dem in Kapitel 4.4 beschriebenen Knoten *Move Base* nicht verwendet werden, da Nachrichten von dem Typ *LaserScan* benötigt werden. Für die Durchführung einer Typkonvertierung von *Image* zu *LaserScan* wird der Knoten *Depthimage to Laserscan* benutzt. Ein relevanter Parameter für die Typkonvertierung ist *scan_height*, der die genutzte Anzahl der horizontal angelegten Pixelreihen des Bildes beschreibt. Der Knoten ermittelt spaltenweise den kleinsten gemessenen Abstand und veröffentlicht diesen Wert mit dem Nachrichtentyp *LaserScan*. Als Ergebnis der Typkonvertierung können die *Kinect*-Sensoren für die Navigation als Observierungsquelle genutzt werden. Im Rahmen dieser Bachelorarbeit werden die konvertierten Nachrichten des Typs *LaserScan* in den Topics *Scan1* und *Scan2* von den Knoten der beiden *Kinect*-Sensoren veröffentlicht. [33]

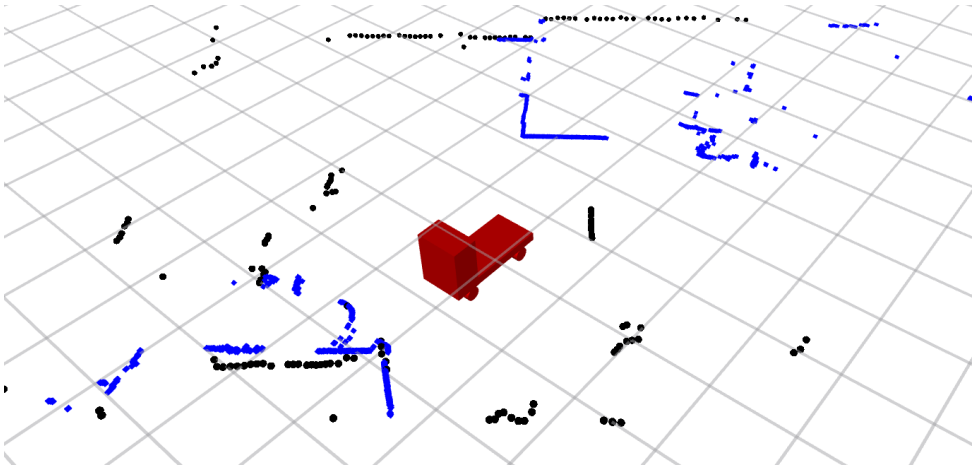


Abbildung 4.16: Darstellung der Sensorsignale in *Rviz* von schräg oben betrachtet. Im Zentrum der Abbildung ist das ALF als Modell dargestellt. Die schwarzen Markierungen sind von *Rviz* abonnierte und visualisierte Sensorsignale des *RPLIDAR A2*. Die blauen Punkte entsprechen den Daten der *Kinect*-Sensoren. Diese Markierungen stellen von den Sensoren erkannte Objekte dar.

In Abbildung 4.16 sind die Sensorsignale des Lidar-Sensors als schwarze und der beiden *Kinect*-Sensoren als blaue Punkte dargestellt. Oben rechts und unten links in der Abbildung ist zu erkennen, dass die Sensordaten teilweise nicht korrelieren. Dies ist auf die *Kinect*-Sensoren zurückzuführen, da Objekte unterhalb der Arbeitshöhe des 360°-Laserscanners erkannt werden. In der Darstellung 4.17 können die in Abbildung 4.16 gezeigten Laserdaten nachvollzogen und realen Objekten zugeordnet werden.

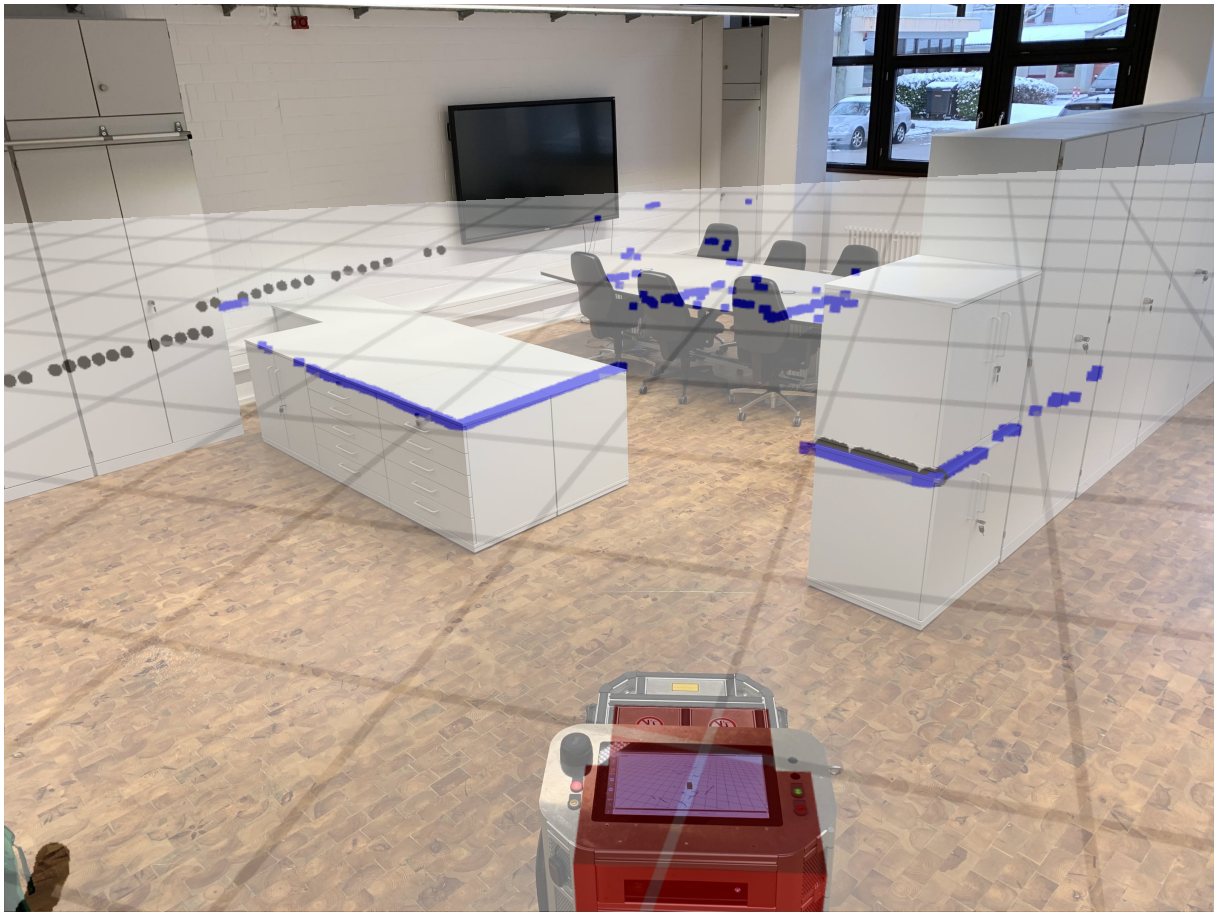


Abbildung 4.17: Visualisierung der *LaserScan*-Topics aus den Abschnitten 4.2.2 und 4.2.3 in *Rviz*. In der Darstellung wurde unter das Bildschirmfoto aus *Rviz* ein Foto aus gleicher Perspektive platziert. Die erfassten Gegenstände der *Kinect*-Sensoren sind hier als blaue Markierungen und die des *RPLIDAR A2* als schwarz dargestellt. Die *Kinect*-Sensoren erkennen im Zentrum der Darstellung einen Schrank, der von dem *RPLIDAR A2* Sensor nicht erfasst wird. In Kapitel 4.3 wird auf die Folgen dieser Beobachtung eingegangen.

4.3 Kartografierung der Umgebung

Das Lastenheft sieht vor, die Umgebung des ALFs mit und ohne eine Bewegungsvorgabe des Benutzers zu kartografieren. In diesem Kapitel werden die für die Umsetzung nötigen Schritte dargelegt.

Zur Kartografierung der Umgebung wird in dieser Bachelorarbeit der Knoten *Hector Slam* verwendet. Dieser abonniert die in den Abschnitt 4.2.2 beschriebene *LaserScan*-Nachrichten aus dem Topic *Scan* und veröffentlicht eine Belegtheitskarte in das Topic *Map* vom Nachrichtentyp *OccupanyGrid*. Diese kann in *Rviz*, wie in Abbildung 4.18, visualisiert oder als Bilddatei exportiert werden. Die Karte erweitert sich durch Bewegungen des ALFs. Der Knoten löst die SLAM Problematik aus Abschnitt 2.6.1 mit den Umgebungsdaten des *RPLIDAR A2* ohne Odometrieinformationen zu benötigen. [34]

Die hier verwendeten Observationsquellen erfassen unterschiedliche Messwerte. Dies ist nicht auf eine fehlerhafte Messung zurückzuführen, sondern auf den in Abschnitt 2.5 beschriebenen unterschiedlichen Arbeitsbereich. Das Veröffentlichen von mehreren Observationsquellen in ein Topic führt unter diesen Umständen zu Komplikationen mit dem verwendeten Algorithmus.

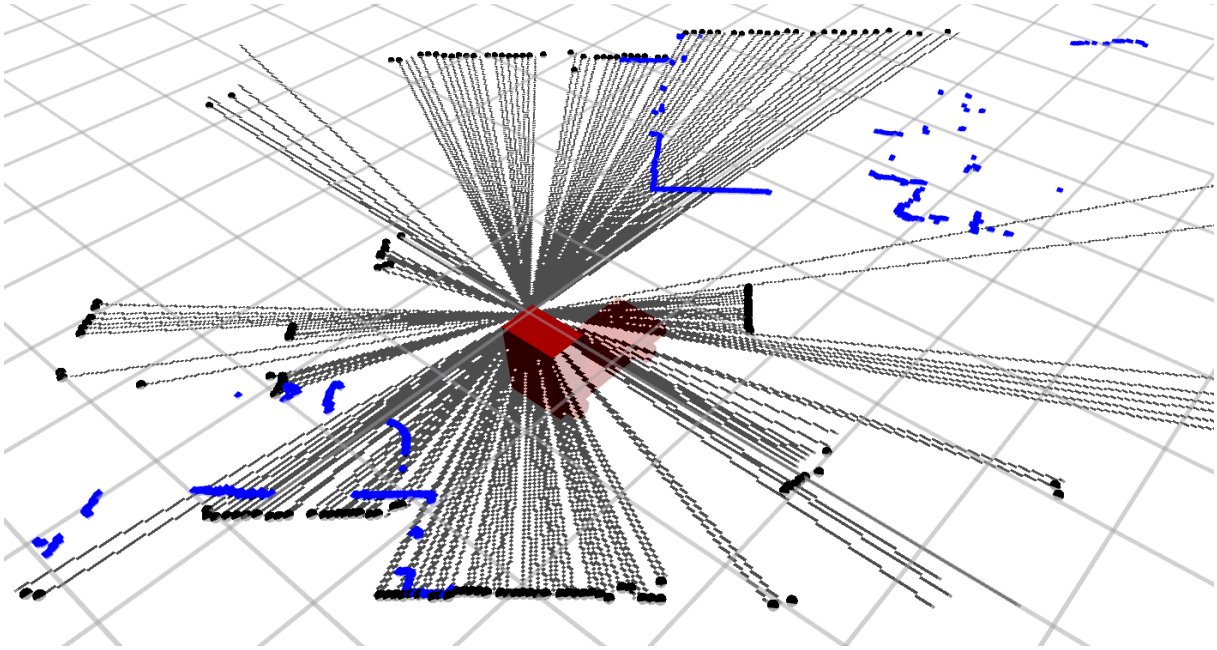


Abbildung 4.18: Abbildung einer in Rviz dargestellten *Occupancy Grid*-Karte aus dem Topic *Map* und der *LaserScan* Nachrichten aus den Abschnitten 4.2.2 und 4.2.3. Zu sehen ist der Kartografierungsprozess aus der Vogelperspektive. Die schwarzen Punkte repräsentieren *LaserScan* Daten des *RPLIDAR A2* und die blauen der *Kinect*-Sensoren. Die grauen Linien enden an einer als belegt markierten Zelle und stellen die Karte dar, die sich bei Bewegung des ALFs vervollständigt. Die Linien repräsentieren bekanntes, freies Gebiet. Für die Kartografierung in der Abbildung wurden lediglich Messwerte des *RPLIDAR A2* genutzt.

Des Weiteren veröffentlicht der Knoten das Topic *slam out pose* des Nachrichtentyps *Pose* mit einer Schätzung der aktuellen Roboterposition. In diesem Nachrichtentyp wird die Orientierung des Roboters durch Quaternionen beschrieben. Für die Verwendung des Knotens *Hector Slam* ist es notwendig einen Baum aus statischen Koordinatentransformationen zu erstellen, der die Positionen aller Sensoren relativ zur geschätzten Roboterpose enthält. [34]

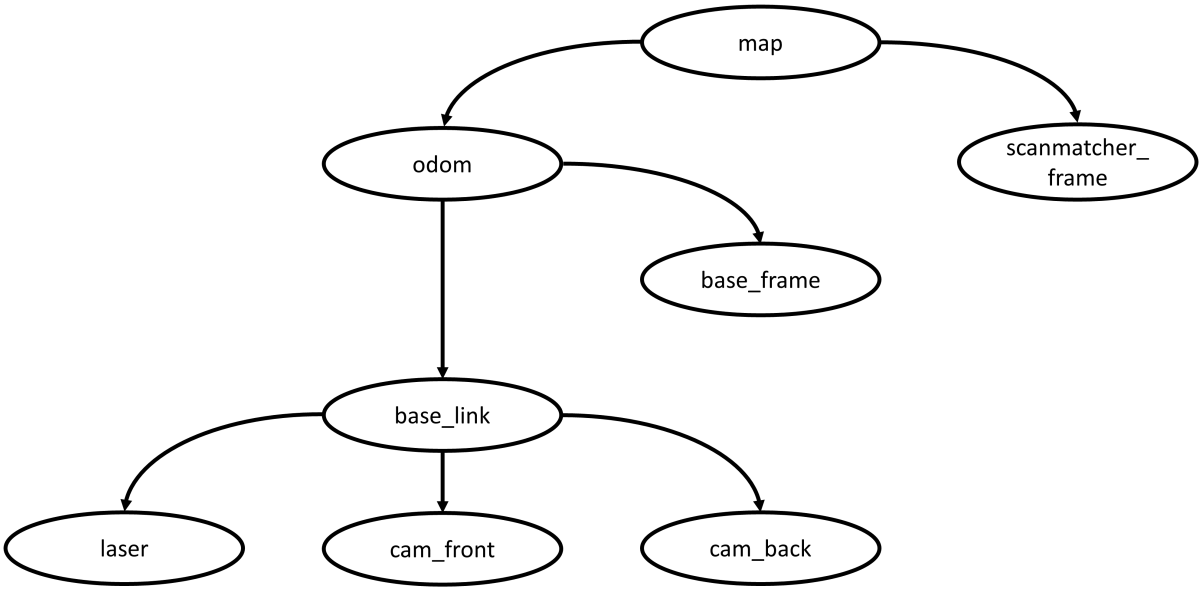


Abbildung 4.19: Darstellung der Baumstruktur der in diesem Projekt verwendeten Koordinatentransformationen. Die Pfeile geben die Transformationsrichtung an. Die Umwandlungen von *map* → *odom* und *map* → *scanmatcher_frame* sind dynamisch. Die Übrigen sind statische Transformationen.

Die in Abbildung 4.19 dargestellten, dynamischen Transformationen *map* → *odom* und *map* → *scanmatcher_frame* werden von *Hector Slam* durchgeführt. Diese sind notwendig, um die Verschiebung der Umgebung im Verhältnis zu dem Roboter darzustellen. Das Koordinatensystem *odom* liegt an derselben Position wie *base_link*, *laser* und der Sensor *RPLIDAR A2*. *Hector Slam* benötigt die Transformationen zwischen diesen Koordinatensystemen für die Kartografierung. Durch die Dokumentation dieses Knotens ist lediglich ein Bezug zwischen der Quelle der Sensorsignale und *base_frame* herzustellen [34]. *Rviz* meldet jedoch Fehler und weist den Benutzer auf die fehlenden Transformationen hin. Die Positionen der *Kinect*-Sensoren werden jeweils durch die Koordinatensysteme *cam_front* und *cam_back* beschrieben. Das *base_frame* beschreibt das Zentrum des Fahrzeugs und ist für eine Planung einer kollisionsfreien Trajektorie notwendig. Der *Static Transform Publisher* veröffentlicht die statischen Koordinatentransformationen in das Topic *tf*. [35]

4.4 Erstellung einer Trajektorie und der daraus folgenden Bewegungsbefehle

Mit den vorhandenen Sensordaten und dem Abbild der Umgebung aus den Kapiteln 4.2 und 4.3 können Trajektorien geplant werden. Die aus dieser Planung entstehenden Bewegungsbefehle werden durch die in Kapitel 4.1 beschriebene Regelung in Bewegungen des Fahrzeugs umgesetzt.

4.4.1 Bestimmung der Bewegungsbefehle

Das Softwarepaket *Navigation* enthält Funktionen für die Umwandlung einer Zielpose in Geschwindigkeitsbefehle. Die Hauptanwendung des Pakets ist der Knoten *Move Base*. Dieser abonniert Observationsquellen, eine *Occupancy Grid*-Karte, Koordinatentransformationen oder Odometrieinformationen und 2D-Zielposen aus dem ROS-Netzwerk. Der Knoten veröffentlicht ein Bewegungsbefehl und *Costmaps*, die in Abschnitt 4.4.2 erläutert werden. [36, 37]

In dieser Bachelorarbeit abonniert *Move Base* die Topics *Scan*, *Scan1* und *Scan2* von den in Kapitel 4.2.2 und 4.2.3 beschriebenen Observierungsquellen. Des Weiteren werden die Informationen über Koordinatentransformationen und 2D-Zielposen aus den Topics *tf* sowie */move_base_simple/goal* abonniert. In der Abbildung A.2 sind alle Abonnements und Veröffentlichungen des Knotens dargestellt.

Der Knoten *Move Base* ist in einzelnen Teilanwendungen aufgeteilt. Der enthaltene *Global Planner* wandelt eine Zielpose in eine Trajektorie um. Durch einen externen Knoten, wie zum Beispiel *Rviz*, kann ein solches Ziel im ROS-Netzwerk veröffentlicht werden. Aus der Trajektorie des *Global Planners* und der geschätzten Position des Roboters erstellt der *Local Planner* das Topic *cmd_vel*. Dieses beinhaltet eine Bewegungsplanung, resultierend aus der aktuellen Positionsschätzung und der geplanten Trajektorie und dient als Grundlage für die Erstellung des Bewegungsziels aus Kapitel 4.1.2. [37, 38]

Der in diesem Projekt verwendete *Teb Local Planner* gilt als Plugin für den in *Move Base* vorhandenen *Local Planner* und arbeitet nach der *Timed Elastic Band* Methode (TEB) [29]. Durch eine prädiktive Regelung, die durch die TEB-Methode beschrieben

ist, lassen sich Vorhersagen über die abzufahrende Trajektorie tätigen, die kontinuierlich optimiert wird [39]. Der *Teb Local Planner* plant die Trajektorie in Abhängigkeit von dem Roboterumriss aus Abbildung 4.20 und der zugrunde liegenden *Costmap*, die im folgenden erkärt wird [29].

4.4.2 Einfluss der Umgebung auf die Planung einer Trajektorie

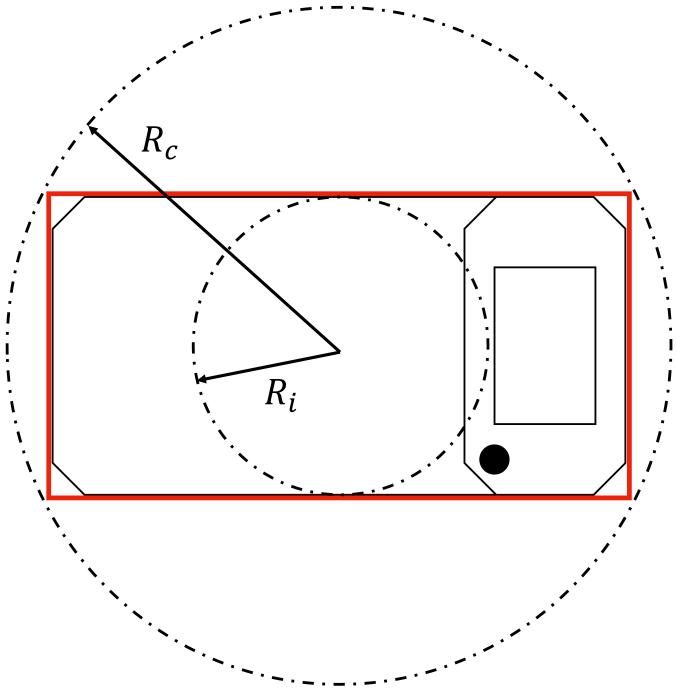


Abbildung 4.20: Darstellung des Fahrzeugs ALF aus der Draufsicht. Als rotes Rechteck ist der Roboterumriss gekennzeichnet. Der *Inscribed Radius* ist mit R_i und der *Circumscribed Radius* mit R_c beschrieben. Die Kreise entsprechen jeweils den Radien R_i und R_c. Der Kreismittelpunkt ist der Ursprung des *base_frame* Koordinatensystems, das in Kapitel 4.3 durch den *Static Transform Publisher* eingeführt wurde

Der Roboterumriss ist als Parameter in den entsprechenden Knoten hinterlegt und wird beim Aufruf von Rviz sichtbar. Abbildung 4.20 stellt den Roboterumriss des ALFs dar und beinhaltet die Radien R_i und R_c, die als *Inscribed-* und *Circumscribed* Radius bezeichnet werden und für die Erstellung der *Costmap* von Bedeutung sind [40]. Das Softwarepaket abonniert, wie in Kapitel 4.2 erörtert, die Sensordaten aus der Umgebung

und verarbeitet diese zu einer sogenannten *Costmap*. Diese Karte dient dazu die Zellen der in Kapitel 2.6.2 beschriebenen *Occupancy Grid*-Karte mit Werten, sogenannten Costs, zwischen 0 und 255 zu gewichten. Die *Costmap* besteht aus mehreren Schichten. Die Informationen über der Positionen der Gegenstände befindet sich in der Hindernisschicht. Sicherheitsradien werden in der Inflationsschicht und die bereits vorhandenen Karten in der statischen Schicht hinterlegt.

In der statischen Schicht wird die oben genannte *Occupancy Grid*-Karte hinterlegt. Diese fungiert als Basis für die Costmap und deren Werteverteilung. In dieser Bachelorarbeit verwendet die statische Schicht das Topic *Map* des in Kapitel 4.3 beschriebenen Knotens *Hector Slam*. Die Hindernisschicht enthält die verarbeiteten Daten aus Kapitel 4.2. Durch Hindernisse werden die entsprechenden Zellen der *Occupancy Grid*-Karte mit dem Wert 254 belegt [40]. Die Inflationsschicht veröffentlicht Informationen über die Gewichtung umliegender Zellen eines eingetragenen Hindernisses auf Basis des Roboterumrisses und den Radien R_i und R_c. Die Gewichtung sinkt mit steigendem Abstand zum *Inscribed Radius* in Form einer e-Funktion und wird mit Gleichung (4.17) berechnet. Der Funktionsverlauf kann von einem Anwender in einer Launchfile durch den Roboterumriss und die ROS-Parameter *Cost Scaling Factor* und *Inflation Radius* beeinflusst werden. In der Gleichung (4.17) ist die Distanz zur besetzten Zelle mit r der *Cost Scaling Factor* der e-Funktion mit f und der *Inscribed Radius* als R_i definiert. [40, 41]

$$cost(r) = \begin{cases} 253, & r \leq R_i \\ e^{-f(r-R_i)} \cdot 253, & r > R_i \end{cases} \tag{4.17}$$

Sämtliche Werte, von der belegten Zelle bis zum *Inscribed Radius* haben den Betrag 253. Erst ab der Distanz $r = R_i$ erfolgt die Werteverteilung nach der e-Funktion.
Die drei Radien haben folgende Bedeutung:

- *Inscribed Radius*: Es gibt definitiv eine Kollision mit dem eingetragenen Gegenstand, wenn das Zentrum des Roboters auf einer Zelle mit dem Wert 253 steht. Das Zentrum des Roboters entspricht den Bezugskoordinatensystem des Roboterumrisses.

- *Circumscribed Radius*: Eine Kollision mit einem Gegenstand ist abhängig von der Orientierung des Roboters.

- *Inflation Radius*: Die Entfernung zum Gegenstand, mit der die Werteverteilung aus Gleichung (4.17) endet. Die Entfernung zum Gegenstand vom *Circumscribed Radius* bis zum *Inflation Radius* dient als Pufferzone. Steht der Roboter in dem Bereich zwischen *Circumscribed* und *Inflation Radius*, so kollidiert er definitiv nicht mit dem in der *Occupancy Grid*-Karte eingetragenen Gegenstand.

Eine Funktionsauswertung mit dem Radius $R_i = 0{,}36$ m aus Abbildung 4.20 und einen Faktor $f = 10$ ergibt den Funktionsverlauf aus Abbildung 4.21.

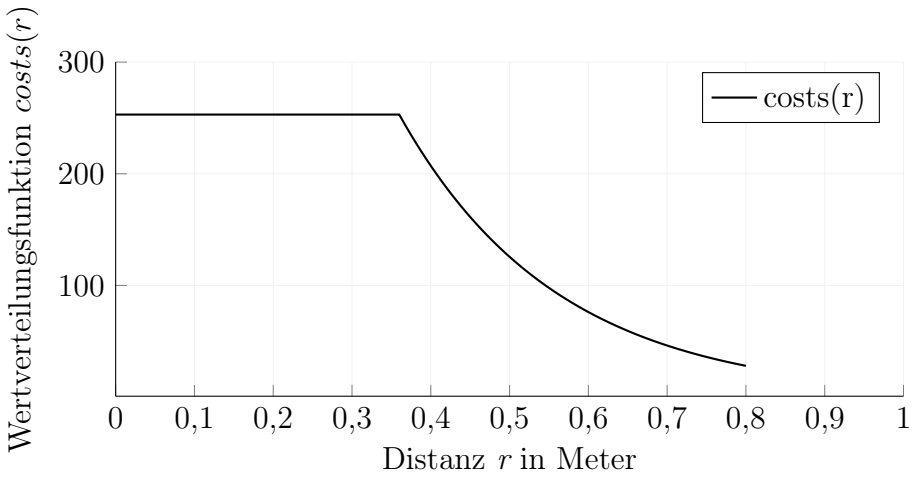


Abbildung 4.21: Verteilung der Werte über die Distanz r. Der *Circumscribed Radius* liegt bei $\approx 0{,}78$ m und der *Inflation Radius* bei 0,8 m. Die Distanz von der besetzten Zelle $r = 0$ m bis zum *Inscribed Radius* $r \approx 0{,}36$ m erhält den Wert 253, die restliche Werteverteilung entspricht dem Funktionsverlauf aus Gleichung (4.17) mit den definierten Parametern.

Abbildung 4.22 zeigt eine vom *Teb Local Planner* geplante Trajektorie in einer *Costmap*. Die Zielpose wurde in *Rviz* manuell in für das System unbekanntes Gebiet gelegt. Die Planung der Trajektorie berücksichtigt die im System bekannten Gegenstände, wie in Abbildung 4.22 zu erkennen ist. In der Abbildung ist die eingegebene Zielpose als grüner Pfeil in unbekanntem Gebiet und die vom *Teb Local Planner* geplante Trajektorie mit roten Pfeilen dargestellt. Der Roboterumriss des Fahrzeugs wird als grünes Rechteck dar-

gestellt und stimmt mit den Dimensionen des Robotermodells beziehungsweise des ALFs überein.

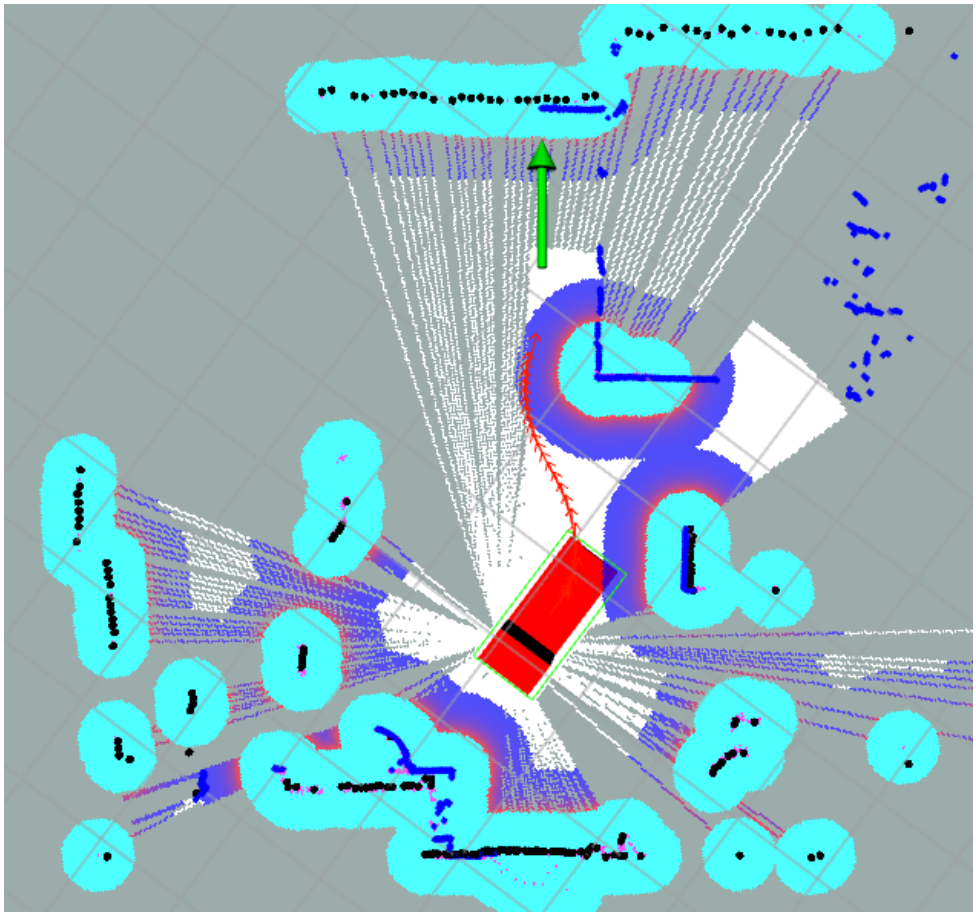


Abbildung 4.22: Darstellung einer *Costmap* mit eingetragener Trajektorie aus der Draufsicht. Anhand der blauen und schwarzen Punkte erkennt man die visualisierten Sensorsignale, wie bereits in Abbildung 4.17 veranschaulicht. In der Darstellung sind um diese Punkte die Gewichtungen der *Costmap* farblich markiert. Als Kette aus roten Pfeilen ist die abzufahrende Trajektorie dargestellt. Oben im Zentrum des Bildes repräsentiert ein grüner Pfeil die Roboterzielpose. Das grüne Viereck in der Mitte zeigt den Roboterumriss. Die vom Roboter ausgehenden, weißen Linien und Flächen demonstriert die Karte aus dem Topic *map*.

Neben der Eingabe einer Zielpose in *Rviz* oder den manuellen Verfahren des Fahrzeugs mit dem Joystick, ermöglicht der Knoten *Explore Lite* eine automatisierte Erkundung der Umgebung. Dabei legt der Knoten die Zielposen in unbekannte Gebiete [42]. Durch das in Kapitel 2.4 beschriebene Lenkungsprinzip ist eine Fahrt ohne Kollisionen gewährleistet.

Der Knoten bewertet unbekanntes Gebiet der *Costmap* als Grenze und versucht diese zu erkunden bis das erreichbare Umfeld vollständig untersucht ist [42].

5 Verifikation

Die zu erreichende Ziele sind in dem Lastenheft im Anhang A.1.3 festgehalten. Die erarbeiteten Ergebnisse dieser Bachelorarbeit werden anhand des folgenden Verifikationsplans geprüft. Im Folgenden ist die Vorgehensweise der Verifikationen mit den entsprechenden Hilfsmitteln geschildert.

Nr./ID	**Nichttechnischer Titel**	**Verifikation der Anforderung**	**Hilfsmittel**
ANF_01	Verwenden der vorhandenen Hardwarestruktur	Die Hardwarestruktur wird untersucht und mit der des Vorgängerprojekts verglichen. Ergebnis: Die Hardwarestruktur gleicht der des Vorgängerprojekts. **Test bestanden.**	Keine

Nr./ID	**Nichttechnischer Titel**	**Verifikation der Anforderung**	**Hilfsmittel**
ANF_02	Drehzahlregelung	Es wird eine gerade Strecke von 5 m abgefahren. Vor Beginn der Fahrt werden im Simulinkmodell die Ist- und Soll-Drehzahlen aufgezeichnet. Anschließend an die Simulation werden die Daten im *Simulation Data Inspector* analysiert. Ergebnis: Nach dem erstmaligen Erreichen des Sollwerts wurde die angegebene Toleranz eingehalten. Dies ist den Abbildungen 5.1, 5.2, 5.3 und 5.4 zu entnehmen. **Test bestanden.**	*Simulink*

Nr./ID	**Nichttechnischer Titel**	**Verifikation der Anforderung**	**Hilfsmittel**
ANF_03	Winkelmessung	Vor Start der Winkelmessung wird das Messwerkzeug auf oder neben dem Fahrzeug positioniert und der angezeigte Wert notiert. Nach einer manuellen Veränderung des Posenwinkels ist die Differenz vom zweiten zum ersten Winkel die zu ermittelnde Posenwinkeldifferenz. Der im Simulinkmodell verarbeitete Winkel wird mit dem gemessenen verglichen. Ergebnis: Der verwendete Kompass zeigte einen Wert von 42°, wobei der erfasste Winkel 43° betrug. **Test bestanden.**	Kompass Geodreieck *Simulink*

Nr./ID	**Nichttechnischer Titel**	**Verifikation der Anforderung**	**Hilfsmittel**
ANF_04	Positionserkennung	Der ALF wird manuell um circa 2 m verschoben. Die X- und Y-Komponente der Verschiebung wird gemessen und mit den Daten des Topics *slam out pose* verglichen. Ergebnis: Die X- und Y-Komponente der Verschiebung aus dem Topic betrug jeweils 1,4 m und 2,1 m. Die tatsächliche Verschiebung ergab nach Messung 1,45 m in X-Richtung und 2,2 m in Y-Richtung. **Test bestanden.**	Bandmaß Geodreieck

Nr./ID	**Nichttechnischer Titel**	**Verifikation der Anforderung**	**Hilfsmittel**
ANF_05	Lageregelung	Vor Start der Verifikation muss der im Simulinkmodell verarbeitete Winkelwert aufgezeichnet werden. Anschließend muss das Fahrzeug durch eine Posenvorgabe des Benutzers in *Rviz* bewegt werden. Im „Simulation Data Inspector“ wird nun der aufgezeichnete Winkel mit dem der Vorgabe verglichen. Ergebnis: Die X- und Y-Komponente der Z aus dem Topic betrug jeweils 1,4 m und 2,1 m. Die tatsächliche Verschiebung ergab nach Messung 1,45 m in X-Richtung und 2,2 m in Y-Richtung. **Test bestanden.**	*Simulink* *Rviz* Joystick

Nr./ID	Nichttechnischer Titel	Verifikation der Anforderung	Hilfsmittel
ANF_06	Maximale Überschwingweite	Es wird eine gerade Strecke von 5 m abgefahren. Vor Beginn der Fahrt müssen im Simulinkmodell die Ist- und Soll-Drehzahlen aufgezeichnet werden. Anschließend an die Simulation werden die Daten im *Simulation Data Inspector* analysiert. Ergebnis: Die Analyse der aufgenommenen Daten aus den Abbildungen 5.1, 5.2, 5.3 und 5.4 ergab ein maximales Überschwingen von 11%. **Test bestanden.**	*Simulink*
ANF_07	Moduswechsel über Joystick	Durch das Betätigen der Taste „B“ am Joystick wird der Fahrmodus gewechselt. Ergebnis: Durch Drücken der entsprechenden Taste wurde der Betriebsmodus umgeschaltet. **Test bestanden.**	*Simulink*

Nr./ID	Nichttechnischer Titel	Verifikation der Anforderung	Hilfsmittel
ANF_08	Anfahren einer vom Benutzer vorgegebenen Zielpose	Zur Durchführung sind folgende Arbeitsschritte durchzuführen: 1. ALF einschalten (Rechner hochfahren und Wahlschlüsselschalter auf „Hand“-Modus stellen) 2. RALFNavi1.launch.xml starten 3. Das in dieser Bachelorarbeit enthaltene Simulinkprogramm starten 4. Resetknopf drücken 5. RALFNavi2.launch.xml starten 6. Am Joystick die Taste „B“ drücken 7. In *Rviz* die Zielpose auf der Karte markieren 8. Am Joystick die Tasten „left trigger“ und „right trigger“ drücken Ergebnis: Die X- und Y-Koordinaten der eingegebenen Zielpose betrugen relativ zu dem globalen Koordinatensystem *map* 0,402 m und 0,954 m mit dem Posenwinkel 45,15°. Die Ist-Pose hatte die X-Koordinate 0,49 m und die Y-Koordinate 0,64 m mit dem Posenwinkel 53,5° **Test bestanden.**	*Rviz* *Simulink* Joystick

Nr./ID	**Nichttechnischer Titel**	**Verifikation der Anforderung**	**Hilfsmittel**
ANF_09	Kartografieren der Umgebung mit Bewegungsvorgabe durch den Benutzer	Zur Durchführung sind folgende Arbeitsschritte durchzuführen: 1. ALF einschalten (Rechner hochfahren und Wahlschlüsselschalter auf „Hand“-Modus stellen) 2. RALFNavi1.launch.xml starten 3. Das in dieser Bachelorarbeit enthaltene Simulinkprogramm starten 4. Resetknopf drücken 5. RALFNavi2.launch.xml starten 6. Bewegungsvorgabe durch den Benutzer mithilfe des Joysticks oder einer Zielvorgabe in *Rviz*. 7. Am Joystick die Tasten „left trigger“ und „right trigger“ drücken Ergebnis: Das Fahrzeug wurde durch die entsprechende Eingabe am Joystick gesteuert. Dabei kartografiert es seine Umgebung. **Test bestanden.**	*Rviz* *Simulink* Joystick

Nr./ID	**Nichttechnischer Titel**	**Verifikation der Anforderung**	**Hilfsmittel**
ANF_10	Kartografieren der Umgebung ohne Bewegungsvorgabe durch den Benutzer	Zur Durchführung sind folgende Arbeitsschritte durchzuführen: 1. ALF einschalten (Rechner hochfahren und Wahlschlüsselschalter auf „Hand“-Modus stellen) 2. RALFNavi1.launch.xml starten 3. Das in dieser Bachelorarbeit enthaltene Simulinkprogramm starten 4. Resetknopf drücken 5. RALFNavi2.launch.xml starten 6. Explore_costmap.launch Starten. 7. Am Joystick die Tasten „left trigger“ und „right trigger“ drücken Ergebnis: Das Fahrzeug kartografiert die Umgebung nicht ohne eine Bewegungsvorgabe durch den Benutzer. **Test nicht bestanden.**	*Rviz* *Simulink* Joystick

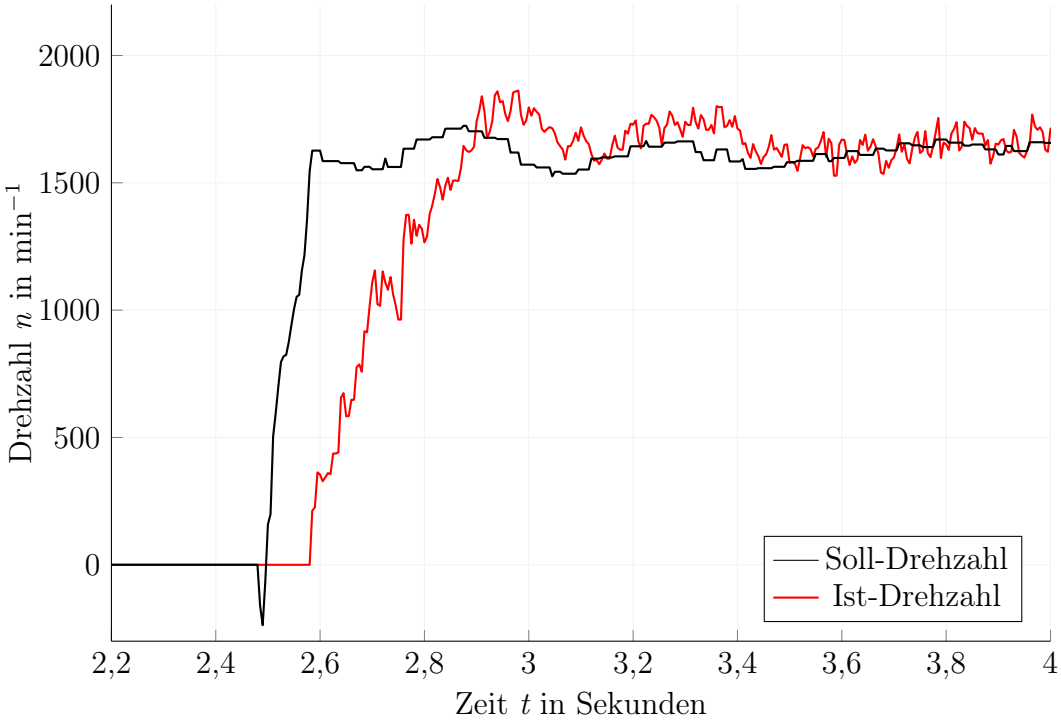


Abbildung 5.1: Messreihe für den Motor hinten rechts.

Abbildung 5.2: Messreihe für den Motor hinten links.

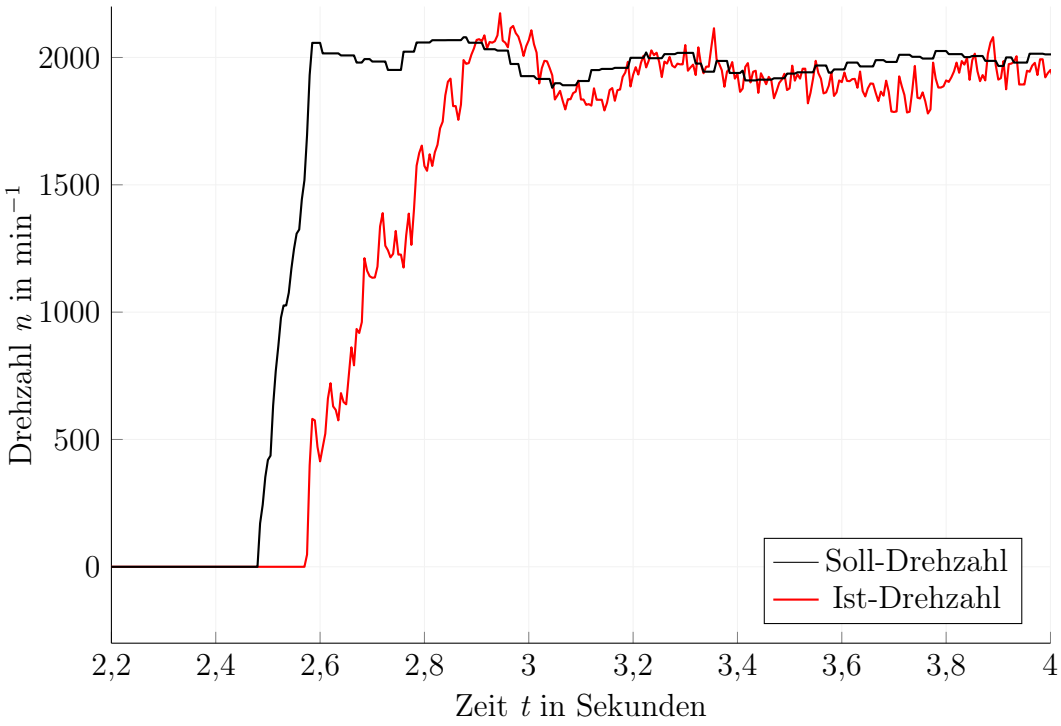


Abbildung 5.3: Messreihe für den Motor vorne rechts.

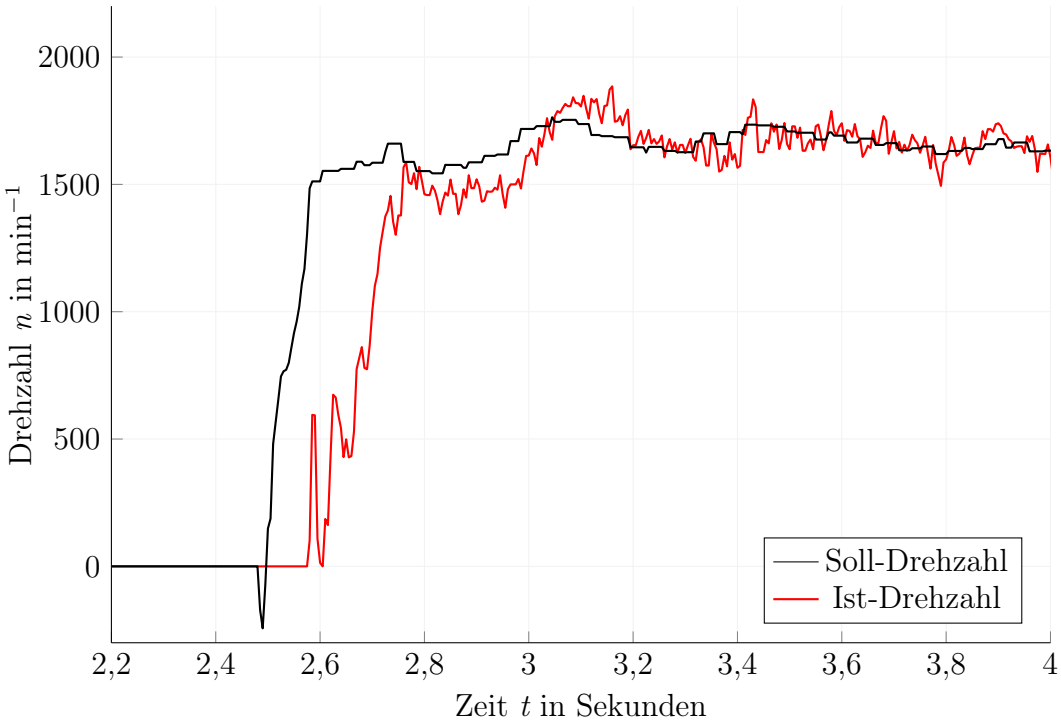


Abbildung 5.4: Messreihe für den Motor vorne links.

6 Zusammenfassung und Ausblick

Im Rahmen dieser Bachelorarbeit wurde eine Schlupfregelung und eine Beispielanwendung für autonomes Fahren konzipiert, integriert und verifiziert. Für die Umsetzung der Ziele bezüglich der Drehzahlregelung wurde eine Simulation des Reglers vorgenommen. Wie in den Abbildungen 5.1, 5.2, 5.3 und 5.4 zu sehen ist, erfüllte dieser die erhobene Anforderung ANF_02. Der Regler zeigte bei der Verifikation maximal 11% Überschwingen für das Erreichen der Soll-Drehzahlen. In Abbildung 5.1 ist das beschriebene Verhalten gezeigt. Die integrierte, zur Drehzahlregelung übergeordnete Lageregelung verhindert die in Kapitel 1 erwähnten Rotationen des Fahrzeugs. Diese werden mithilfe eines aus Umgebungsdaten generierten Posenwinkels ausgeregelt. Eine Modusumschaltung wurde integriert, um die Betriebsmodi per Knopfdruck wechseln zu können. Das ALF kann durch manuelle Eingaben am Joystick oder durch autonome Fahrfunktionen gesteuert werden. Die vorhandene Sensorik wird zur Erfassung der Umgebung verwendet. Dabei kartografiert das ALF die Umgebung durch den SLAM-Algorithmus. Die Anforderung ANF_10 konnte nicht erfüllt werden, da das Fahrzeug ohne Bewegungsvorgabe durch den Benutzer den Kartografierungsvorgang nicht selbstständig fortsetzt. Zu Beginn der Kartografierung ist das Fahrzeug umgeben von kleinen, unbekannten Gebieten. Diese resultieren aus der Auflösung des *RPLIDAR A2* und schränken die Manövrierfähigkeit des Roboters ein. Erfolgt manuell eine Bewegung, wird die Umgebung als bekannt markiert und das ALF kartografiert selbstständig die Umgebung.

Die gegebenen Hardwareeigenschaften des Roboters ermöglichen die Einbindung der Schlupfregelung und der Fähigkeit des autonomen Fahrens, wobei die Rechenleistung des installierten Rechners bei vollumfänglichen Betrieb nahezu ausgelastet ist. Diese Problematik hat bei einer Erweiterung der Software beziehungsweise bei höherer Auslastung Auswirkungen auf den in *Simulink* verbauten Regler. Ein mögliches Folgeprojekt kann die Implementierung des Reglers in eine andere Hardwarestruktur beinhalten.

Der Lidar-Sensor erfasst die Umgebung und reproduziert mit *Rviz* ein Abbild dieser. Aufgrund der in Kapitel 4.2.3 beschriebenen Bauhöhe des Sensors und der damit verbundenen Komplikationen ist eine Erweiterung der Sensorik zur Erfassung der Umgebung zu empfehlen. Erfasst der Roboter seine Umgebung vollständig, ist eine omnidirektionale Fahrweise im automatisierten Betrieb möglich.

Aufgrund des in Kapitel 4.4.2 beschriebenen *Costmap*, wird der Roboter wegen seiner Größe und Form in der Bewegungsplanung eingeschränkt.

Der Posenwinkel wird mithilfe von *Hector Slam* erfasst. Während der Entwicklungsphase der Schlupfregelung zeigte eine Auswertung des auf dem *Raspberry Pi* montierten Gyrosensor mit Signalrauschen überlagerte Daten. Durch eine Integration der gemessenen Winkelgeschwindigkeit mit dem überlagerten Rauschen entsteht mit der Zeit eine Abweichung von realem und aus der Winkelgeschwindigkeit integrierten Winkel. Die Untersuchung und Einbindung eines geeigneten Filters kann als Folgeprojekt behandelt werden.

Aktuell werden Personen, die sich in der zu kartografierenden Umgebung befinden, auf der zu erstellenden Karte als Objekt markiert. Die Zellen der Belegtheitskarte werden allerdings dauerhaft als belegt markiert, auch wenn sich die Person durch die Umgebung bewegt. Dies hat Auswirkungen auf die Trajektorieplanung, da mit der Zeit keine freien Gebiete mehr vorhanden sind. Eine Untersuchung und Implementierung eines sogenannten *Social Layers* in die *Costmap* könnte das Problem beheben. Wenn sich das Fahrzeug in unbekanntem Gebiet befindet und eine Zielpose autonom anfahren soll, werden Fahrmanöver anfänglich nur unregelmäßig durchgeführt. Das Fahrzeug ist für eine Fahrt ohne Kollisionen konzipiert und verfährt nur in bereits bekannte Gebiete, womit die eben erwähnte Problematik begründet wird. Das Hinzufügen einer statischen Karte, die den Operationsbereich des Fahrzeugs enthält, in das bereits vorhandene System würde das Ansprechverhalten des ALFs optimieren.

Quellenverzeichnis

[1] Fraunhofer-Institut für Materialfluss und Logistik IML. *Autonomes Fahren in der Logistik.*

[2] Lena Anzenhofer. *Amazon rüstet seine Lager mit immer mehr Robotern aus — und lässt sich das Hunderte Millionen Dollar kosten.* Zugriff: 15. Feb. 2019. https://www.businessinsider.de/amazon-ruestet-seine-lager-mit-immer-mehr-robotern-aus-und-laesst-sich-das-hunderte-millionen-dollar-kosten-2019-1.

[3] Deutsche Presse-Agentur. *Studie: Autonomes Fahren kann Logistik-Kosten halbieren.* Zugriff: 15. Feb. 2019. https://www.zeit.de/news/2018-09/13/studie-autonomes-fahren-kann-logistik-kosten-halbieren-180913-99-937711.

[4] Dominik Eickmann und Dennis Hotze. *Entwicklung und Verifikation eines autonomen Logistik-Fahrzeugs.* Masterthesis. Hochschule Bochum - Bochum University of Applied Sciences, Feb. 2018.

[5] FabianSacilotto. *ROS.org.* Zugriff: 31. Jan. 2019. http://wiki.ros.org/de.

[6] TullyFoote. *ROS.org Concepts.* Zugriff: 31. Jan. 2019. http://wiki.ros.org/de/ROS/Concepts.

[7] BradMiller. *ROS.org Parameter Server.* Zugriff: 31. Jan. 2019. http://wiki.ros.org/Parameter%20Server.

[8] YanqingWu. *ROS.org Master.* Zugriff: 31. Jan. 2019. http://wiki.ros.org/Master.

[9] Heinz Unbehauen. *Regelungstechnik I: Klassische Verfahren zur Analyse und Synthese linearer kontinuierlicher Regelsysteme.* 9. Auflage.

[10] Jan Lunze. *Regelungstechnik 1: Systemtheoretische Grundlagen, Analyse und Entwurf einschleifiger Regelungen.* 10. Auflage.

[11] Serge Zacher und Manfred Reuter. *Regelungstechnik für Ingenieure: Analyse, Simulation und Entwurf von Regelkreisen*. 15. Auflage. Springer Vieweg, 2017.

[12] *NM203AR 203mm Heavy duty Mecanum Wheel ,Right*. Zugriff: 31. Jan. 2019. http://robotchassisparts.com/wp-content/uploads/2016/09/NM203A-layout.pdf.

[13] *Mecanum wheels (Ilon wheel)*. Zugriff: 31. Jan. 2019. http://robotchassisparts.com/wp-content/uploads/2016/09/Mecanum-wheel1-16.pdf.

[14] Li Xie, Christian Scheifele, Weiliang Xu und Karl A. Stol. *Heavy-duty omnidirectional Mecanum-wheeled robot for autonomous navigation: System development and simulation realization*. 2015 IEEE International Conference on Mechatronics (ICM), März 2015.

[15] Joachim Hertzberg, Kai Lingemann und Andreas Nüchter. *Mobile Roboter*. Springer Vieweg, 2011.

[16] Peter Pfeffer und Manfred Harrer. *Lenkungshandbuch: Lenksysteme, Lenkgefühl, Fahrdynamik von Kraftfahrzeugen*. 2. Auflage. Springer Vieweg, 2013.

[17] Ralph Pütz und Ton Serné. *Rennwagentechnik - Praxislehrgang Fahrdynamik: Eine praktische Anleitung für Amateure und Profis*. Springer Vieweg, 2017.

[18] Hietzinger Friedhof. *llustration of Ackermann steering geometry*. Zugriff: 31. Jan. 2019. https://upload.wikimedia.org/wikipedia/commons/d/dd/Ackermann_radius_M.svg.

[19] Ltd. Shanghai Slamtec Co. *RPLIDAR A2*. 2016. http://www.slamtec.com/en/Lidar/A2.

[20] Evan-Amos. *Xbox-One-Kinect.jpg*. Zugriff: 31. Jan. 2019. https://upload.wikimedia.org/wikipedia/commons/f/f6/Xbox-One-Kinect.jpg.

[21] Bruno Siciliano und Oussama Khatib. *Handbook of Robotics*. Springer, 2007.

[22] Matthias Haun. *Handbuch Robotik: Programmieren und Einsatz intelligenter Roboter*. 2. Auflage. Springer Vieweg, Dez. 2006.

[23] Leonie Sautter. *Graphbasiertes SLAM mit integrierter Kalibrierung für mobile Roboter*. Diplomarbeit. Karlsruher Institut für Technologie, 2015.

[24] Kai Arrasm Maren Bennewitz Wolfram Burgard, Cyrill Stachniss. *Introduction to Mobile Robotics*. Zugriff: 31. Jan. 2019. http://ais.informatik.uni-freiburg.de/teaching/ss12/robotics/slides/12-slam.pdf.

[25] Kristof Schroeter. *Probabilistische Methoden für die Roboter-Navigation am Beispiel eines autonomen Shopping-Assistenten*. Doktorarbeit. Technische Universität Ilmenau, 2009.

[26] Herbet Süße und Erik Rodner. *Bildverarbeitung und Objekterkennung: Computer Vision in Industrie und Medizin*. Springer Vieweg, 2014.

[27] Prof. Dr. Klaus Wüst. *Grundlagen der Robotik*. Zugriff: 15. Feb. 2019. `https://homepages.thm.de/~hg6458/Robotik/Robotik.pdf`.

[28] Wolfgang Schneider und Berthold Heinrich. *Praktische Regelungstechnik: Effektiv lernen durch Beispiele*. Springer Vieweg, 2016.

[29] ChristophRoesmanns. *ROS.org teb local planner*. Zugriff: 12. Feb. 2019. `http://wiki.ros.org/teb_local_planner`.

[30] WilliamWoodall. *ROS.org rviz*. Zugriff: 31. Jan. 2019. `http://wiki.ros.org/rviz`.

[31] KintZhao. *ROS.org rplidar*. Zugriff: 31. Jan. 2019. `http://wiki.ros.org/rplidar`.

[32] Lingzhu Xiang. *Libfreenect2 Troubleshooting*. Zugriff: 31. Jan. 2019. `https://github.com/OpenKinect/libfreenect2/wiki/Troubleshooting`.

[33] NickLamprianidis. *ROS.org depthimage to laserscan*. Zugriff: 31. Jan. 2019. `http://wiki.ros.org/depthimage_to_laserscan`.

[34] StefanKohlbrecher. *ROS.org hector slam*. Zugriff: 31. Jan. 2019. `http://wiki.ros.org/hector_slam`.

[35] jarvisschultz. *ROS.org tf*. Zugriff: 31. Jan. 2019. `http://wiki.ros.org/tf`.

[36] MatthiasGruhler. *ROS.org navigation*. Zugriff: 31. Jan. 2019. `http://wiki.ros.org/navigation`.

[37] NicolasVaras. *ROS.org move base*. Zugriff: 31. Jan. 2019. `http://wiki.ros.org/move_base`.

[38] derzu. *ROS.org Robots TIAGo Tutorials motions cmd vel*. Zugriff: 31. Jan. 2019. `http://wiki.ros.org/Robots/TIAGo/Tutorials/motions/cmd_vel`.

[39] C. Rösmann, F. Hoffmann und T. Bertram. *Timed-Elastic-Bands for time-optimal point-to-point nonlinear model predictive control*. Juli 2015. 2015 European Control Conference (ECC).

[40] NickLamprianidis. *ROS.org costmap 2d*. Zugriff: 04. Feb. 2019. `http://wiki.ros.org/costmap2d`.

[41] DavidLu. *ROS.org costmap 2d hydro inflation*. Zugriff: 04. Feb. 2019. `http://wiki.ros.org/costmap_2d/hydro/inflation`.

[42] hrnr. *ROS.org Explore Lite*. Zugriff: 04. Feb. 2019. `http://wiki.ros.org/explore_lite`.

A Anhang

A.1 Abbildungen

Abbildung A.1: Darstellung des ALFs.

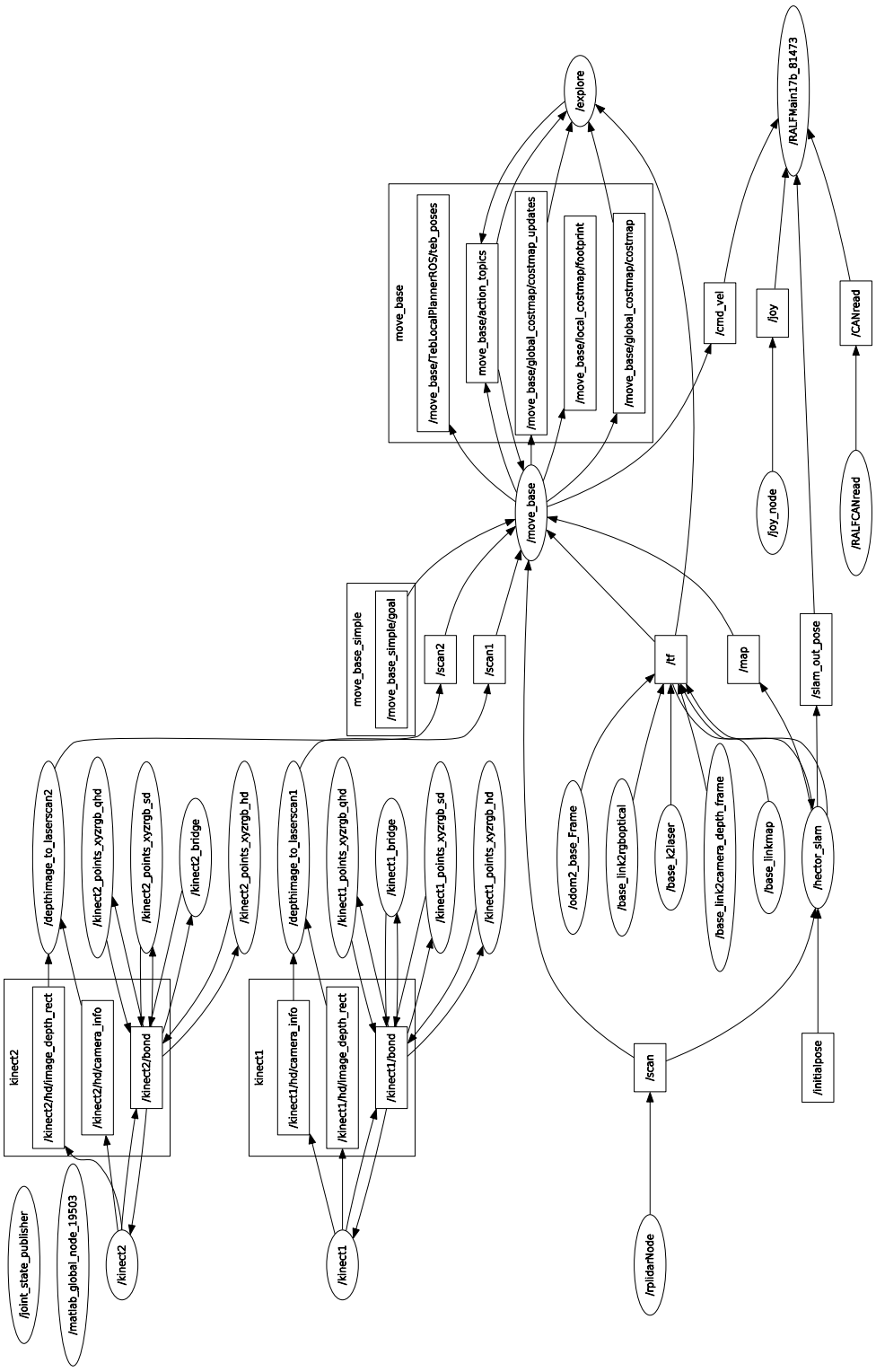


Abbildung A.2: Darstellung des im Projekt erstellten ROS-Netzwerks. Knoten sind in dieser Abbildung oval und Topics viereckig dargestellt. Die Pfeile symbolisieren den Datenfluss der veröffentlichten und abonnierten Nachrichten. Die in einem Viereck zusammengefassten Topics repräsentieren gruppierte Topics.

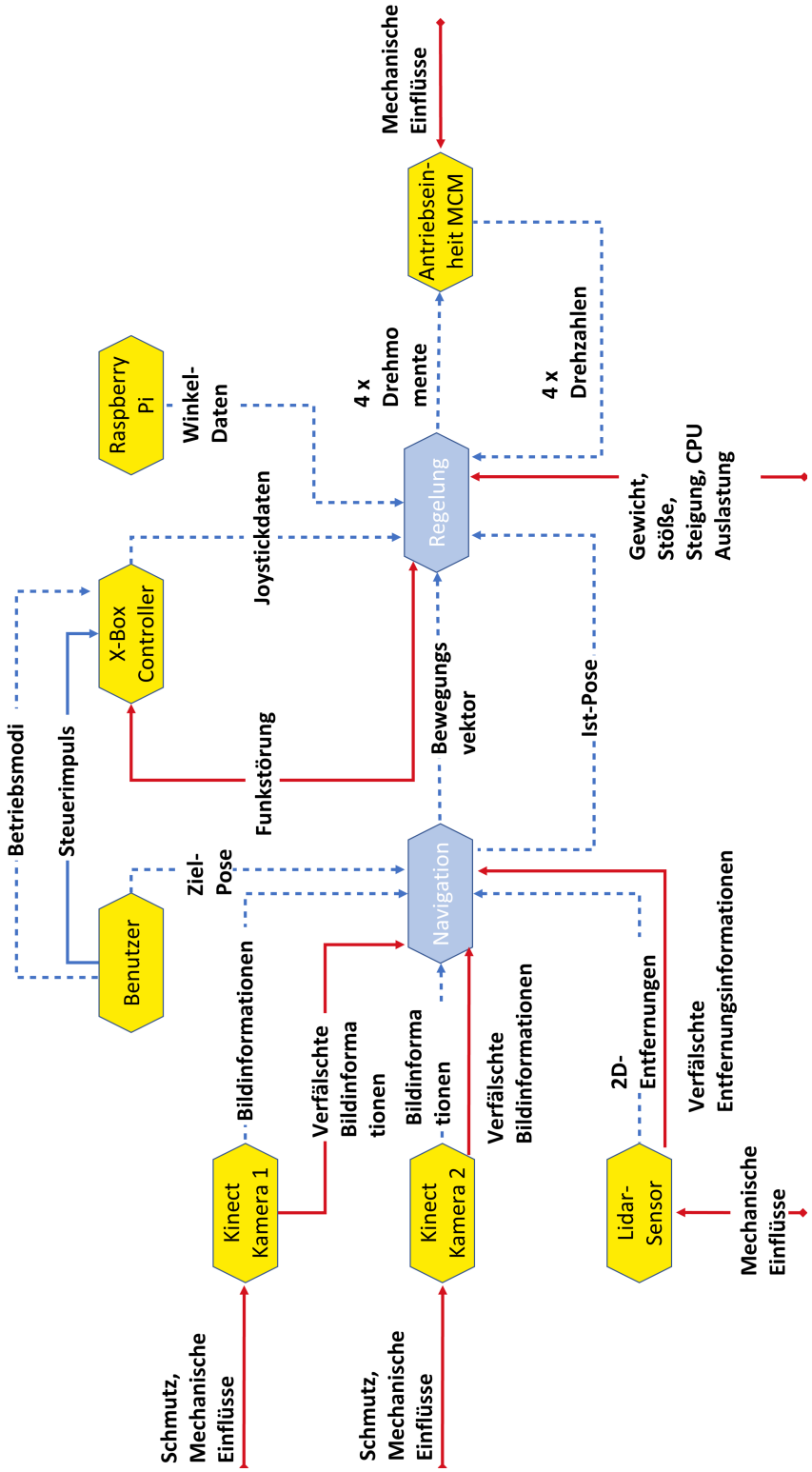


Abbildung A.3: Darstellung des in diesem Projekt entwickelten Umfeldmodells. Der Informationsfluss ist in dieser Abbildung mit blauen, gestrichelten Pfeilen abgebildet. Rote Pfeile symbolisieren Störeinflüsse, die auf das entwickelte System einwirken.

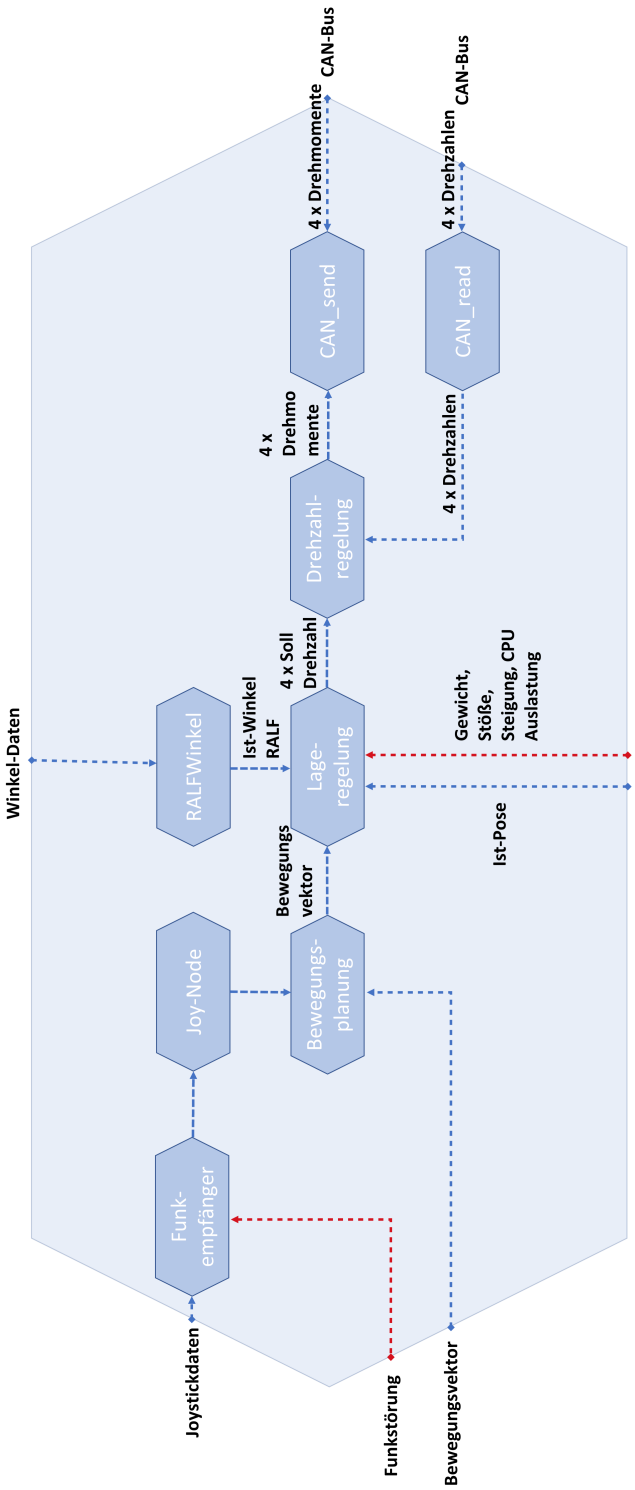


Abbildung A.4: Darstellung einer Wirkstruktur der in diesem Projekt konzipierten Schlupfregelung. Der Informationsfluss ist in dieser Abbildung mit blauen, gestrichelten Pfeilen abgebildet. Die roten, gestrichelten Pfeile zeigen Störeinflüsse, die auf die entwickelte Regelung einwirken.

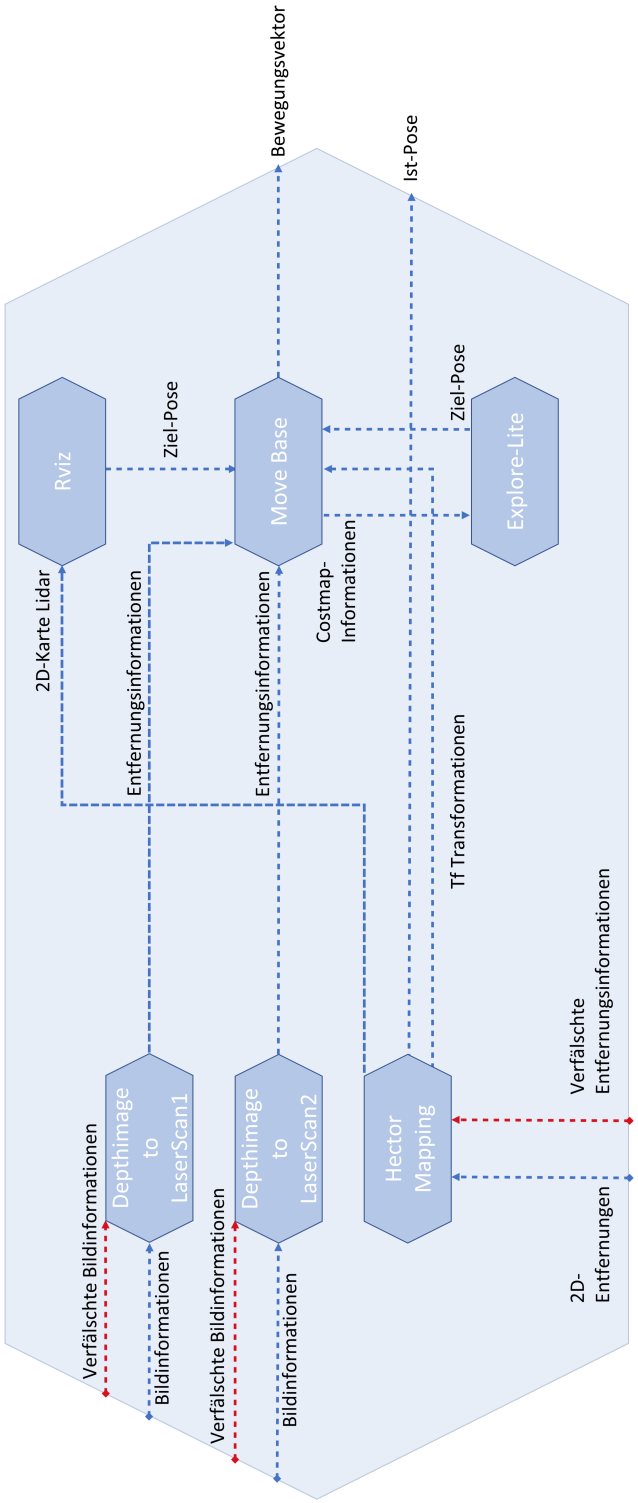


Abbildung A.5: In der Darstellung ist das Konzept der Navigation und Kartografierung gezeigt. Die Pfeile symbolisieren den Informationsfluss, wobei die roten Störeinflüsse darstellen.

A.2 Inhalt Datenträger

1 Bachelorarbeit

2 Datenblätter

3 Lastenheft

4 Software

- Robot Operating System
- Simulinkmodell