MODELLBASIERTE REGELUNG DEZENTRALER SYSTEME IN UNBEKANNTEN UMGEBUNGEN

Dissertation

zur Erlangung des Grades eines

D o k t o r s d e r I n g e n i e u r w i s s e n s c h a f t e n

der Technische Universität Dortmund
an der Fakultät für Informatik

von

Alexander Puzicha

Dortmund

2023

Datum der mündlichen Prüfung: 10.11.2023
Dekan: Prof. Dr.-Ing. Gernot A. Fink
Gutachter: Univ.-Prof. Dr. rer. nat. Peter Buchholz, Univ.-Prof. Dr. Jian-Jia Chen

„Deine Lebensgrundsätze werden stets ihre Gültigkeit für dich behalten, solange dir die ihnen entsprechenden Grundbegriffe nicht abhanden gekommen sind. Das aber kannst du verhindern, indem du dieselben immer wieder zu neuem Leben in dir anfachst und über das, was notwendig ist, nicht aufhörst nachzudenken — : wobei dich nichts zu stören vermag, weil alles, was zu deinem Gedankenleben von außen hinzutritt, als solches keinen Einfluß auf dasselbe hat. Halte dich also nur so, daß es dir äußerlich bleibt! Hast du aber deine Lebenshaltung einmal eingebüßt: Du kannst sie wieder gewinnen. Sieh die Dinge wieder gerade so an, wie du sie angesehen hattest! Darin besteht alles Wiederaufleben.“
Marc Aurel, Selbstbetrachtungen, 7.2

Danksagung

Die vorliegende Arbeit entstand während meiner Tätigkeit im Graduiertenkolleg 2193 am Lehrstuhl für praktische Informatik der Fakultät Informatik der Technischen Universität Dortmund. Ich möchte mich bei Univ.-Prof. Dr. rer. nat. Peter Buchholz bedanken, dass er meine wissenschaftliche Arbeit ermöglicht und großzügig unterstützt hat. Ich bin sehr dankbar für sein wertvolles Feedback, sein Vertrauen und seine anregenden Vorschläge. Er hat mir stets die Freiheit gegeben, meine eigenen wissenschaftlichen Erkenntnisse zu entwickeln und sie auf lokalen und internationalen Konferenzen zu präsentieren.
Ich möchte auch Univ.-Prof. Dr. Jian-Jia Chen danken, der sich bereit erklärt hat, meine Arbeit als Zweitprüfer zu begutachten. Ich weiß seine Unterstützung und sein Fachwissen vor allem im Bereich der eingebetteten Systeme und der zeitkritischen, hardwarenahen Programmierung sehr zu schätzen.
Darüber hinaus möchte ich mich bei allen Mitarbeitern des Lehrstuhls für praktische Informatik für die angenehme und produktive Arbeitsatmosphäre bedanken. Alexander Frank bin ich sehr dankbar für die Zeit, die wir damit verbracht haben, viele mathematische, informatische und weitere theoretische Themen zu diskutieren und für die Reisen, die wir gemeinsam unternommen haben. Ein besonderer Dank geht auch an Christin Schumacher für den Austausch in vielen Themenbereichen. Alina Timmermann danke ich für das Versprühen von Motivation und Freude und Andreas Blume für seine passenden trocknen Witze dazu. Für die stets freundliche Unterstützung bei administrativen und technischen Fragen möchte ich mich bei Nicole Kusmierz, Ute Winter und Falko Bause bedanken.
Einen anderen Blickwinkel auf die Anfertigung von Abschlussarbeiten habe ich Herrn Priv.-Doz. Dr. Frank Weichert des Lehrstuhls für grafische Systeme durch die gemeinsame Betreuung einiger Arbeiten zu verdanken.
Weiterhin hat mich Herr Arno Schmidt weit über die Kooperation mit der Rheinmetall Electronics GmbH hinaus mit technischen Fachgesprächen und möglichen Anwendungsfällen inspiriert stets weiter zu forschen.
Ferner möchte ich mich für die finanzielle Unterstützung durch die Deutsche Forschungsgemeinschaft (DFG) im Rahmen des Graduiertenkollegs 2193 bedanken. Worte der Dankbarkeit sind meinen Freunden gewidmet, die mir geholfen haben, Ablenkung und Entspannung zu finden. Schließlich bin ich meiner Familie sehr dankbar für ihre kontinuierliche, bedingungslose Unterstützung und Ermutigung nie aufzugeben während meines gesamten Lebens.

Dortmund, 10.11.2023 Alexander Puzicha

Kurzfassung

Diese Arbeit beschreibt ein Konzept eines Autonomiesystems für autonome Roboterschwärme in Katastrophengebieten. Die Realisierung des Konzeptes und die Integration dieses in einen Prototypen wird ebenfalls dargelegt.
Autonome Roboterschwärme können künftig für eine Vielzahl an Katastropheneinsätzen eingesetzt werden. Typische Aufgaben sind die Erkundung eines Gebietes, die Eskortierung von Konvois, der Objekttransport oder der Aufbau von Netzwerken, wenn die Infrastruktur zusammenbricht. Die autonome Regelung der Roboter ist eine anspruchsvolle Aufgabe, vor allem unter schwierigen Umweltbedingungen, die die Kommunikation einschränken. Folglich muss die Regelungssoftware der Roboterschwärme sorgfältig getestet werden, bevor die Roboter in Katastrophengebieten eingesetzt werden. Da Feldtests aufwändig, teuer und teilweise nicht möglich sind, ergeben Simulationen die einzige brauchbare Alternative. Der Test der Software mit Echtzeitanforderungen muss jedoch in einer Echtzeitumgebung durchgeführt werden, wodurch strenge Anforderungen an die Simulationssoftware gestellt werden. Die Regelung muss Teil des Simulators sein, die Dynamik der Roboter muss realistisch simuliert werden, die Umgebungsbedingungen beschrieben werden und die mobile Kommunikation muss modelliert werden, einschließlich der begrenzten Kommunikationsreichweiten und Paketverluste. Dies sollte benutzerfreundlich und effizient implementiert sein, um die Simulation von größeren Schwärmen, wie sie in der Praxis eingesetzt werden, zu ermöglichen.
Da zu diesem Zeitpunkt kein System, mit den geforderten Funktionen bekannt ist, werden alle notwendigen Hard- und Softwarekomponenten sowie Protokolle eigenständig entwickelt oder integriert. Dazu wird ein stützstellenbasierter modellprädiktiver Regler für die mobile Robotik adaptiert und weiterentwickelt. Dieser erzielt eine sehr hohe, empirisch nachweisbare Regelgüte im Vergleich zu einer Auswahl an Optimierungsalgorithmen in umfangreichen Tests. Dabei wird eine Zeitschranke von 300 ms für einen 30 s langen Planungshorizont gewahrt. Dies wird durch eine exponentielle Zerlegung des Horizontes erreicht. Die theoretische Stabilität ist jedoch aufgrund der Kostenfunktionsfreiheit und der exponentiellen Verteilung nicht beweisbar.
Um mit Hilfe eines Schwarms ein Netzwerk für Rettungskräfte aufzubauen und um die Agenten in weitläufigem, infrastrukturlosem Gebiet von bis zu 25 km^2 zu betreiben, werden Netzwerkqualitätsprädiktionen und kostengünstige Knotenpunktnetzwerke mit großer Reichweite und geringer Übertragungsrate vorgestellt. Die geringe Übertragungsrate erzwingt die Entwicklung einer anwendungsspezifischen Datenkompression innerhalb des Autonomiesystems. Dieser besteht aus C++ Bibliotheken mit baumartiger Abhängigkeitsstruktur, die vollständig plattform- und prozessorarchitekturunabhängig sowie modular nutzbar sind. Das System ermöglicht den Betrieb autonomer Bodenfahrzeuge sowie die Simulation großer Schwärme mit bis zu 70 Agenten. Dabei können diese rein virtuell, aus emulierter Hardware oder auch aus

realen Komponenten aufgebaut sein. Die Realzeitsimulation kann gemischt zentral und dezentral berechnet werden und unterstützt eine Verschmelzung von Simulation und Realdaten bis hin zur Emulation.
Der Entwurf des Systems ermöglicht eine einfache Integration in verschiedene Visualisierungssysteme wie Cocos2d-X, CryEngine und Unity. Somit lassen sich sowohl realzeitfähige Anwendungen ohne Oberfläche oder mit einfacher zweidimensionaler Darstellung als auch fotorealistische Anwendungen in der virtuellen Realität erstellen. Der Umfang von über 16 Missionen, den die Agenten bieten, basiert auf einer erweiterten Potenzialfeldtechnik. Aufgrund der durch den Optimierer gebotenen Freiheit, existieren keine Beschränkungen bezüglich der Stetigkeit oder Differenzierbarkeit der zugrundeliegenden Funktionen. Weiterhin wird durch die Freiheit der Akkumulation, Faltung und Verkettung der Funktionen ein modularer Ansatz geboten. Mit diesem lassen sich auch Verhaltensbaumverfahren oder separierte lokale und globale Planer abbilden. Dabei reichen die Missionen von klassischer Wegpunktnavigation über Überwachungsaufgaben bis zu komplexen, dezentralen Schwarmtransporten und Rendezvous. Auch die Störsenderdetektion sowie ein Energiemanagement sind implementiert.
Zur Validierung des gesamten Softwaresystems wird ein realer Prototyp für Außeneinsätze unter starken Ressourcenbeschränkungen entwickelt und realisiert. Ergänzend dazu wird zur schnelleren Testdurchführung der Software eine Hardwaresimulation mit der entwickelten Simulation verknüpft, wodurch eine dezentrale Multilevelsimulation erreicht wird. Die Fähigkeit, sich zum einen mit Hardwaresimulatoren für einzelne Roboter und zum anderen mit verschiedenen Visualisierern zu verbinden, ist nur aufgrund der hohen Modularität und der effizienten Implementierung in C++ möglich.
Folglich wird ein einheitliches Autonomiesystem entworfen, das den gesamten Einsatzbereich von großen Schwarmsimulationen bis hin zur Regelung einzelner realer Agenten in unbekannten Umgebungen abdeckt.

Abstract

This thesis presents a concept of an autonomy system for autonomous robot swarms in disaster areas. To demonstrate the functionality, it is implemented and integrated into a prototype.
Autonomous robot swarms are promising for a variety of disaster missions. Typical tasks include reconnaissance of an area, convoy escort, object transportation, or network setup when infrastructure collapses. The Autonomous control of the robots is a challenging task, especially under tough environmental conditions limiting communication. Before the robot swarms can be deployed in disaster areas, their control software must be carefully tested. As field testing is complex, expensive, and in some cases impossible, the only viable solution is simulations. However, testing software with real-time requirements must be conducted in a real-time environment, which places strict requirements on the simulation software. Control must be part of the simulator, robot dynamics must be realistically simulated, environmental conditions must be described, and mobile communications must be modeled, including limited communication ranges and packet losses. The simulator should be user-friendly and efficiently implemented to allow simulation of larger swarms as they are used in practical applications.
Since no system with the required functions is known at this time, all necessary hardware and software components and protocols are developed or integrated. For this purpose, a sample-based model predictive controller for mobile robots is adapted and further developed. This controller achieves a very high, empirically verifiable control quality compared to a selection of optimization algorithms in extensive tests. A time bound of 300 ms is maintained for a 30 s long planning horizon. This is achieved by an exponential decomposition of the horizon. The theoretical stability is not provable due to cost function freedom and exponential distribution.
To build a network for rescue responders using a swarm of robots, and to operate the robots in wide-area infrastructure-less areas of up to 25 km^2, network quality predictions and low-cost mesh networks with long range and low transmission rate are presented. The low transmission rate forces the development of an application-specific data compression within the autonomy system. The system consists of C++ libraries with a tree-like dependency structure that are completely platform and processor architecture independent as well as modular. The system enables the operation of autonomous ground vehicles and the simulation of large swarms with up to 70 agents. These agents can be purely virtual, built from emulated hardware or from real components. The real-time simulation can be computed mixed centrally and decentrally and supports a fusion of simulation and real data up to emulation.
The design of the system allows easy integration into different visualization systems like Cocos2d-X, CryEngine and Unity. Thus, real-time applications without a graphical user interface or with a simple two-dimensional representation as well as photoreali-

stic applications in virtual reality can be created. The scope of over 16 missions offered by the agents is based on an extended potential field technique. Due to the freedom provided by the optimizer, no constraints exist on the continuity or differentiability of the underlying functions. Furthermore, the freedom of accumulation, convolution and concatenation of the functions provides a modular approach. With this, behavior tree methods or separated local and global planners can also be mapped. The missions range from classical waypoint navigation to surveillance tasks to complex, decentralized swarm transports and rendezvous. Jammer detection and energy management are also implemented.

To validate the entire software system, a real prototype is developed and implemented for field operations under severe resource constraints. In addition, a hardware simulation is linked to the developed simulation for a faster test execution of the software, whereby a decentralized multilevel simulation is achieved. The ability to connect to hardware simulators for single robots on the one hand and to different visualizers on the other hand is only possible due to the high modularity and the efficient implementation in C++.

Consequently, a unified autonomy system is designed that covers the whole range of applications from large swarm simulations to the control of single real agents in unknown environments.

Inhaltsverzeichnis

Nomenklatur

Matrizen, Vektoren und Variablen

$\mathbf{A}$	Systemmatrix
$\mathbf{B}$	Eingangsmatrix
$\mathbf{C}$	Ausgangsmatrix, Kovarianzmatrix
$\mathbf{C}_{a,b}$	Kovarianzmatrix
$\mathbf{C}_e$	Diagonalkovarianzmatrix der empfohlenen Stellgröße
$\mathbf{C}_u$	Diagonalkovarianzmatrix der Ableitungen
$\mathbf{K}$	Rückführungsmatrix
$\mathbf{P}$	Polygon
$\mathbf{P}_m$	Monotones Polygon
$\mathbf{Q}$	Zustandskostenmatrix, Kovarianzmatrix
$\mathbf{Q}_\gamma$	Tupelliste
$\mathbf{R}$	Stellgrößenkostenmatrix
$\mathbf{b}$	Dreiecksbasis
$\mathbf{d}$	Störgröße
$\Delta\mathfrak{t}$	Zeitschrittfolge, Zeitschrittdauervektor
$\mathbf{d}^{\hookrightarrow}$	Nutzdaten
$\mathbf{d}^{\hookrightarrow}_{\text{weiter}}$	Weitergeleitete Nutzdaten
$\mathbf{e}$	Reglerabweichung
$\mathbf{g}$	Ziel
$\mathbf{g}_p$	Pakete am Ziel $\mathbf{g}$
$\mathbf{i}_k$	Intervallstellgrößenvektor zum Zeitschritt k
$\mathfrak{i}_k$	Abgetasteter Intervallstellgrößenvektor zum Zeitschritt k
$\mathbf{l}$	Lot, Höhe des Dreiecks
$\mathbf{m}^{\hookrightarrow}$	Netzwerknachricht
$\mathbf{m}^{\hookrightarrow}_{\text{weiter}}$	Weitergeleitete Netzwerknachricht
$\mathfrak{p}^{\hat{r}}$	Potenzialobjekt des Agentenmodells $\hat{r}$
$\mathbf{q}^{\hat{r}}$	FIFO-Liste eines Agentenmodells $\hat{r}$
$\mathfrak{r}$	Agentenvektor
$\hat{\mathfrak{r}}$	Agentenmodellvektor
$\overset{\times}{\mathbf{s}}$	Sensor
$\mathfrak{t}$	Zeitreihe, Zeitpunktvektor
$\mathbf{u}$	Stellgrößenvektor
$\overset{\circ}{\mathbf{u}}$	Zeitkontinuierlicher Stellgrößenvektor
$\overset{\circ}{\mathbf{u}}^{\text{ref}}$	Kontinuierlicher Referenzstellgrößenvektor
$\overset{\diamond}{\mathbf{u}}$	Zeitdiskreter Stellgrößenvektor
$\hat{\mathbf{u}}_k$	Geschätzter Stellgrößenvektor zum Zeitschritt k

$\Delta\mathbf{u}_{k,\wp}$	Diskreter Stellgrößendifferenzvektor
$\overset{\diamond,\dashv}{\mathbf{u}}$	Beschränkter zeitdiskreter Stellgrößenvektor
$\overset{\diamond,\dashv^{r_k}}{\mathbf{u}}_k$	Beschränkter zeitdiskreter Stellgrößenvektor des Agenten r_k zum Zeitschritt k
$\overset{\diamond^{\mathrm{ref}}}{\mathbf{u}}$	Diskreter Referenzstellgrößenvektor
$\overset{\diamond}{\mathbf{u}}_{[n]}$	n-tes Element des Stellgrößenvektors
$\mathbf{u}_k^{g}$	basierter Stellgrößenvektor zum Zeitschritt k
$\mathbf{u}_k$	Stellgrößenvektor zum Zeitschritt k
$\mathbf{u}_k^{(n)}$	n-ter Stellgrößenvektor zum Zeitschritt k
$\mathbf{u}_-$	Minimaler Stellgrößenvektor
$\mathbf{u}_+$	Maximaler Stellgrößenvektor
$\tilde{\mathbf{u}}$	Optimaler Stellgrößenvektor
$\mathbf{w}$	Führungsgröße
$\mathbf{x}$	Systemzustandsvektor
$\mathbf{x}_0$	Initialer Systemzustandsvektor
$\overset{\diamond,\dashv^{\mathfrak{r}}}{\mathbf{x}}_0$	Initialer beschränkter zeitdiskreter Systemzustandsvektor des Agenten $\mathfrak{r}$
$\overset{\diamond,\dashv^{r_k}}{\mathbf{x}}_k$	Beschränkter zeitdiskreter Systemzustandsvektor des Agenten r_k zum Zeitschritt k
$\overset{\circ}{\mathbf{x}}$	Zeitkontinuierlicher Systemzustandsvektor
$\check{\mathbf{x}}_k$	Gemessener Systemzustandsvektor
$\overset{\circ^{\mathrm{ref}}}{\mathbf{x}}$	Kontinuierlicher Referenzsystemzustandsvektor
$\overset{\diamond}{\mathbf{x}}$	Zeitdiskreter Systemzustandsvektor
$\overset{\diamond,\dashv}{\mathbf{x}}$	Beschränkter zeitdiskreter Systemzustandsvektor
$\overset{\diamond,\dashv}{\mathbf{x}}_k\overset{\mapsto}{(}\overset{\diamond,\dashv}{\mathbf{u}}_k)$	Beschränktes zeitdiskretes Systemzustandsvektorinkrement abhängig von $\overset{\diamond,\dashv}{\mathbf{u}}_k$ zum Zeitpunkt k
$\dot{\mathbf{x}}$	Systemzustandsvektorableitung
$\overset{\diamond^{\mathrm{ref}}}{\mathbf{x}}$	Diskreter Referenzsystemzustandsvektor
$\overset{\diamond}{\mathbf{x}}_{[n]}$	n-tes Element des Systemzustandsvektors
$\mathbf{x}_k^{g}$	Gradientenbasierter Systemzustand zum Zeitschritt k
$\hat{\mathbf{x}}$	Geschätzter Systemzustandsvektor
$\hat{\mathbf{x}}_K$	Geschätzter Endzustandsvektor
$\hat{\mathbf{x}}_k$	Geschätzter Systemzustandsvektor zum Zeitschritt k
$\mathbf{x}_k$	Systemzustandsvektor zum Zeitschritt k
$\mathbf{x}_k^{(n)}$	n-ter Systemzustandsvektor zum Zeitschritt k
$\mathbf{x}_{\bowtie}$	Nächster Systemzustandsvektor
$\mathbf{x}^o$	Systemzustandsvektor des Objekts o
$\mathbf{x}^{\mathrm{ref}}$	Referenzsystemzustandsvektor
$\tilde{\mathbf{x}}_K$	Optimaler Endzustandsvektor
$\tilde{\mathbf{x}}_k$	Optimaler Systemzustandsvektor zum Zeitschritt k

$\tilde{\mathbf{x}}_k^1$	Zweitoptimaler Systemzustandsvektor zum Zeitschritt k
$\tilde{\mathbf{x}}_k^2$	Drittoptimaler Systemzustandsvektor zum Zeitschritt k
$\mathbf{y}$	Ausgangsvektor
$\mathbf{z}$	Trajektorie
$\mathbf{z}_{:,\wp-1}^{(:)}$	Vorherige Trajektorie abhängig von mehreren Prädiktionsschritten und Stichproben
$\mathbf{z}_k$	Trajektoriensegment zum Zeitschritt k
a	Hilfsvariable
c	Hilfsvariable
$\odot_a$	Achsenmittelpunkt
a_F	Gewichtungsfaktor der Formationsmission
$a_{\mathbf{g}}$	Attraktivität des Ziels $\mathbf{g}$
$a_{k,\wp}$	Größe a zum Zeitpunkt $t_{\wp+k}$
a_l	Obere Kostenschranke
$a_{\log}$	Parameter der logarithmischen Kostenfunktion
$a^{\hookrightarrow}$	Antwortcode
$a^{\leftsquigarrow}$	Störsender
b	Hilfsvariable
b_{1l}	Kostenfaktor
b_{2l}	Kostenfaktor
b_{cluster}	Entscheidungsvariable zur Clusterzuordnung
b_F	Gewichtungsfaktor der Formationsmission
$b_{\text{Kettenabstand}}$	Horizontaler Abstand zwischen den Kettenmitten
b_l	Kostenfaktor
$b_{\log}$	Parameter der logarithmischen Kostenfunktion
c	Cluster
c_l	Kostenkonstante
$c_{\log}$	Parameter der logarithmischen Kostenfunktion
CR	Kodierrate
$c^{\leftsquigarrow}$	Lichtgeschwindigkeit
d	Distanz
d_{cluster}	Zuordnungsdistanz
$d_\dagger$	Maximaler Blockierabstand
$\overset{\diamond}{\Delta t}_0$	Diskrete initiale Zeitschrittdauer
$\overset{\diamond}{\Delta t}_k$	Diskrete Zeitschrittdauer zum Zeitschritt k
Δt^*	Restzeitdifferenz
Δt^{tot}	Diskrete Totzeitdauer
d_{far}	Distanz des am weitesten entfernten Agenten
$d_{\mathbf{g}}$	Zwischenzielabstand
$\tilde{d}_{\mathbf{g}}$	Minimale Distanz zum Ziel $\mathbf{g}$
d_{join}	Vereinigungsdistanz
$d_{\mathbf{l}}$	Lotabstand
d_o	Abstand zum Objekt o
d^+	Distanzinkrement

$d^{\leftrightsquigarrow}$	Sendedistanz, Übertragungsstrecke
d_W	Wartedistanz
d_{wunsch}	Wunschdistanz
e	Kante
$e_{\log}$	Zielwert der logarithmischen Kostenfunktion
e_n	n-te Eingabe eines Neurons
$e^{\hookrightarrow}$	Netzwerkkante, Netzwerkverbindung
$f^{\leftrightsquigarrow}$	Trägerfrequenz
$\Delta f_c^{\leftrightsquigarrow}$	Kanalbreite
$\Delta f_t^{\leftrightsquigarrow}$	Trägerabstand
G_p	Genauigkeitsgewinn
$G_{r^{\leftrightsquigarrow}}$	Antennenverstärkung des Empfängers $r^{\leftrightsquigarrow}$
$G_{s^{\leftrightsquigarrow}}$	Antennenverstärkung des Senders $s^{\leftrightsquigarrow}$
i	Indexvariable
$i^{\hookrightarrow}$	Netzwerknachrichtidentifizierungsnummer
j	Indexvariable
K	Horizontlänge, Prädiktionshorizont
k	Zeitpunktindexdifferenz, Zeitschritt, Prädiktionsschritt
k_B	Bolzmann-Konstante
k^+	Zeitpunktindexdifferenz für die Totzeit
$\mathfrak{L}_{\min}$	Gesamtkostenminimum
l	Hilfsvariable, Indexvariable
L^{KNN}	Ausgabeschicht, Schichtanzahl eines KNNs
L_+	Kostenmaximum
L_-	Kostenminimum
l_{Rad}	Radabstand vom Achsmittelpunkt
M	Aufrufanzahl
m	Mission
$\odot_m$	Massenschwerpunkt
$m_{\mathbf{g}}$	Steigung der linearen Kostenfunktion des Ziels $\mathbf{g}$
m^{Rakete}	Masse
N	Stichprobenanzahl, Anzahl
n	Indexvariable
N_{eff}	Effektive Stichprobenanzahl
$N_{\text{eff}}^{\min}$	Minimale effektive Stichprobenanzahl
N^{KNN}	Neuronenanzahl einer KNN-Schicht
N_{rand}	Anzahl der zufällig generierten Stichproben
o	Objekt
$o^{\leftrightsquigarrow}$	Übertragungshindernis
p_1	Hilfsvariable
p_2	Hilfsvariable
p_3	Hilfsvariable
$P_{\mathcal{A}}^{\leftrightsquigarrow}$	Störsenderrauschen
p_{diff}	Maximale Roboter-zu-Ziel-Differenz
$P_e^{\leftrightsquigarrow}$	Grundrauschen

$P_{-}^{\leftrightsquigarrow}$	Sensitivitätsminimum
$P_{\mathcal{O}^{\rightsquigarrow}}$	Hindernisdämpfung
p^{+}	Maximaler Potenzialfunktionswert
$P_{r^{\rightsquigarrow}}$	Empfangsfeldstärkenindex, Empfangsfeldstärke (RSSI)
$P_{s^{\rightsquigarrow}}$	Sendeleistung
$\wp$	Situationszeitpunkt
$q_k^{(n)}$	n-te Realisierung einer Dichteverteilung
R	Zuverlässigkeit, Restterm
r	Radius
$r_{\mapsto\odot_a}$	Achsabstand
$r_{\mapsto\odot_m}$	Abstand des Ruders vom Massenschwerpunkt $\odot_m$
r_{Antrieb}	Radius des Antriebsrads
$\bar{r}$	Startradius
$\hat{r}$	Agentenmodell
$\hat{R}_N$	Geschätzte Zuverlässigkeitsabschätzung basierend auf N Stichproben
$r_{\ddagger}$	Blockierradius
r_{far}	Am weitesten entfernter Agent
$r_{\mathbf{g}}$	Bremsradius des Ziels $\mathbf{g}$, Zielakzeptanzradius
r_k	Agent zum Zeitschritt k
${r_k}^0$	Führeragent
$\hat{r}_k^0$	Führeragentenmodell
${r_k}^1$	Ankeragent 1
$\hat{r}_k^1$	Ankeragentmodell 1
${r_k}^2$	Ankeragent 2
$\hat{r}_k^2$	Ankeragentmodell 2
r_l	Kostenradius
$r_{\log}$	Einflussradius der logarithmischen Kostenfunktion
$r_{\tilde{\mathfrak{L}}}$	Optimale (minimale) Kosten bzw. maximale Gesamtbelohnung
$r'_{\mathfrak{L}}$	Diskontierte Gesamtbelohnung
r^{+}	Radiusschrittweite
$r^{\leftsquigarrow}$	Empfänger
s	Strecke
SF	Spreizfaktor
$s^{\hookrightarrow}$	Nachrichtengröße
s^{Rakete}	Zurückgelegte Distanz
s^{r_k}	Schenkel, dem der Agent r_k zugeordnet ist
$s_{\text{old}}^{r_k}$	Schenkel, dem der Agent r_k zugeordnet war
$s^{\leftsquigarrow}$	Sender
$\#s^{\leftsquigarrow}$	Senderanzahl
T	Trajektorie
t	Zeit
t_0	Startzeitpunkt
t^{Aus}	Ausregelzeit

$\diamond^{\text{ref}}$	
T	Diskrete Referenztrajektorie
$T_{[k]}$	k-tes Element der Trajektorie T
T^{g}	Gradientenbasierte Trajektorie
$t_{\mathbf{g}}$	Wartezeit am Ziel $\mathbf{g}$
t^{ges}	Benötigte Gesamtzeit
t_k	Zeitpunkt zum Zeitschritt k
$T^{(n)}$	Trajektorie mit Index n der Trajektorienschar $\mathcal{T}$
$t^{\hookrightarrow}$	Nachrichtentyp
$t_{\odot}$	Zeitpunkt des Rendezvous
$t^{\dashrightarrow}$	Aktualisierungszeitperiode, Zeit bis zum nächsten Inkrement
t^{+}	Zeitinkrement
$T^{\mathfrak{r}}$	Trajektorie des Agenten $\mathfrak{r}$
T^{ref}	Referenztrajektorie
$T^{\mathfrak{r}}_{\mathbf{x}_0^{\diamond,\dashv^{\mathfrak{r}}}}$	Trajektorie des Agenten $\mathfrak{r}$ beginnend im Initialzustand $\mathbf{x}_0^{\diamond,\dashv^{\mathfrak{r}}}$
t_{Schritt}	Zeitschrittwachstumsfaktor
$t_s^{\leftrightsquigarrow}$	Symbolzeit
$\tilde{T}$	Optimale Trajektorie
$T^{\leftrightsquigarrow}$	Temperatur
t_W	Wartezeitschwellwert
u	Stellgröße
$u^{\hookrightarrow}$	Zeitstempel
$u^{\hookrightarrow}_{-}$	Kleinerer Zeitstempel
$u^{\hookrightarrow}_{\text{weiter}}$	Zeitstempel einer weitergeleiteten Nachricht
$u^{\leftrightsquigarrow}$	Unterträger
$\acute{U}$	Stellgrößenzufallsvariable
v	Geschwindigkeit
$\bar{v}$	Mittlere Geschwindigkeit
$\dagger^{g}_{v_k}$	Gradientenbasierte Vortriebsstellgröße zum Zeitschritt k
$\dagger^{,\dashv}_{v}$	Beschränkte Vortriebsstellgröße (Geschwindigkeit)
$\dagger^{,\dashv}_{v\ -}$	Beschränkte minimale Vortriebsstellgröße (Geschwindigkeit)
$\dagger^{,\dashv}_{v\ +}$	Beschränkte maximale Vortriebsstellgröße (Geschwindigkeit)
v_I	Geschwindigkeit als ganzzahliger Wert
v_{Kette}	Kettenlaufgeschwindigkeit
v_{KetteL}	Linke Kettenlaufgeschwindigkeit
v_{KetteR}	Rechte Kettenlaufgeschwindigkeit
v_l	Linke Radgeschwindigkeit
$v^{\hookrightarrow}$	Netzwerkknoten
$v^{\hookrightarrow}_{+}$	Netzwerkknoten mit einer höheren Lamport-Uhr
$v^{\hookrightarrow}_{r^{\leftrightsquigarrow}}$	Empfängerknoten
$v^{\hookrightarrow}_{s^{\leftrightsquigarrow}}$	Senderknoten
$v_{\%}$	Verteilungsanteil in Prozent
v_r	Rechte Radgeschwindigkeit

v^{Rakete}	Geschwindigkeit
v_{RPM}	Geschwindigkeit in Umdrehungen pro Minute
w	Index
$w^{(j)}$	j-ter Gewichtungsfaktor
$w^{\leftrightsquigarrow}$	Symbolbreite
x	Systemzustand
$\hat{x}$	Geschätzte x-Koordinate
$\overset{\dagger,\dashv}{x}$	Beschränkte x-Koordinate
$\overset{\dagger^{\text{ref}}}{x}$	Referenz-x-Koordinate
$\acute{X}$	Zustandszufallsvariable
$\hat{y}$	Geschätzte y-Koordinate
$\overset{\dagger,\dashv}{y}$	Beschränkte y-Koordinate
Z	Normalisierungskonstante
$\acute{q}_k^{(n)}$	Stichprobenzufallsvariable einer Dichteverteilung
$\acute{\mathbf{u}}_k^{(n)}$	n-te Stellgrößenvektorzufallsvariable zum Zeitschritt k
$\acute{\mathbf{Z}}_k$	Trajektorienstückzufallsvariable

Funktionen

a	Aktivierungsfunktion
a^{KNN}	Anpassungskosten in einem KNN
c	Stellgrößenkostenfunktion
cs	Prüfsumme einer Netzwerknachricht $\mathbf{m}^{\hookrightarrow}$
d	Distanzmetrik, Abstandsfunktion
$d_{\mathbf{g}}$	Abstand von Ziel $\mathbf{g}$
diag	Diagonalmatrix
$d_{\text{LINE}}^{r_k}$	Abstand des Agenten r_k von der Formationslinie
$d^{\text{Zuverlässigkeit}}$	Distanzmetrik der Zuverlässigkeit
$\mathbb{E}$	Erwartungswertfunktion
$f_r^{\leftrightsquigarrow}$	Basiszirpfrequenzgang
$f_r w^{\leftrightsquigarrow}$	LoRa-Symbolfrequenzgang
h	Historienfunktion
hop	Hopabbildung eines Netzwerknoten $v^{\hookrightarrow}$
id	Identifikationsnummerabbildung eines Netzwerknoten $v^{\hookrightarrow}$
$\mathfrak{L}$	Gesamtkosten- bzw. Belohnungsfunktion
l	Allgemeine Kostenfunktion
l_{CENTER}	Zentrierungskostenfunktion
$\mathfrak{L}^{\text{CPBP}}$	Gesamtkostenfunktion des CPBP Algorithmus
$l_{\ddagger}^{r_k}$	Blockierungskostenfunktion des Agenten r zum Zeitschritt k
$L_{\text{EXPLORE}}^{r_k}$	Explorationskostenfunktion des Agenten r zum Zeitschritt k
$L_{\text{FOLLOW}}^{r_k}$	Eskortierungskostenfunktion des Agenten r zum Zeitschritt k
$l_{\text{FOLLOW}}^{r_k}$	Teil der Eskortierungskostenfunktion des Agenten r zum Zeitschritt k
$L_{\text{FORMATION}}^{r_k}$	Formationskostenfunktion des Agenten r zum Zeitschritt k

$l^{r_k}_{\text{FORMATION LINE}}$	Linienformationskostenfunktion des Agenten r zum Zeitschritt k
$L^{r_k}_{\text{GOTO}}$	Zielansteuerungskostenfunktion des Agenten r zum Zeitschritt k
$\hat{\mathfrak{L}}$	Geschätzte Gesamtkostenfunktion
$l_{\text{INTERSECTION}}$	Überschneidungskostenfunktion
$l'_{\text{INTERSECTION}}$	Alternative Überschneidungskostenfunktion
l_{konst}	Konstante Kostenfunktion
l_{linear}	Lineare Kostenfunktion
$ll_{\log}$	Logarithmische Kostenfunktion
$L^{r_k}_{\text{MANET}}$	MANET-Kostenfunktion des Agenten r zum Zeitschritt k
l_{MANET}	MANET-Kostenfunktion
$\mathfrak{L}^{\text{Optimum}}$	Gesamtkostenfunktion des optimalen Algorithmus
$l_{\text{quadratisch}}$	Quadratische Kostenfunktion
$\mathfrak{L}^{\mathfrak{r}}$	Gesamtkostenfunktion eines Agenten $\mathfrak{r}$
l_{rauschen}	Rauschende Kostenfunktion
$L^{r_k}_{\text{RENDEZVOUS}}$	Rendezvouskostenfunktion des Agenten r zum Zeitschritt k
L^{r_k}	Kostenfunktion eines Agenten r_k zum Zeitschritt k
l_{sozial}	Soziale Kostenfunktion
$L^{r_k}_{\text{TRANSPORT}}$	Transportkostenfunktion des Agenten r zum Zeitschritt k
$l_{\text{unbegrenzt}}$	Unbegrenzte Kostenfunktion
$\hat{m}_{\text{bwd}}$	Rückwärtsgerichtete geschätzte Nachricht
$\hat{m}_{\text{fwd}}$	Vorwärtsgerichtete geschätzte Nachricht
$m_{l\to k}$	Nachricht von l nach k
$\hat{m}_{l\to k}$	Geschätzte Nachricht von l nach k
O	Laufzeitkomplexität
π	Regelwerk
$\tilde{\pi}$	Optimales Regelwerk
Q^{π}	Erwartungswertfunktion der Gesamtbelohnung unter Regelwerk π
$\tilde{Q}$	Optimaler Erwartungswert der Gesamtbelohnung
r	Radius
$r_{\mathfrak{L}}$	Belohnungswertfunktion
$r^{r_k}_{\rightarrow}$	Sensorradiusüberschneidung eines Sensors $\overset{\ltimes}{\mathbf{s}}$ eines Agenten r zum Zeitschritt k
$b_{\text{TRANSPORT}}$	Binäre Transportzustandsfunktion
$s^{r_k}_{\ddagger}$	Blockierungsfunktion
V	Kostenfunktion
v	Zustandswertfunktion
v^{KNN}	Verlustfunktion eines KNN
w	Gewichtungsfunktion
z	Kontinuierliche Aktivierungszeitfunktion
$\mathfrak{f}$	Modell-, Übertragungsfunktion
$\mathfrak{f}^{\diamond,\dashv}$	Beschränkte zeitdiskrete Modellfunktion

$\mathfrak{f}^{\hookrightarrow}$	Fragmentierungsfunktion
$\mathfrak{G}$	Wegpunktliste, Pfad
$\mathfrak{N}$	Gaußfunktion
$\mathfrak{P}$	Wahrscheinlichkeit
$\mathfrak{s}$	System
$\mathfrak{u}$	Lamportuhr eines Netzwerknotens $v^{\hookrightarrow}$
$\mathfrak{z}$	Diskrete Aktivierungszeitfunktion
$\mathcal{B}_k$	Fragmentglaubwürdigkeit des Fragmentes k
$\hat{\mathcal{B}}_k$	Geschätzte Fragmentglaubwürdigkeit des Fragmentes k
$\oplus$	Applikationsoperator

Mengen

$\mathbb{D}_{\mathfrak{L}}$	Definitionsmenge der Gesamtkostenfunktion
$\mathbb{N}$	Natürliche Zahlen
$\mathbb{R}$	Reelle Zahlen
$\mathbb{T}_k$	Trajektorienmenge abhängig von k
$\mathbb{U}$	Zulässiger Stellgrößenraum
$\mathbb{U}_{\mathbb{X}_{\mathbf{x}_0}(t_k)}$	Zulässiger Stellgrößenraum abhängig vom zulässigen Zustandsraum ausgehend vom Startzustand $\mathbf{x}_0$ zum Zeitpunkt t_k
$\mathbb{X}$	Zulässiger Zustandsraum
$\mathbb{X}_{\mathbf{x}_0}(t_k)$	Zulässiger Zustandsraum ausgehend vom Startzustand $\mathbf{x}_0$ zum Zeitpunkt t_k
$\mathcal{A}^{\rightsquigarrow}$	Störsendermenge
$\mathcal{C}$	Clustermenge
$\mathcal{D}$	Dreiecksmenge
$\mathcal{E}$	Kantenmenge
$E^{\hookrightarrow}$	Netzwerkkantenmenge, Verbindungsmenge
G	Graph
$\mathcal{G}$	Ablage-, Paket-, Zielortmenge
$\mathcal{G}_p$	Absolute Paketmenge
$G^{\hookrightarrow}$	Netzwerkgraph
$\mathcal{M}$	Missionsmenge
$\mathcal{O}$	Objektmenge
$\mathcal{O}^{\rightsquigarrow}$	Netzwerkhindernismenge
Π	Orientierungsraum
$\mathcal{R}$	Agentenmenge
$\mathcal{R}_k$	Agentenmenge für einen Prädiktionsschritt k
$\hat{\mathcal{R}}_k$	Agentenmodellmenge für einen Prädiktionsschritt k
$\hat{\mathcal{R}}_k^{\text{left}}$	Agentenmodellmenge für den linken Schenkel
$\hat{\mathcal{R}}_k^{\text{right}}$	Agentenmodellmenge für den rechten Schenkel
$\mathcal{S}$	Streckenmenge
$\mathcal{T}$	Baumstruktur, Trajektorienschar
$\mathcal{U}$	Stellgrößenraum
$\mathcal{V}$	Knotenmenge
$\mathcal{V}_{\mathcal{M}}$	Missionsverteilungstupelmenge

$V^{\hookrightarrow}$	Netzwerkknotenmenge
$\mathcal{W}^{\leftrightsquigarrow}$	Symbolmenge
$\mathcal{X}$	Zustandsraum
$\mathfrak{P}_m$	Menge der monotonen Polygone
$\mathfrak{R}$	Agentenvektormenge
$\hat{\mathfrak{R}}$	Agentenvektormodellmenge

Verteilungen

$\mathcal{N}$	Normalverteilung
$\mathcal{P}$	Wahrscheinlichkeitsdichtefunktion, Bedingte Wahrscheinlichkeit
$\mathcal{P}_k$	Fragmentwahrscheinlichkeitsdichtefunktion des Fragmentes k
q	Dichteverteilung

Abkürzungen und Akronyme

ACPBP	Advanced Control Particle Belief Propagation
BFGS	Broyden-Fletcher-Goldfarb-Shanno
BP	Belief Propagation
CFS	Chirp Frequency Spreading
CPBP	Control Particle Belief Propagation
CRC	Zyklische Redundanzprüfung (Englisch: „Cyclic Redundancy Check“)
CS	Kompasssuche (Englisch: „Compass Search“)
CSMA / CA	Carrier Sense Multiple Access / Collision Avoidance
DARPA	Defense Advanced Research Projects Agency
DLL	Dynamische Progammbibliothek (Englisch: „Dynamic Linked Library“)
DQL	Deep Q-Learning
DQN	Deep Q-Network
DRL	Tiefgehendes Verstärkungslernen (Englisch: „Deep Reinforcement Learning“)
DSSS	Direct Sequence Spread Spectrum
DWA	Dynamic Window Approach
EKF	Erweiterter Kalman-Filter (Englisch: „Extended Kalman-Filter“)
ESA	Europäische Weltraumorganisation (Englisch: „European Space Agency“)
EST	Expansive Space Tree
FDMA	Frequency Devision Multiple Access
FIFO	First In First Out
FMT	Fast Marching Tree
GMM	Normalverteiltes Mischmodell (Englisch: „Gaussian Mixture Model“)
GPIO	Vielzweckein- und Vielzweckausgaberegister (Englisch: „General Purpose Input Output“)
GPS	Global Positioning System
HDMI	High Definition Multimedia Interface
IMU	Inertia Measurement Unit

IoT	Internet of Things
IPOPT	Interior Point Optimizer
ISM	Industrial, Scientific, Medical
ID	Identifikationsnummer (Englisch: „Node Identification Digit“)
KNN	Künstliches neuronales Netz
Knoten-ID	Knotenidentifikationsnummer (Englisch: „Node Identification Digit“)
kurz	Abgekürzt mit
LCM	Lightweight Communication and Marschalling
LIDAR	Laserabstandsmesser (Englisch: „Light Detection and Ranging“)
LM	Levenberg-Marquardt
LOD	Multiskalenrepräsentation (Englisch: „Level Of Detail“)
LoRa	Long Range
LoRaWAN	Long Range Wide Area Network
LOS	Strahlenmodell, Sichtverbindung (Englisch: „Line Of Sight“)
LSTM	Long Short-Term Memory
MANET	Mobiles Ad Hoc Netzwerk (Englisch: „Mobile Ad Hoc Network“)
MDP	Markov-Entscheidungsproblem (Englisch: „Markov Decision Problem“)
MIMO	Multiple Input Multiple Output
MOS-FET	Metall-Oxid-Halbleiter-Feldeffekttransistor (Englisch: „Metal-Oxid-Semiconductor Field-Effect Transistor“)
MPC	Modellprädiktive Regelung (Englisch: „Model Predictive Control“)
NASA	National Aeronautics and Space Administration
NMEA	National Marine Electronics Association
OFDM	Orthogonales Frequenzmultiplexverfahren (Englisch: „Orthogonal Frequency Devision Multiplexing“)
OMPL	Open Motion Planning Library
PBP	Particle Belief Propagation
PCMV	Prädiktiver Regler mit modellbasierter Validierung (Englisch: „Predictive Control with Model Validation“)
PDR	Packet Delivery Rate
PFMPC	Particle Filter Model Predicitve Control
PPO	Proximal Policy Optimization
PRM	Probablistic Road Map
PSK	Phase-Shift Keying
PSO	Partikelschwarmoptimierung (Englisch: „Particle Swarm Optimization“)
QAM	Quadrature Amplitude Modulation
QPSK	Quadrature Phase-Shift Keying
ReLU	Rectified Linear Unit
RL	Verstärkungslernen (Englisch: „Reinforcement Learning“)
RNN	Rekurrentes neuronales Netz

RN	Repeater Node
ROS	Robot Operating System
RRT	Rapidly Exploring Random Trees
RSSI	Empfangsfeldstärkenindex (Englisch: „Receiver Signal Strength Index“)
RT-RRT*	Time Rapidly Exploring Random Tree*
SDR	Short Range Device
SNR	Signalrauschabstand (Englisch: „Signal to Noise Ratio“)
SN	Sensor Node
SPI	Serielle Peripherieschnittstelle (Englisch: „Serial Peripheral Interface“)
SRAM	Direktzugriffspeicher (Englisch: „Static Random Access Memory“)
SSD	Solid State Drive
TDMA	Time Devision Multiple Access
TEB	Time-Elastic-Band
TEDS	Time-slotted Event-Driven System
UDP	User Datagram Protocol
VR	Virtuelle Realität (Englisch: „Virtual Reality“)
WLAN MAC	Wireless Local Area Network Media Access Control

Griechische Symbole

α	Diskontierungsfaktor
$\alpha_{\mathbf{g}}$	Gewichtungsfaktor der Transportkostenfunktion
α_k	Kostenpotenzialfunktion
β_k	Stellgrößenpotenzialfunktion
Δ	Dreieck
δ	Konfidenzintervallbreite
ϵ_H	A priori Genauigkeit
ϵ_l	Gleichverteilter Kostenwert
$\epsilon_{\bar{v}}$	Sicherheitsspanne für die Geschwindigkeit
η	Wirkungsgrad, Leistungsfaktor
Γ_k	Indikatorfunktion
γ_{1l}	Kostenexponent
γ_{2l}	Kostenexponent
$\overset{\dagger,\mathcal{G}}{\gamma}$	Gradientenbasierte Orientierung
$\overset{\dagger,\dashv}{\gamma}$	Beschränkte Orientierung
$\gamma_{\mathbf{g}}$	Gewichtungsfaktor der Transportkostenfunktion
Γ_k	Menge der Nachbarknoten eines Knotens k
$\gamma^{\leftrightsquigarrow}$	Signalqualität
$\gamma_{\bar{v}}$	Gewichtungsexponent
$\overset{\dagger,\mathcal{G}}{\iota}_k$	Gradientenbasierte Rotationsstellgröße zum Zeitschritt k
$\overset{\dagger,\dashv}{\iota}$	Beschränkte Rotationsstellgröße

$\overset{\dagger,\dashv}{\iota}_{+}$	Beschränkte maximale Rotationsstellgröße
$\overset{\dagger,\dashv}{\iota}_{-}$	Beschränkte minimale Rotationsstellgröße
ι_l	Lenkwinkel des linken Rads
ι_r	Lenkwinkel des rechten Rads
κ	Gewichtungsfaktor
κ_1	Gewichtungsfaktor
κ_2	Gewichtungsfaktor
κ_3	Gewichtungsfaktor
κ_{D_1}	Gewichtungsfaktor des Duffing-Oszillators
κ_{D_2}	Gewichtungsfaktor des Duffing-Oszillators
κ_{D_3}	Gewichtungsfaktor des Duffing-Oszillators
κ_{R_1}	Gewichtungsfaktor der Rakete
κ_{R_2}	Gewichtungsfaktor der Rakete
λ	Regularisierungsparameter
μ_1	Schlupfvariable
μ_2	Schlupfvariable
μ_3	Schlupfvariable
μ_4	Schlupfvariable
$\boldsymbol{\mu}$	Mittelwert
$\boldsymbol{\mu}_{k,\wp}^{(n)}$	Mittlerer empfohlener Stellgrößenvektor
μ_t	Mittelwert der Zeit t
ω	Schwellwert
ω_1	Gewichtungsfaktor
ω_2	Gewichtungsfaktor
ω_3	Gewichtungsfaktor
$\omega_{\ddagger}$	Gewichtungsfaktor der Blockierungsfunktion
ω^{l}	Gewichtungsfunktion
ϕ	Hilfswinkel
$\varphi^{\leftrightsquigarrow}$	Signalrauschabstand (SNR)
Ψ_{bwd}	Rückwärtsgerichtetes Übergangspotenzial
Ψ_{fwd}	Vorwärtsgerichtetes Übergangspotenzial
ψ_k	Kombinierte Potenzialfunktion
$\Psi_{k,l}$	Übergangspotenzial von k nach l
ρ	Hilfsvariable
$\rho^{\leftrightsquigarrow}$	Bruttoübertragungsrate
$\rho^{\overset{\times}{\leftrightsquigarrow}}$	Nettoübertragungsrate, Nutzdatenrate
$\rho^{\overset{\times}{\leftrightsquigarrow}}{}_{s}$	Nettoübertragungsrate pro Sender
$\rho_c^{\leftrightsquigarrow}$	Chiprate
$\rho_s^{\leftrightsquigarrow}$	Symbolrate
$(\rho^{\leftrightsquigarrow})'$	Basisdatenrate, Übertragungsrate
σ_0	Gewichtungsfaktor der Kovarianzmatrix
σ_1	Gewichtungsfaktor der Kovarianzmatrix
σ_2	Gewichtungsfaktor der Kovarianzmatrix
$\boldsymbol{\sigma}_0$	Standardabweichung

σ_1	Standardabweichung der ersten Ableitung
σ_2	Standardabweichung der zweiten Ableitung
σ_x	Standardabweichung des Systemzustandsvektors
σ_m	Mutationsfaktor der Kovarianzmatrix
σ_t	Standardabweichung der Zeit t
τ	Integrationsparameter
$\tau_{w^{\rightsquigarrow}}$	Symbolzeitverzug
$v^{\rightsquigarrow}$	Anzahl an parallelen Datenströmen
ξ	Gewichtungsfaktor
$\boldsymbol{\xi}$	Störgröße
ζ	Verteilungsfunktion

Der Aufbau der Nomenklatur unterliegt folgenden Regeln:

- In der Regel bezeichnen fettgedruckte Großbuchstaben Matrizen und fettgedruckte Kleinbuchstaben Vektoren.
- Punkte über einer Größe entsprechen den zeitlichen Ableitungen.
- $\hat{a}$ bezeichnet die Schätzung einer Größe a.
- $\breve{a}$ bezeichnet die Messung einer Größe a.
- $\breve{a}$ bezeichnet die Beobachtung einer Größe a.
- $\bar{a}$ bezeichnet den Mittelwert einer Größe a.
- $\tilde{a}$ bezeichnet das Optimum einer Größe a.
- $\acute{a}$ bezeichnet die Zufallsgröße einer Größe a.
- $\overset{\circ}{a}$ bezeichnet eine kontinuierliche Größe a.
- $\overset{\diamond}{a}$ bezeichnet eine diskrete Größe a.
- $\mathbf{a}_{[n]}$ bezeichnet das n-te Element eines Vektors $\mathbf{a}$.
- Kalligraphische Großbuchstaben bezeichnen vor allem Mengen.
- Großbuchstaben in Frakturschrift und „blackboard bold“ definieren spezielle Mengen.

1 Einleitung

Zu Beginn dieser Arbeit wird ein Einblick in das Themenfeld gegeben und Querbezüge aufgezeigt, die die Dissertation motivieren und die Forschungsfragen fixieren. Darauf folgt die Formulierung der Zielsetzung sowie die Erläuterung der Dokumentstruktur.

1.1. Motivation und Einordnung der Arbeit in das Forschungsfeld

Die Anwendung autonomer Roboter in der Industrie, dem Baugewerbe, der Geländeerkundung sowie im Katastrophenmanagement nimmt deutlich zu; ebenso der Einsatz in Krisengebieten z. B. zur Aufklärung (vgl. [131]). Besonders im Katastrophenfall, z. B. nach einem Erdbeben oder einem schweren industriellen Unfall, entstehen für den Menschen gefährliche Aufgaben wie die Erkundung, Rettungsaktionen und Bewegungen auf sowie Räumungen von Schutthaufen. Die Erledigung dieser Aufgaben und der Aufbau eines Kommunikationsnetzes sind wichtig und müssen so schnell wie möglich umgesetzt werden. Roboterschwärme sind im Prinzip gut geeignet, um diese Operationen durchzuführen. Idealerweise definiert ein menschlicher Einsatzleiter das übergeordnete Ziel der Mission und die Agenten des Schwarms stimmen sich selbstständig ab und treffen autonome Entscheidungen über Zwischenziele. Dabei steht ihnen in der Regel nur begrenztes Wissen über die Situation und den Zustand des gesamten Schwarms zur Verfügung. Am Markt verfügbar sind einerseits einzelne intelligente und komplexe autonome Roboter, die für eine Vielzahl an Missionen entwickelt worden sind. Jedoch sind diese sehr teuer in der Beschaffung und werden daher nur vereinzelt eingesetzt. Dies führt zu einem Systementwurf, bei dem jeder Roboter so konstruiert ist, dass er die Missionen möglichst alleine lösen kann. Daher wird die Kommunikation und Kollaboration mit anderen Agenten zumeist vernachlässigt. Andererseits gibt es kleine, kostengünstige Schwarmroboter, die sich auf Kommunikation und Zusammenarbeit konzentrieren, aber nur unzureichende Ressourcen und Fähigkeiten haben, um komplexe und lang andauernde Aufgaben zu absolvieren. Die Kombination von komplexen und teuren Roboterschwärmen zur Bewältigung anspruchsvoller Probleme in größerem Maßstab wird in der Forschung noch nicht vollständig berücksichtigt, denn die Vorteile und die Umsetzbarkeit dieser

intelligenten Schwärme müssen erst noch nachgewiesen werden.
Ein wichtiger Aspekt zur Erzeugung eines emergenten Schwarms ist die Kommunikation. Dazu beschäftigt sich diese Arbeit mit den frei verfügbaren Frequenzbändern. In einigen Veröffentlichungen wird versucht, die Netzwerkqualität und ihre Garantien zu verbessern, um in Realzeit Systeme im Netzwerk zu regeln (vgl. [36] und [112]) oder unter Annahme einer idealen Kommunikation das Optimalsteuerungsproblem für wenige Agenten zentral zu lösen (vgl. [34]). Es werden dazu Netzwerke mit hoher Übertragungsleistung angewendet, welche in der Regel nur sehr geringe Reichweiten aufweisen. Reale Anwendungsfälle zur Koordination von Rettungskräften auf den Philippinen (vgl. [24]) oder in Chile (vgl. [21]) verwenden stattdessen Technologien mit weiten Reichweiten, hoher Störresilienz und geringen Übertragungsraten, weil andere Netzwerktechniken eine zu große Infrastruktur benötigen, die nicht vorhanden ist. Folglich ist zu untersuchen inwieweit sich Roboterschwärme mit solchen Netzwerken betreiben lassen und sich die Kommunikation auf ein Minimum beschränken lässt. Ein solches Netzwerk kann ebenfalls von Rettungskräften genutzt werden, um im urbanen Raum durch Gebäude hindurch Daten zu versenden.
Jedoch muss die geringe Übertragungsrate und die damit beschränkte Informationslage in der Regelung, der Missionsplanung sowie der Pfadplanung beachtet werden. Mit vollständigen Ausfällen einzelner Agenten in Katastrophengebieten muss ebenfalls gerechnet werden. Dabei ist die Regelung einzelner autonomer robotischer Systeme in der Luft, zu Wasser und an Land ein langjähriges Forschungsfeld. Sehr früh hat sich auch die Potenzialfeldtechnik für Fahrzeuge, Schreitroboter und Luftfahrzeuge entwickelt (vgl. [181] und [95]). Diese Technik besitzt allerdings Schwachstellen bei lokalen Minima, falls die Gradienten der Felder verwendet werden (vgl. [35]). Innerhalb der Regelungstechnik ist die Pfadplanung eines der wesentlichen und am meisten erforschten Gebiete in der Robotik, insbesondere im Bereich der mobilen Robotik. Zunächst sind einfache Strategien und Heuristiken entwickelt worden, von denen einige kurzfristige Vorhersagen verwenden, wie der „Dynamic Window Approach“ (kurz DWA, vgl. [41]), oder optimale Lösungen für Gitterstrukturen, wie A* (vgl. [102]). Alternativ dazu werden graphbasierte Lösungen auf der Grundlage des Dijkstra-Algorithmus (vgl. [5]) angewandt. Mit zunehmender Komplexität der Umgebungen und Missionen für mobile Roboter wird die Berechnung exakter Lösungen zu aufwändig und Lösungen, die durch einfache Heuristiken abgeleitet werden, sind nicht gut genug. Daher werden graphbasierte Ansätze mit Zufallskomponenten, Metriken und Heuristiken zu „Rapidly Exploring Random Trees“ (kurz RRT, vgl. [123]) erforscht und zu Versionen, die zur Laufzeit funktionieren, mit erhöhter Konvergenzgeschwindigkeit unter Ausnutzung weiterer Informationen verbessert (vgl. [71], [59] und [120]).
Derzeit wird die Pfadplanung oft in lokale und globale Pfadplanung unterteilt, um die wachsende Komplexität zu bewältigen (vgl. [83]). Bei der globalen Pfadplanung werden komplexe Pfadstücke oder Zwischenpunkte auf einer viel gröberen Struktur berechnet, für die die oben genannten optimalen Algorithmen verwendet werden können (vgl. [198] und [62]). Diese Teile müssen dann von einem lokalen Planer in realisierbare Trajektorien für das jeweilige Fahrzeug umgewandelt werden. Sehr erfolgreiche Ansätze sind das „Time-Elastic-Band“ (kurz TEB, vgl. [158] und [156]) und dessen Weiterentwicklungen basierend auf der modellprädiktiven Regelung (Englisch: „Model

Predicitive Control“, kurz MPC, vgl. [160]). Die MPC ist eine äußerst rechenintensive, aber sehr leistungsfähige Methode zur Regelung komplexer Systeme (vgl. [50],[43] und [121]) und erlaubt ebenfalls die Verwendung von Kosten- bzw. Potenzialfunktionen und nutzt die durch ein Modell der Regelstrecke erhaltenen Informationen zur Erzeugung eines dünn besetzten Optimierungsproblems (vgl. [155]). Durch eine konvexe Formulierung der Potenzialfunktion ist es mit Hilfe der MPC möglich, einen Satellitenschwarm in Echtzeit zu regeln (vgl. [118]), der sich in einer Umgebung mit geringer Komplexität und Dynamik befindet. Rösmann [155] erforscht die zeitoptimale modellprädiktive Regelung und erzeugt Hypergraphen, um die dünn besetzte Struktur des Problems effizient auszunutzen, sodass die Anzahl der berechneten Gradienten zur Optimierung deutlich reduziert wird. Jedoch ist solch eine kontinuierliche Optimierung stets aufwändig, sodass Ansätze zur Reduzierung des Suchraumes durch Diskretisierung des Stellgrößenraumes untersucht werden (vgl. [113]). Diese Trajektorienscharregelung eignet sich besonders in Fällen, bei denen die Laufzeiteffizienz als wichtiger als die Regelgüte erachtet wird. Um den Rechenaufwand weiter zu reduzieren, kann erneut unter Reduktion der Regelgüte von zeitdiskreter modellprädiktiver Regelung zu ereignisdiskreter Regelung übergegangen werden (vgl. [52], [39], [98] und [132]). Dieses Vorgehen ist jedoch nur dann möglich, wenn die Agenten bei jedem internen sowie externen Ereignis benachrichtigt werden. Andernfalls wird erneut Rechenzeit durch die Verarbeitung kontinuierlicher Daten zur Detektion solcher Ereignisse verbraucht.

Das Problem der Aufteilung in globale und lokale Bahnplaner ist jedoch, dass der globale Planer aufgrund von dynamischen Änderungen oder Hindernissen keine realisierbaren Trajektorien generieren kann und zu langsam auf diese reagiert, während der lokale Planer die Bahnstücke verbindet und auf Hindernisse reagieren kann. Dabei optimiert er den lokalen Pfad, ohne Informationen von möglicherweise übergeordneten Missionen oder Formationen zu berücksichtigen. Mehrere Arbeiten stellen Methoden vor, die versuchen, die Pfadplanung wieder zu vereinen; ihre berechenbaren Missionen sind jedoch zu einfach und die Optimierungsqualität ist nicht ausreichend (vgl. [26] und [76]). In der aktuellen Forschung zur globalen Pfadplanung werden Partikelschwarmoptimierungen genutzt, um einzelne Agenten zu steuern und ausschließlich radiale Hindernisse zu umgehen (vgl. [26]). Dabei werden jedoch weder ein tatsächlicher Schwarm erzeugt, weil die Agenten ausschließlich für sich selbst agieren und keinerlei Kommunikationsmöglichkeiten besitzen, noch dynamische Umgebungen berücksichtigt. Ferner werden keine Missionen abgebildet, sondern einfache Trajektorien erzeugt, die sogar zu Kollisionen führen können. Die Erzeugung der Abtastpunkte für die Partikelschwarmoptimierung erfolgt zufällig und ist nicht modellbasiert, wodurch nicht mögliche Zustände zur Optimierung verwendet werden.

Der aktuelle Fokus der Schwarmforschung liegt vor allem auf unbemannten Luftfahrtzeugen, weil im Luftraum wenige Hindernisse vorkommen und die Kinematik der Systeme erforscht ist. Für die Regelung von Landfahrzeugen sind im Allgemeinen nichtlineare Modelle nötig, die nur schwer linearisiert werden können (vgl. [25]). Ansätze zur Steuerung mehrerer Systeme beruhen auf der Fluidtechnik (vgl. [202]) oder auf expliziten Lastverteilungsalgorithmen wie dem „Adaptive Response Threshold Model“ (vgl. [20]). Diese verwenden teilweise die Potenzialfeldtechnik auf unterschiedliche

Weise (vgl. [70]), um aufwändige Strategien zur Vermeidung von Kollisionen innerhalb von Formationen einzusparen (vgl. [200]). Damit eine Emergenz gebildet werden kann, müssen die autonomen Systeme innerhalb der Schwärme ihre Nachbarn beachten und ihr Verhalten schätzen. Dazu wird in der modellprädiktiven Regelung einzelner Systeme die Information über die Nachbarn berücksichtigt (vgl. [177], [152] und [34]). Jedoch löst bei diesen Ansätzen jeder Teilnehmer das Optimalsteuerungsproblem für jedes andere Mitglied im Schwarm mit. Dies ist nur unter der Annahme einer idealen Kommunikation möglich und führt zu einer zentralen Lösung des Problems. Folglich wird die Komplexität deutlich gesteigert, sodass maximal drei Systeme mit drei Zielen ermöglicht werden (vgl. [34]). Außerdem bildet dieses Vorgehen ein dezentrales und ausfallbehaftetes Schwarmverhalten nicht vollständig ab, da alle Informationen über die Gesamtsituation zu jeder Zeit vorliegen und verwendet werden. Die angewendeten genetischen Algorithmen zur Optimierung weisen langen Berechnungszeiten auf, die in der Luftfahrt zwar akzeptabel sind, aber bei Landfahrzeugen bereits zu Kollisionen führen können (vgl. [177]). Der Einfluss der Kommunikation, der bei der Regelung von Schwärmen meistens vernachlässigt wird (vgl. [177], [152] und [34]), muss daher deutlich mehr beachtet werden; ebenso wie der Einfluss der Missionen auf die lokalen Trajektorien. Um somit mobile Roboter in Katastrophengebieten als wirklich nützliche autonome Agenten und nicht als menschengesteuerte Unterstützer einsetzen zu können, ist es unerlässlich, dass die Agenten verschiedene, sich dynamisch verändernde Missionen in Abhängigkeit der Daten und ihrer Wahrnehmung auf nützliche, aber nicht zwingend optimale Weise erfüllen.

In diesem Zusammenhang werden der Flexibilität und Nutzbarkeit der Missionsergebnisse eine höhere Priorität eingeräumt als der Optimalität ihrer berechneten Pfade. Die aktuelle Missionsliteratur beschäftigt sich jedoch hauptsächlich mit optimaler Pfadplanung, z.B. für autonomes Fahren im Verkehr, mit der Exploration von Gebieten sowie mit der dezentralen Verteilung von Aufgaben an Agenten. Das letztgenannte Problem kann als ein „Job Shop Scheduling“ (vgl. [184] und [163]) oder als eine Aufgabenzuweisung (Englisch: „Task Allocation“, vgl. [74] und [166]) einzelner Missionen betrachtet werden, anstatt als ein kollaboratives Problem. Für jede dieser Aufgaben existiert eine spezifische und optimale Lösung, die in vielen Fällen nicht mit akzeptablem Aufwand berechenbar ist. Folglich wird eine allgemeine Missionsbeschreibung für verschiedene Aufgaben benötigt, die automatisch auf eine Menge von simulierten und realen Robotern verteilt wird, um große Roboterschwärme zu testen. Der Missionssatz sollte jederzeit erweiterbar sein und vollständig dezentral arbeiten, da die Nachrichten in der Regel über unzuverlässige Knotenpunktnetzwerke oder Rundfunksendungen (Englisch: „Broadcast“) gesendet werden. Daher können sich die Roboter nicht auf ein vollständiges Wissen über den Schwarm verlassen.

Um solche Ansätze zu testen und zu evaluieren, sind Simulationen notwendig, bevor ein Schwarm beschafft wird. Allerdings sind die aktuellen Robotersimulationsumgebungen nicht in der Lage solche Schwärme adäquat abzubilden (vgl. [88], [147], [83], [110], [82] und [116]). Es gibt das weit verbreitete Simulationswerkzeug „Gazebo“ (vgl. [82]) des „Robot Operating System“ (kurz ROS, vgl. [147]). Es wird für Herausforderungen der „Defense Advanced Research Projects Agency“ (kurz DARPA) (vgl. [23]) und der „National Aeronautics and Space Administration“ (kurz NASA) in

Tabelle 1.1.: Beschreibung der drei Grundbausteine eines Autonomiesystems sowie je drei Standardverfahren je Baustein.

Erkennung & Wahrnehmung		
Lokalistion & Kartografie	Sensordatenfusion & -überwachung	Tiefe selbstlernende Netze
⇓		
Planung & Entscheidung		
Missionplanung	Bewegungsplanung	Verhaltensplanung
⇓		
Regelung		
Pfadfolgeregelung	Regelungslogiken	Bestärkendes Lernen

der Robotik verwendet, ist aber derzeit nicht in der Lage, größere Schwärme, physikalische Übertragungseinflüsse und große Flächen von bis zu $25\,\mathrm{km}^2$ in Echtzeit zu simulieren. Es ist eher auf den Betrieb verschiedener Robotikanwendungen ausgerichtet und unterstützt viele verschiedene Systeme und Sensoren. Andere Simulatoren für große Schwärme arbeiten oft auf Gitterstrukturen oder unterstützen nur simple Regelungstechniken und physikalische Prinzipien. Daher konzentriert sich diese Arbeit auf realistische Annahmen mit Dynamik in einer kontinuierlichen Welt, was sich deutlich von bisherigen Forschungen zur Schwarmintelligenz in gitterbasierten Welten unterscheidet (vgl. [130] und [129]).

1.2. Zielsetzung

Da die meisten Forschungsaktivitäten zu autonomen Roboterschwärmen sich mit fliegenden Robotern beschäftigen, rückt die Dissertation bewusst Landfahrzeuge und die weniger erforschten Schreitroboter in den Fokus. Jedoch soll eine gewisse Generalität gewahrt bleiben, um auch mit heterogen Schwärmen arbeiten zu können, ohne das Konzept anzupassen. Die Bewegungsdimensionen sind eingeschränkter als diejenigen der fliegenden und tauchenden Systeme, jedoch existieren im Umfeld der betrachteten Systeme deutlich mehr statische und dynamische Hindernisse. Zudem sind die Möglichkeiten diese zu umgehen deutlich beschränkter als im Luftraum. Die Hindernisse beeinträchtigen sowohl die Pfadplanungs- als auch die Kommunikationsmöglichkeiten. Aktuelle Herausforderungen der Schwarmregelung sind die automatische Missionsplangenerierung, die eigenständige Verteilung der Unteraufgaben ohne Angabe eines expliziten Regelsatzes sowie die Koordinierung der Gesamtheit der Systeme ohne zentrale Instanz und unter Beschränkung des Wissens über die Gesamtsituation und der Kommunikationsmöglichkeiten. Das Ziel dieser Arbeit ist die Modellierung und Analyse einer verteilten, nichtlinearen modellbasierten Regelung für autonome Landfahrzeugschwärme, die sowohl die sensorischen als auch die durch eine realistische Kommunikation und die Umgebung induzierten Unsicherheiten berücksichtigt. Weiterhin soll der Regler nicht für ein spezielles Szenario optimiert werden, sondern universal einsetzbar sein. Dadurch wird die Regelungsgüte voraussichtlich geringer sein als diejenige der spezialisierten Lösungen. Ferner folgt aus der beschränkten Kommunikationsfähigkeit, dass kein Agent eine vollständige Wissensbasis der Gesamtlage

besitzt, noch mit Hilfe einer idealen unbeschränkten Kommunikation instantan dieses Wissen erlangen kann. Des Weiteren soll bewusst das Verhalten der anderen Agenten im Schwarm in die Regelung miteinbezogen werden, ohne deren Optimalsteuerungsproblem zu lösen. Dazu soll die Nutzbarkeit von Knotenpunktnetzwerken mit großen Reichweiten und geringen Übertragungsraten untersucht werden. Insgesamt soll ein vollständiges Autonomiesystem entwickelt werden, welches alle drei Grundbausteine der Autonomie umfasst (siehe Tabelle 1.1). Es soll sowohl simulativ den Nutzen komplexer und dezentraler Roboterschwärme anhand von modularen Missionen als auch die praktische Umsetzbarkeit solcher Techniken mit Hilfe realer Prototypen zeigen. Dies soll durch eine einheitliche Software geschehen, um Unterschiede zwischen realen und virtuellen Robotern zu minimieren und eine Verschmelzung realer und synthetischer Daten zu ermöglichen. Dadurch soll es möglich sein, die Software ausgiebig auf realen Prototypen zu testen und zeitgleich seltene Katastrophenszenarien durchspielen zu können.

1.3. Aufbau der Arbeit

Diese Arbeit wird in fünf grundlegende, weitgehend unabhängige Themenbausteine des Autonomiesystems aufgeteilt, die abschließend zur Lösung der Zielsetzung zusammengesetzt werden.

Kapitel 2 Das Kapitel beginnt mit einer Einführung in die lernenden Regelungsansätze (siehe Abschnitt 2.1), um diese anschließend gegenüber den planenden Ansätzen (siehe Abschnitt 2.2) abzugrenzen und zu untermauern, warum letztere in dieser Arbeit angewendet werden. Daraufhin werden die teilweise eigenständig formulierten nichtlinearen Modelle der Regelstrecke in Abschnitt 2.3 aufgestellt. Auf Grundlage der Modelle und der Problemdefinition (siehe Abschnitt 2.4.1) wird der zur Optimierung der Trajektorie leicht angepasste Basisalgorithmus (siehe Abschnitt 2.4.2) aus der Literatur beschrieben. Die wesentlichen Erweiterungen und Modifikationen der Optimierung werden anschließend in Abschnitt 2.4.3 erläutert. Darauf folgt die ausführliche empirische Zuverlässigkeits- und Güteanalyse des weiterentwickelten Optimierers anhand verschiedener Vergleichstests (siehe Abschnitt 2.5). Einige Testfälle stammen aus der Literatur und dienen als Maßstab, andere sind eigenständig entwickelt, um die Vor- und Nachteile der Verfahren hervorzuheben. Im Zwischenfazit (siehe Abschnitt 2.6) wird zusätzlich eine Übersicht über bekannte Verfahren und deren problembezogenen Eigenschaften gegeben.

Kapitel 3 Die Kommunikation der Agenten über drahtlose Knotenpunktnetzwerke wird in Kapitel 3 beschrieben. Zu Beginn wird die aus der Literatur bekannte logische Netzwerkordnung und -synchronisation auf Basis der Lamport-Uhren für Knotenpunktnetzwerke um den Aspekt der Weiterleitung erweitert (siehe Abschnitt 3.2). Nach der Beschreibung der verwendeten Frequenzbänder (siehe Abschnitt 3.3) wird ein eigenes Modell zur Schätzung der Übertragungsraten im 2,4 GHz-Netzwerk vorgestellt (siehe Abschnitt 3.4). Daraufhin folgt eine lite-

raturbasierte Einführung in die „Long Range“-Technologie für Netzwerke mit geringer Übertragungsrate und hoher Reichweite, die ebenfalls den Stand der Forschung in diesem Bereich abdeckt (siehe Abschnitt 3.5). Daraus abgeleitet werden in Kombination mit den gestellten Anforderungen ein eigenständiges Netzwerkprotokoll sowie dessen Realisierung in den Abschnitten 3.6 und 3.7. Das Protokoll wird anschließend in Abschnitt 3.8 getestet und auf Nutzbarkeit hin bewertet.

Kapitel 4 Die simulierte Hardware der Agenten und die virtuelle sowie reale Übertragungstechnik sind ein Teil der entwickelten Multiagentensimulation, auf deren Konzept im Detail in Kapitel 4 eingegangen wird. Zu Beginn wird die Optimierung als austauschbare Bibliothek gekapselt und in die Abhängigkeitsstruktur des neu entwickelten Autonomiesystems eingeordnet (siehe Abschnitt 4.1). Dessen einzelne Module werden daraufhin erklärt und softwaretechnische Kniffe erläutert (siehe Abschnitte 4.2.2). Die folgende Einbettung in verschiedene Visualisierer verwendet zu einem geringen Anteil Elemente einer vorherigen Simulationsversion (siehe Abschnitt 4.3). Abschließend folgt die Leistungsanalyse der Konzeptrealisierung in Abschnitt 4.4 mit einem Zwischenfazit (siehe Abschnitt 4.5).

Kapitel 5 In Kapitel 5 werden mathematische Modelle für modulare Missionen erarbeitet, die vollständig dezentral von den einzelnen Agenten verarbeitet werden und dennoch ein emergentes Verhalten induzieren. Dazu werden kurz die Erweiterung der Potenzialfeldtechnik um logarithmische Funktionen (siehe Abschnitt 5.1) sowie die missionsdefinierenden (siehe Abschnitt 5.2) und -behandelnden (siehe Abschnitt 5.3) Module des Autonomiesystems präsentiert. Darauf folgt stets mit identischem Aufbau die Beschreibung der eigenständig entworfenen Missionen (siehe Abschnitte 5.4 bis 5.16). Jede dieser Missionen wird zunächst eingeführt und formal definiert, bevor die Funktionsfähigkeit durch Experimente und deren Analyse dargelegt wird. Zum Schluss folgt die automatische Verteilung der Missionen (siehe Abschnitt 5.17) und eine Übersicht (siehe Abschnitt 5.18).

Kapitel 6 Die Realisierung des Autonomiesystems wird mittels eines Prototyps validiert, dessen eigenständige Entwicklung aus verschiedenen Beschränkungen und Anforderungen resultiert (siehe Abschnitte 6.1 und 6.2). Weiterhin wird die Fähigkeit zur Multilevelsimulation und die Integration realer Hardware zur Emulation in den Abschnitten 6.3 und 6.4 experimentell aufgezeigt.

Kapitel 7 Die gewonnenen Erkenntnisse werden in Kapitel 7 zusammengefasst, die Limitationen aufgezeigt sowie ein Ausblick auf weitergehende Forschung beschrieben.

2

Regelung autonomer Agenten

Aufgrund der in Abschnitt 1.2 beschriebenen Forschungsziele besteht die Aufgabe der Regelung autonomer Agenten in dieser Arbeit darin, verschiedene Missionen durch kooperierende und geplante Aktionen und Bewegungen der Agenten zu absolvieren und dabei auf Störungen zu reagieren. Dazu werden Regler benötigt. Im Kontext der Robotik wird oft der Begriff der Pfad- bzw. Bewegungsplanung verwendet (vgl. [114]). Es ist innerhalb des Forschungsfeldes eines der wichtigsten und meist-erforschten Themen; vor allem im Bereich der mobilen Robotik. Dabei ist dies nur eine Komponente des vollständigen Regelkreises (siehe Abbildung 2.1a). Er unterscheidet sich grundlegend von einer Steuerung durch die Messung und Rückführung des Systemausgangs (siehe Abbildung 2.1b). Für allgemeine Grundlagen der Regelungstechnik wird auf Adamy [3], Lunze [105] und Lunze [107] verwiesen.

Da die einzelnen Agenten ein Mehrgrößensystem bilden (siehe Kapitel 6) und weder eine explizite Führungsgröße noch eine Regelabweichung existieren (siehe Kapitel 5), ist die Verwendung von Proportional-Integral-Differential-Reglern (kurz PID-Regler) sowie deren Erweiterung mit „Anti-Windup“ und Durchgriff (siehe Anhang C.1) für die alleinige Regelung nicht ausreichend. Sie werden allerdings als kaskadierte Regler (vgl. [148]) in den unteren Ebenen der Motorregler verwendet (siehe Kapitel 6). Die Informationen über die Zustände und Ausgangssignale der Gesamtheit der Agenten, auch als Schwarm bezeichnet, sind nicht zentral verfügbar (siehe Abschnitt 1.2), da sie über ein dezentrales Netzwerk verbreitet werden (siehe Kapitel 3). Zudem erfordern einige Missionen (siehe Kapitel 5) Informationen über die Vergangenheit und bieten Zukunftsprädiktionen, die für eine erhöhte Regelungsgüte verwendet werden können. Daher entfällt die Möglichkeit reine Zustandsregler zu verwenden (siehe Abschnitt 2.1.8), da sie ausschließlich den aktuellen Zustand verwenden. Es ist zwar

(a) Regelung (b) Steuerung

Abbildung 2.1.: Vergleich zwischen Steuerung und Regelung.

möglich, den Zustandsvektor um alte Zustände des eigenen Systems zu erweitern, aber aufgrund der dezentralen verteilten Systeme und der Zukunftsbetrachtung ist dies nicht ausreichend und somit nicht zielführend. Stattdessen werden Konzepte benötigt, die diese Dezentralität der Informationen sowie deren zeitlichen Verlauf in Form von Zeitreihen verarbeiten können. Es gibt zwei Hauptkategorien, die dies leisten:

1. Lernende Ansätze und Algorithmen (siehe Abschnitt 2.1)

   Definition 2.0.1: *Als lernende Ansätze werden modellfreie Optimierungsalgorithmen klassifiziert, die aufgrund von Interaktion mit einem System aus dessen Rückgabewerten datenbasiert Fähigkeiten mittels neuronaler Netze erlernen.*

2. Planende Ansätze und Algorithmen (siehe Abschnitt 2.2)

   Definition 2.0.2: *Als planende Ansätze werden modellbasierte Optimierer klassifiziert, die ihre Ausgabe vor der Interaktion mit einem System anhand eines Modells dessen bewerten und optimieren, um anschließend die Rückgabewerte des Systems zur Anpassung zu verwenden.*

Dabei unterscheiden sich die Konzepte grundlegend. Allerdings lassen sie sich auch kombinieren und ineinander integrieren (vgl. [7]). Sie arbeiten jeweils auf Kostenfunktionen, die die gewählten Aktionen bewerten. Im Kontext der lernenden Ansätze wird statt des Begriffs der Kostenfunktion „Belohnungsfunktion“ verwendet. Der Unterschied besteht ausschließlich im Optimierungsziel. Ist das Ziel eine Maximierung, wird der Begriff der Belohnung verwendet. Diese lässt sich durch die Umkehrung des Vorzeichens in Kombination mit einer Minimierung in Kosten transformieren. Somit lassen sich Missionsziele vorgeben und das Verhalten der Agenten vergleichen und optimieren. Lernende Ansätze arbeiten in der Regel datenbasiert und erlernen daraus Verhaltensregeln, wohingegen die planenden Ansätze explizit das Verhalten anhand von Modellen prädizieren und daraus die Aktionen zur Laufzeit ableiten. Dem gegenüber benötigen die lernenden Algorithmen eine vorherige Trainingsphase, die jedoch zu einer deutlichen reduzierten Laufzeit zur Ausführungszeit führt (vgl. [108] und [188]).

2.1. Lernende Ansätze und Algorithmen

Im Folgenden werden verschiedene Verfahren beschrieben und auf ihre Anwendbarkeit für die in Kapitel 1 definierten Anforderungen hin untersucht. Des Weiteren wird die für die Regelungstechnik typische Notation verwendet, um Analogien und Gemeinsamkeiten zwischen den lernenden und planenden Methoden hervorzuheben.

2.1.1. „Deep Reinforcement Learning“-gestützte Regelung

Die Anwendung des tiefgehenden Verstärkungslernen (Englisch: „Deep Reinforcement Learning“, kurz DRL) wird für die missionsbasierte Regelung der Agenten auf

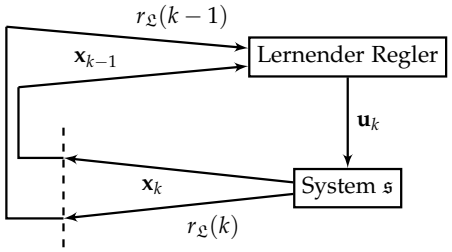


Abbildung 2.2.: Zyklische Struktur einer „Reinforcement Leraning"-Regelung (vgl. [180]). Der lernende Regler gibt eine Stellgröße $\mathbf{u}_k$ auf das System $\mathfrak{s}$ und erhält diesem einen neuen Zustand $\mathbf{x}_k$ und einen Belohnungswert $r_{\mathfrak{L}}(k)$.

Tauglichkeit analysiert (vgl. [180]). DRL versucht, optimierte Stellgrößen $\mathbf{u}_k$ für Zustandstransitionen eines Systems $\mathfrak{s}$ über einen zeitlichen Horizont der Länge $K \in \mathbb{N}$, im Kontext der lernenden Verfahren auch als Episode bezeichnet, zu erlernen. Zur Bestimmung einer Stellgröße $\mathbf{u}_k \in \mathcal{U}$ des Stellgrößenraumes $\mathcal{U}$ zum Zeitpunkt $t_k \in \mathfrak{t} = [t_k]_{k=1,2,\dots,K}$ für ein System $\mathfrak{s}$ im Zustand $\mathbf{x}_{k-1} \in \mathcal{X}$ des Zustandsraumes $\mathcal{X}$, welche zu einer Zustandstransitionen $\mathfrak{s}(\mathbf{x}_{k-1}, \mathbf{u}_k, \cdots) = \mathbf{x}_k$ führt, wird die Belohnung $r_{\mathfrak{L}}^{r_k}(k) \in \mathbb{R}$ einer Belohnungsfunktion $\mathfrak{L}$ (siehe Kapitel 5) für einen Agenten $[r_k]_{k=1,2,\dots,K} = \mathfrak{r} \in \mathfrak{R}$ einer Agentenmenge $\mathfrak{R}$ in einem künstlichen neuronalen Netz (kurz KNN) (siehe Abschnitt 2.1.3) verarbeitet. Durch die Interaktion mit dem System $\mathfrak{s}$ wird aufgrund der Maximierung der $r_{\mathfrak{L}}^{r_k} = \sum_{k=1}^{K} r_{\mathfrak{L}}^{r_k}(k)$ ein gewünschtes Verhalten erlernt. Dabei bildet der Aktions-Belohnungs-Zyklus das Grundprinzip des verstärkenden Lernens (Englisch: „Reinforcement Learning", kurz RL), siehe Abbildung 2.2. Das Anlernen des KNN, bezeichnet als Training, beginnt ohne Kenntnis über das System $\mathfrak{s}$ im Zustand $\mathbf{x}_0$ und durchläuft eine Episode bis zum letzten Zeitschritt K zur Zeit t_K oder bis ein unzulässiger Zustand $\mathbf{x} \in \mathcal{X} \setminus \mathbb{X}$ des Zustandsraums erreicht wird. Hier bezeichne $\mathbb{X}$ die Menge der zulässigen Zustände. Nach dem Abschluss der Episode oder dem Erreichen eines zulässigen Zustands wird die Belohnung verarbeitet und das System in den Zustand $\mathbf{x}_0$ zurückgesetzt. Darauf folgt die nächste Episode bis zur Konvergenz des KNNs, erkennbar an einer asymptotisch gleichbleibenden Belohnung $r_{\mathfrak{L}}$.

2.1.2. Markov-Entscheidungsproblem

Zur Verarbeitung der Belohnung $r_{\mathfrak{L}}$ wird das System $\mathfrak{s}$ als sequenzielles Entscheidungsproblem aufgefasst. Dieses lässt sich als Markov-Entscheidungsproblem (Englisch: „Markov Decision Problem", kurz MDP) formalisieren (vgl. [179, 45]), welche in Abbildung 2.3 abgebildet sind. MDPs sind als Tupel $(\mathcal{X}, \mathcal{U}, \mathcal{P}, \mathfrak{L}^{\mathrm{MDP}})$ bestehend aus dem Zustandsraum $\mathcal{X}$, den Stellgrößen $\mathcal{U}$, den bedingten Wahrscheinlichkeiten $\mathcal{P}(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_k) \in [0,1]$ für eine Transition von $\mathbf{x}_{k-1}$ nach $\mathbf{x}_k$ unter Anwendung von $\mathbf{u}_k$ sowie der Belohnungsfunktion $\mathfrak{L}_{\mathbf{u}_k}^{\mathrm{MDP}}(\mathbf{x}_{k-1}, \mathbf{x}_k) \in \mathbb{R}$ für eine Stellgröße $\mathbf{u}_k$ von $\mathbf{x}_{k-1}$ nach $\mathbf{x}_k$ definiert. Dabei gilt, dass die Summe der Wahrscheinlichkeit $\mathcal{P}$ über einen Zustand $\mathbf{x}_{k-1}$ für alle Aktionen $\mathbf{u}_k \in \mathcal{U}$ Eins ergibt:

$$\forall \mathbf{u}_k \in \mathcal{U}. \sum_{\mathbf{x}_k \in \mathcal{X}} \mathcal{P}(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_k) = 1 \tag{2.1.1}$$

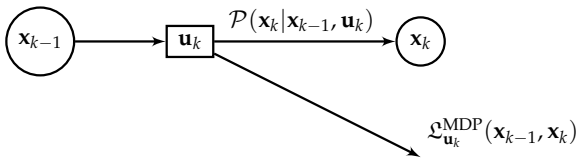


Abbildung 2.3.: Struktur eines Markov-Entscheidungsproblems (vgl. [45]).

Ein vollständig bekanntes MDP kann mit der Methode der dynamischen Programmierung optimal gelöst werden (vgl. [153] und [12]). Jedoch korreliert die Komplexität des Systems $\mathfrak{s}$ mit der Größe des zugehörigen MDPs. In der Regel entspricht die Korrelation einem exponentiellen Wachstum. Als Konsequenz steigen auch die benötigten Ressourcen zur optimalen Lösung (vgl. [115]). Um dies zu verhindern, approximiert RL die Wahrscheinlichkeitsverteilung $\mathcal{P}$ und die Belohnungsfunktion $\mathfrak{L}^{\text{MDP}}$ für $\mathbf{u}_k$ mittels der im Training erfahrenen Episodendaten (vgl. [153]). Für MDPs und somit auch für die RL-Lösungsstrategie muss die Annahme gelten, dass der Folgezustand $\mathbf{x}_k$ und die Belohnung $r_{\mathfrak{L}}(k)$ ausschließlich vom aktuellen Zustand $\mathbf{x}_{k-1}$ abhängt. Somit darf keine zeitliche Abhängigkeit zwischen den Zuständen und der Stellgröße existieren. Dies ist einer der größten Nachteile und Einschränkungen der RL-Verfahren, da die Belohnungsfunktion bzw. Kostenfunktion komplexer Missionen Zeitabhängigkeiten aufweisen und meistens nicht nur über den Systemzustand eines einzelnen Agenten definiert sind (siehe Kapitel 5).
Zur Approximation existieren verschiedene Methoden wie beispielsweise das tabellenorientierte „Q-Learning“ (vgl. [190, 38]) sowie dessen Erweiterung das „Deep Q-Learning“ (kurz DQL) (vgl. [150]), welches die Tabelle zur Funktionsapproximation durch ein KNN ersetzt und ein Rückspielpuffer (Englisch: „Replay Buffer“) ergänzt. Dies führt zu einer erhöhten Effizienz (vgl. [8]). Das Ziel dieser Algorithmen ist es, die Gesamtbelohnung $r_{\mathfrak{L}}$ ausgehend von einem Zeitpunkt t_0 mit Zustand $\mathbf{x}_0$ für eine gegebene Horizontlänge $K \in \mathbb{N}$ zu maximieren.
Dazu sei $Q^{\pi}(\mathbf{x}_0, \mathbf{u}_1)$ eine Funktion über dem Zustand $\mathbf{x}_0 \in \mathcal{X}$ und der folgenden Stellgröße $\mathbf{u}_1 \in \mathcal{U}$, die den Erwartungswert der Gesamtbelohnung für ein Regelwerk π definiert (vgl. [93]):

Definition 2.1.1: *Ein Regelwerk (Englisch: „Policy“) $\pi(\mathbf{u}|\mathbf{x})$ beschreibt die Strategie der Stellgrößenwahl $\mathbf{u}$ innerhalb eines MDPs ausgehend vom Zustand $\mathbf{x}$.*

Beispiel 2.1.1: *Für ein lineares System*

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{A}\mathbf{x}(t) - \mathbf{B}\mathbf{u}(t) \\ \mathbf{y}(t) &= \mathbf{C}\mathbf{x}(t) \end{aligned} \tag{2.1.2}$$

mit der Kostenfunktion

$$V(\mathbf{x}(t), \mathbf{u}(t)) = \int_t^{\infty} \mathbf{x}^T(\tau)\mathbf{Q}\mathbf{x}(\tau) + \mathbf{u}^T(\tau)\mathbf{R}\mathbf{u}(\tau)d\tau \tag{2.1.3}$$

kann ein optimales Regelwerk $\tilde{\pi}$ mit

$$\tilde{\pi}(\mathbf{x}(t)) = \mathbf{u}(t) = -\mathbf{K}\mathbf{x}(t) \tag{2.1.4}$$

angegeben werden, das die Kostenfunktion minimiert (vgl. [107] und [188]).

Diese Strategien sind nicht vorgegeben, sondern werden von einem RL-Algorithmus erlernt, um ein optimales Regelwerk $\tilde{\pi}$ zu bestimmen, welches zu einer maximalen Gesamtbelohnung $\tilde{r_{\mathfrak{L}}}$ führt. Um die Gesamtbelohnung zu begrenzen und die Berechnung eines optimalen Regelwerks zu bestimmen, muss die Gesamtbelohnung diskontiert werden (vgl. [93]):

$$r'_{\mathfrak{L}} = \sum_{k=1}^{\infty} \alpha^{k-1} r_{\mathfrak{L}}(k) \tag{2.1.5}$$

Sei $\alpha \in [0, 1)$. Dies ermöglicht beliebige Horizontlängen K und spiegelt zeitgleich die zunehmenden anzunehmenden Unsicherheiten über den zukünftigen Systemzuständen $\mathbf{x}$ des Systems $\mathfrak{s}$ wider, welche stets in realen Systemen inhärent sind. Diese können in der Regel nie vollständig modelliert werden oder unterliegen, wie in der Ausgangssituation in Kapitel 1 beschrieben, nicht kontrollierbaren Störgrößen. Aufgrund der Belohnungsstruktur fokussieren die RL-Verfahren auf die Optimierung der ersten Stellgröße (vgl. [93]). Mit Hilfe der diskontierten Gesamtbelohnung lässt sich der sogenannte Zustandswert $v(\mathbf{x})$ eines Zustandes $\mathbf{x}$ für ein Regelwerk π definieren:

$$v(\mathbf{x}) = \mathbb{E}_{\pi}\left[r'_{\mathfrak{L}} | \acute{X} = \mathbf{x}\right] \tag{2.1.6}$$

Dieser Erwartungswert gibt die erwartete diskontierte Gesamtbelohnung des Zustands $\mathbf{x}$ unter Anwendung des Regelwerks für eine beliebige, allerdings fest gewählte, Episodendauer K an. Auf Grundlage dieses Wertes kann die Funktion $Q^{\pi}(\mathbf{x}, \mathbf{u})$ nun definiert werden (vgl. [194]):

$$Q^{\pi}(\mathbf{x}_0, \mathbf{u}_1) = \mathbb{E}\left[r'_{\mathfrak{L}} | \acute{X} = \mathbf{x}_0, \acute{U} = \mathbf{u}_1\right] \tag{2.1.7}$$

Unter Anwendung der Bellmann-Gleichung (vgl. [12]) optimiert DQL den Erwartungswert, indem rückwärtsgewandt $k = K-1, K-2, \ldots, 1$ das optimale Teilregelwerk $Q^{\pi}(\mathbf{x}_k | \mathbf{u}_{k+1})$ in jedem Episodenschritt k mit $Q^{\pi}(\mathbf{x}_{k_1}, \mathbf{u}_k) = r_{\mathfrak{L}}(k) + \alpha Q^{\pi}(\mathbf{x}_k, \mathbf{u}_{k+1})$ ausgewählt wird. Somit wird das optimale Regelwerk $\tilde{\pi}$ für alle Zustände und Stellgrößen ermittelt:

$$\tilde{Q}(\mathbf{x}, \mathbf{u}) = \max_{\tilde{\pi}} Q^{\tilde{\pi}}(\mathbf{x}, \mathbf{u}) \tag{2.1.8}$$

Ein KNN wird zur Näherung von $\tilde{Q}$ verwendet, da die Verteilung $\mathcal{P}$ nicht vollständig bekannt sein muss bzw. nicht kontrollierbaren Störeinflüssen unterliegt.

2.1.3. Künstliche neuronale Netze

Die Funktionsapproximation innerhalb des DQLs erfolgt mittels künstlicher neuronaler Netze (Englisch: „Feed-forward neuronal networks“, kurz KNN, vgl. [176]). Diese werden bei überwachten Lernverfahren dazu verwendet, nichtlineare Funktionen nach einer Anlernphase zu approximieren und werden im Kontext DQL als „Deep Q-Network“ (kurz DQN) bezeichnet (vgl. [150]). Bei überwachtem Lernen werden Tupel, bestehend aus Zustand $\mathbf{x}$, Stellgröße $\mathbf{u}$ und Belohnung $r'_{\mathfrak{L}}$ mit der Approximation durch das KNN in einer Verlustfunktion verrechnet, um durch die Minimierung dieser das Netz anzupassen. Der grundlegende Aufbau sowie die Funktionsweise von

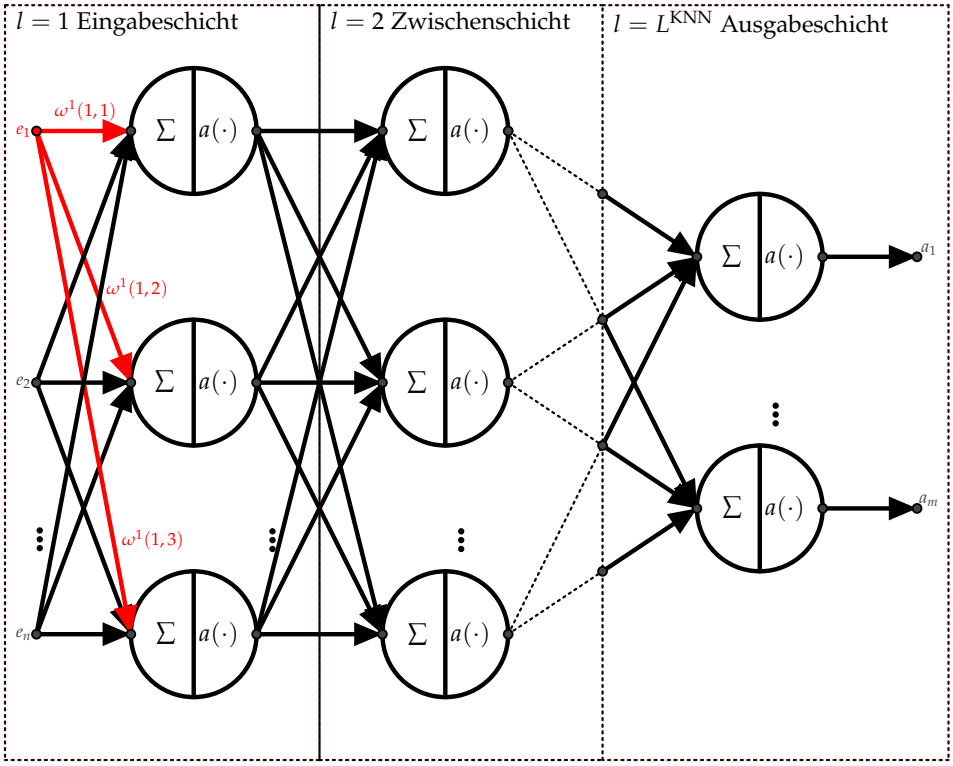


Abbildung 2.4.: Struktur eines künstlichen neuronalen Netzes.

KNNs werden in der Monografie von Sutton und Barto [179] beschrieben. Weiterhin werden verschiedene Anlernverfahren der Netze in der Monografie von Wilmott [194] detailliert erläutert.

2.1.3.1. Neuronen und Schichten

KNN sind geschichtete gerichtete Graphstrukturen mit $l = 1, 2 \ldots, L^{\mathrm{KNN}}$ Schichten (Englisch: „Layer“), visualisiert in Abbildung 2.4. Die Schichten bestehen aus N^{KNN} Neuronen, die jeweils N^{KNN} Eingaben e_n mit $n = 1, 2, \ldots, N^{\mathrm{KNN}}$ und eine Ausgabe besitzen. Die Eingänge werden jeweils mit $\omega^l(n_1, n_2) \in [-1, 1]$ Gewichten gewichtet aufsummiert und als Parameter für eine Aktivierungsfunktion a verwendet. Diese Funktion bildet die Ausgabe des Neurons, welche für nachfolgende Neuronen als Eingabe dient. Als Aktivierungsfunktion können beispielsweise Tangens Hyperbolicus, Sigmoid-Funktionen oder „Rectified Linear Unit“ (kurz ReLU) angewendet werden (vgl. [126]). Die Eingabeschicht $l = 1$ definiert die Eingabe des gesamten KNNs und die Ausgabeschicht $l = L^{\mathrm{KNN}}$ definiert die Ausgabe. Die Verknüpfung der Schichten kann je nach Netztyp vollständig sein, bezeichnet als „voll vernetzt“ (Englisch: „Fully Connected Layer“), oder auch nur teilweise. Weitere Modifikationen ermöglichen unterschiedliche Neuronenanzahlen pro Schicht, z. B. in „Convolutional Neural Networks“, oder die Verwendung von Ausgaben nachgelagerter Neuronen als Eingabe für Neuronen vorheriger Schichten (siehe Abschnitt 2.1.6 und 2.1.7, vgl. [194]).

2.1.3.2. Anlernen künstlicher neuronaler Netze

Die Trainingsphase eines KNNs soll die Funktionsapproximation auf die Trainingsdaten anpassen, sodass optimale Vorhersagen erzielt werden. Dazu wird eine Verlustfunktion v^{KNN} definiert, welche mittels rückwärtiger Anpassung (Englisch: „Backpropagation“) durch das KNN minimiert wird:

$$v^{\mathrm{KNN}} = (Q^{\pi}(\mathbf{x}, \mathbf{u}) - a^{\mathrm{KNN}}) \tag{2.1.9}$$

Die rückwärtige Anpassung ist ein Gradientenabstiegsverfahren, welches die Gewichte $\omega^{l}(n_1, n_2)$ anhand eines Tupels $(\mathbf{x}_{k-1}, \mathbf{u}_k, r_{\mathfrak{L}}(k), \mathbf{x}_k)$ modifiziert. Dies wird in den Veröffentlichungen von Peters [133] und Rumelhart, Hinton und Williams [161] erörtert. Um mit v^{KNN} den Fehler zwischen der Vorhersage im Schritt k mit $k = 1, 2, \ldots, K$ und der tatsächlichen erhaltenen Belohnung auszudrücken, wird a^{KNN}(k) wie folgt definiert:

$$a^{\mathrm{KNN}}(k) = \begin{cases} r_{\mathfrak{L}}(k), \text{ falls } (\mathbf{x}_{K-1}, \mathbf{u}_K, r_{\mathfrak{L}}(k), \mathbf{x}_K) \\ r_{\mathfrak{L}}(k) + \alpha \max\limits_{\mathbf{u}_k \in \mathcal{U}} Q^{\pi}(\mathbf{x}_k, \mathbf{u}_{k+1}), \text{ sonst} \end{cases} \tag{2.1.10}$$

Damit die Stabilität der Vorhersage verbessert wird, werden Rückspulpuffer (Englisch: „Replay Buffer“) angewendet, die mehrere Tupel speichern und gruppenweise verarbeiten (vgl. [150] und [154]).

2.1.4. Direkter Regelwerkaufbau

DQL gehört zur Klasse der vorab trainierten Verfahren (Englisch: „Off-Policy“, vgl. [179] und [153]). Dahingegen existieren auch dynamisch trainierte Verfahren (Englisch: „On-Policy“), wie beispielsweise „Proximal Policy Optimization“ (kurz PPO) (vgl. [164]). Diese verwenden die letzten N^{PPO} Tupel $(\mathbf{x}_{k-1}, \mathbf{u}_k, r_{\mathfrak{L}}(k), \mathbf{x}_k)$ aus der Interaktion mit dem System $\mathfrak{s}$, um eine optimale Strategie zu bestimmen (vgl. [179]). Durch die konstante Anpassung des Regelwerks basierend auf einer Fenstergröße von N^{PPO} Tupeln erhöht sich die Stabilität und die Anpassungsfähigkeit der Strategien gegenüber Parameteränderungen des Systems und erhöht die Konvergenzgeschwindigkeit im Vergleich zu vorab trainierten Verfahren (vgl. [27, 179]).
Im PPO-Verfahren werden Ableitungen des Regelwerks (Englisch: „Policy Gradients“) bestimmt; diese werden in der Veröffentlichung von Peters [133] beschrieben. Dynamisch trainierte Verfahren wenden die KNN nicht zur Approximation der Belohnungswerte an, sondern stattdessen, um direkt optimale Stellgrößen $\tilde{\mathbf{u}}$ zu ermitteln (vgl. [164]). Zur Formalisierung einer dafür nötigen Verlustfunktion kann z. B. „Vanilla Policy Gradient“ (vgl. [164]) genutzt werden. Die Ausgabe des KNN entspricht anschließend einer Wahrscheinlichkeitsverteilung über dem Stellgrößenraum $\mathcal{U}$. PPO weist somit einige Parallelen zum „Control Particle Belief Propagation“-Algorithmus auf (siehe Abschnitt 2.4.2). Ferner implementiert PPO die Funktionsweise des „Macher-und-Prüfer“-Prinzips (Englisch: „Actor-Critic Prinziple“) (vgl. [179, 164]). Dieses besteht aus einem Aktorregelwerk π und einer Bewertungsfunktion, die mittels eines Zeitversatzfehlers (Englisch: „Time Difference Error“) bestimmt, ob die auf das System gegebene Stellgröße eine höhere oder niedrigere Belohnung ausgibt als erwartet.

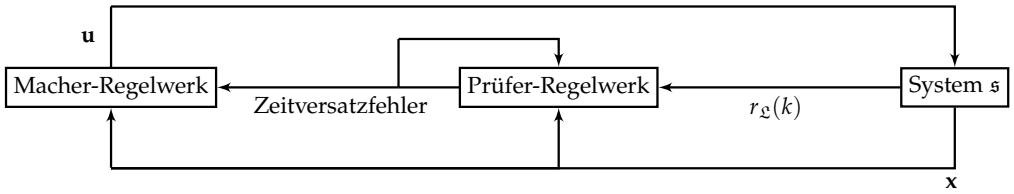


Abbildung 2.5.: Wirkprinzip eines „Macher-und-Prüfer"-Systems (vgl. [180]). Es besteht aus einem „Macher-Regelwerk", das eine Stellgröße **u** auf das System $\mathfrak{s}$ gibt, und aus einem Prüfregelwerk, welches den Zustand **x** und die Belohnung $r_{\mathfrak{L}}(k)$ mit der berechneten Erwartung vergleicht. Daraus resultiert der Zeitversatzfehler (Englisch: „Time Difference Error"), der die Regelwerke anpasst.

Die Bewertungsfunktion kann dabei ebenfalls durch ein KNN abgebildet werden (vgl. [180]). Das Wirkungsprinzip ist in Abbildung 2.5 schematisch dargelegt.

2.1.5. Markov-Entscheidungsproblem mit reduzierter Beobachtbarkeit

MDPs setzen voraus, dass alle zur Bestimmung der optimalen Stellgröße **u** benötigten Informationen im Zustand **x** enthalten sind (siehe Abschnitt 2.1.2). Komplexere Missionen (siehe Kapitel 5) erfordern zur Lösung jedoch auch zeitliche Informationen sowie vergangene Zustände sowie Wissen über andere Agenten im Schwarm. Dies entspricht einem teilweise beobachtbarem Markov-Entscheidungsproblem (Englisch: „Partially Observable Markov Decision Problem", kurz POMDP), bei dem sowohl Unsicherheiten bezüglich der Stellgrößenwahl als auch bezüglich des Zustands existieren (vgl. [192]). Eine Möglichkeit zur Auflösung dieses Problems ist die Erweiterung des Zustands **x** zu einer Sequenz einer festen Länge vorheriger Zustände sowie die Ergänzung des Zustands eines Agentensystems um die Zustände der anderen Agenten im Schwarm (vgl. [117]). Somit wird die teilweise Beobachtbarkeit in eine vollständige Beobachtbarkeit umgewandelt, sodass das POMDP in ein MDP umgewandelt wird. Dieses kann erneut durch die RL-Verfahren DQL (siehe Abschnitt 2.1.1) und PPO (siehe Abschnitt 2.1.4) gelöst werden. Allerdings ist die Systemkomplexität nun deutlich erhöht, wodurch die Konvergenzgeschwindigkeit sinkt (vgl. [179, 164]).

2.1.6. Rekurrente neuronale Netze

Statt zeitliche Abhängigkeiten der Zustände über Zustandssequenzen als Eingabe in das KNN zu implementieren, können Zyklen in die Graphstruktur eingebaut werden. Die Zyklen erzeugen Eingaben für Neuronen vorgelagerter Schichten aus Ausgaben derselben oder nachfolgender Schichten. Dies induziert eine zeitverzögerte Eingabe, wodurch indirekt vergangenes Wissen abgebildet wird. Dieser Ansatz wird als rekurrentes neuronales Netz (kurz RNN) bezeichnet (vgl. [49]) und in Abbildung 2.6 dargestellt. Diese Zyklen werden ebenfalls gewichtet und mit der rückwärtigen Anpassung (Englisch: „Backpropagation") angepasst (vgl. [193]).

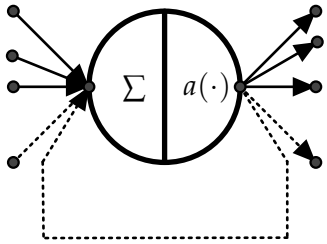


Abbildung 2.6.: Es ist ein rekurrentes Neuron abgebildet, bei dem der Ausgang als Eingang zurückgeführt wird.

2.1.7. Langfristiges Kurzzeitgedächtnis

In dem in Abschnitt 2.1.6 vorgestellten RNN existieren keine Kontrollmechanismen zur Speicherung und Löschung der Informationen eines rekurrenten Neurons. In wie weit die Information eines Neurons erhalten bleibt, beruht ausschließlich auf der Gewichtung der Zyklenkante (vgl. [176]). Diese rückgerichteten Kanten bilden ein integrierendes Verhalten analog zum Integralanteil eines PID-Reglers (siehe Abschnitt 2 und Anhang C.1). Der Integralanteil kann zu einem unerwünschten unbeschränkten Wachstum der Stellgröße führen; folglich werden bei PID-Reglern mit Stellgrößenbeschränkungen „Anti-Windup"-Elemente eingebaut (siehe Anhang C.1). Bei RNN wird die langfristige Kurzzeitgedächtnismethode (Englisch: „Long Short-Term Memory", kurz LSTM) (vgl. [64, 49]) verwendet, welche die rekurrenten Neuronen erweitert (siehe Abbildung 2.7):

1. Neuroneingangsgatter, welches dem vorherigen Eingang entspricht
2. Löschungsgatter, welches die alte Information löschen kann
3. Eingangsaktivierungsgatter, welches den eigentlichen Eingang „abschalten", also mit Null multiplizieren, kann
4. Durchgriffsgatter, welches den Eingang mittels eines Durchgriffs direkt auf den Ausgang schalten kann

Der Aufbau eines rekurrenten LSTM-Neurons ist ähnlich zu einem stellgrößenbeschränkten PID-Regler mit „Anit-Windup" und Durchgriff (siehe Anhang C.1).

2.1.8. Bewertung der Anwendbarkeit

Aus den in Kapitel 1 formulierten Anforderungen an das zu entwickelnde System resultieren dynamische Einsatzszenarien in unbekannten Umgebungen. Diese Generalität des Einsatzes erfordert umfangreiche und zufällig gestalte Trainingsdaten, die zu einer hohen Anlernzeit führen. Weiterhin werden die modularen und erweiterbaren

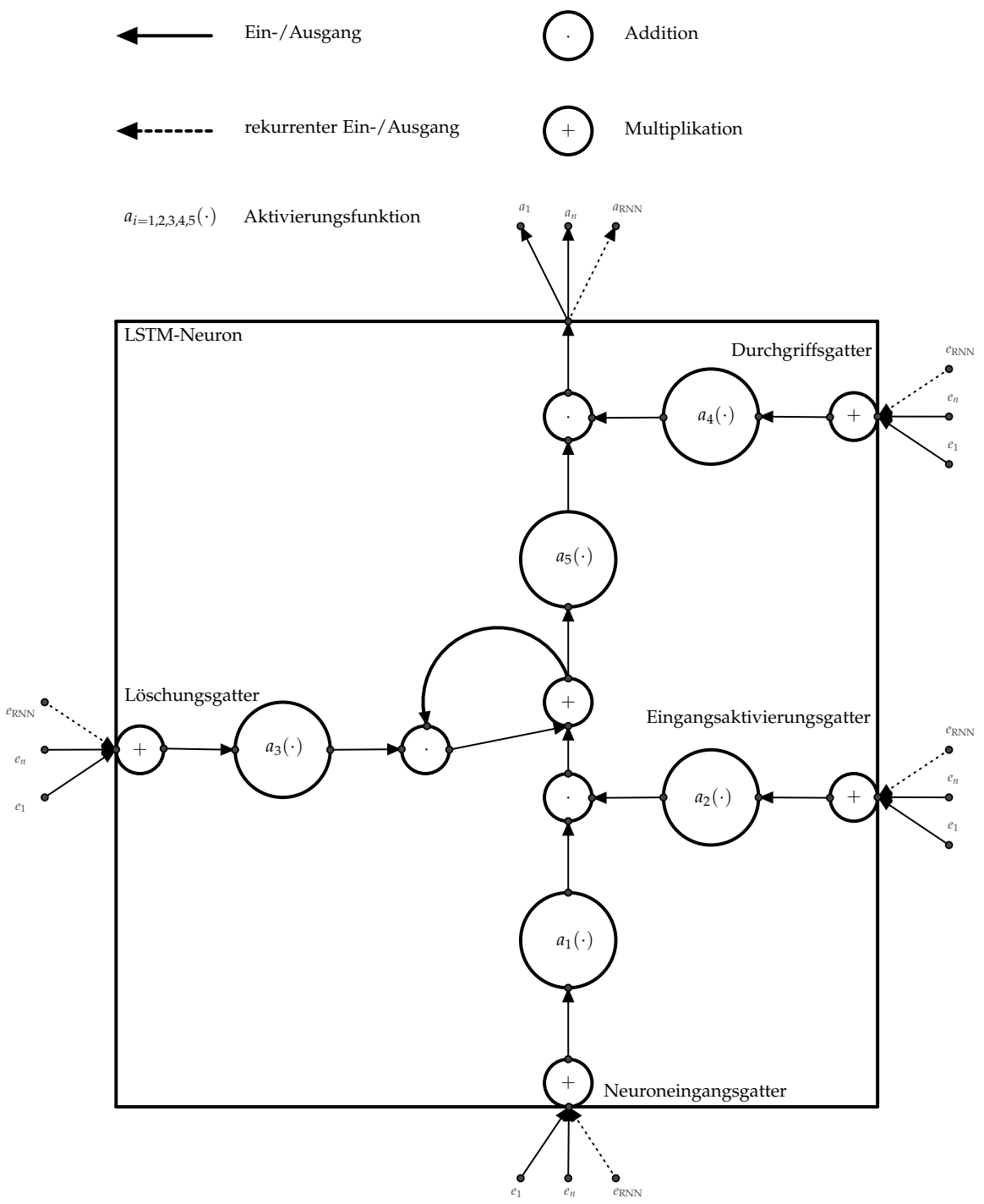


Abbildung 2.7.: Neuron der langfristigen Kurzzeitgedächtnismethode (vgl. [64]). Das Neuron akzeptiert Eingangskanten an allen vier Gattern, welche mit Hilfe der Aktivierungsfunktionen $a_{i=1,2,3,4}(\cdot)$ gesteuert werden. Die Eingangskanten können sowohl klassische Kanten als auch rekurrente Kanten sein. Das Eingangsaktivierungsgatter kann aufgrund einer Null-Multiplikation den Neuroneneingang deaktivieren. Das Löschungsgatter kann vorherige Informationen addieren oder subtrahieren. Ferner erlaubt das Durchgriffsgatter einen direkten Einfluss des Eingangs auf den Ausgang. Die Basisaktivierungsfunktion des LSTM-Neurons bildet $a_5(\cdot)$.

Missionen (siehe Kapitel 5) erst zur Laufzeit generiert und können stets angepasst werden. Dadurch ist ein vollständiges Training der verwendeten Netze vor dem Einsatz in einer Produktivumgebung nicht möglich. Jedoch können gerade in den angestrebten Katastrophenszenarien zufällige Aktionen des Systems zum dynamischen Anlernen zu ungewolltem Systemverhalten führen. Daher sind solitäre lernende Ansätze nicht als Regler für den in dieser Arbeit angestrebten Einsatz nutzbar.
Darüber hinaus können zeitabhängige Missionen zwar mittels rekurrenter Netze abgebildet werden; allerdings erzwingt die Interaktion mit anderen Agenten im Schwarm die Erweiterung des eigenen Zustandsraumes. Dies resultiert in einer deutlich erhöhten Systemkomplexität und verringert die Dezentralität des Lösungsverfahrens. Als letztes Argument gegen den Einsatz lernender Verfahren spricht, dass die Größe des Schwarms flexibel ist, wodurch auch die Anzahl der Eingänge des KNNs zur Laufzeit änderbar sein muss, falls der Zustandsraum um die anderen Agenten im Schwarm erweitert wird.
Vorteilhaft ist jedoch die automatische Adaption der Verfahren an geänderte Parameter und das implizite Erlernen nicht bewusst entwickelter Szenarien durch Generalisierung. Daher eignen sich die Verfahren in dem Kontext dieser Arbeit zur Modellbildung des Realsystems und der Fusion von Sensordaten zur Rekonstruktion der Systemzustände. Dem gegenüber ermöglicht die explizite mathematische Modellierung der Missionen und des Systemverhaltens eine allgemeinere und prädiktive Regelung mittels Optimierern zur Laufzeit.

2.2. Planende Ansätze: Modellbasierte Regelung

Die Beschreibung der Eigenschaften lernender Ansätze (siehe Abschnitt 2.1) sowie deren Abgleich mit den Anforderungen der Zielsetzung führt zur Auswahl der modellbasierten Regelung. Allgemein kann ein solcher Regler wie folgt definiert werden (vgl. [203]):

Definition 2.2.1: *Ein modellbasierter Regler beinhaltet das Modell der zu regelnden Strecke als Bestandteil des Regelalgorithmus.*

Es wird dafür vorausgesetzt, dass die Übertragungsfunktion der Regelstrecke genau bestimmt wird. Diese Reglerklasse kann in drei Unterklassen aufgeteilt werden:

1. Kompensationsregler sind Regler, die die Strecke vollständig kompensieren.

2. Smith-Prädiktor ist ein Kompensationsregler mit Totzeit.

3. Prädiktiver Regler ist ein Regler, der die Stellgröße während des Regelvorgangs an die gewünschte Sprungantwort anpasst.

Dabei wird Signallaufzeit vom Systemeingang bis zum Systemausgang als Totzeit bezeichnet (vgl. [3]). Zur Lösung der Aufgabe wird ein prädiktiver Regler verwendet. Die anderen Regler spielen eine untergeordnete Rolle zum Beispiel im Bereich der Störungskompensation zur Geschwindigkeitsregelung.

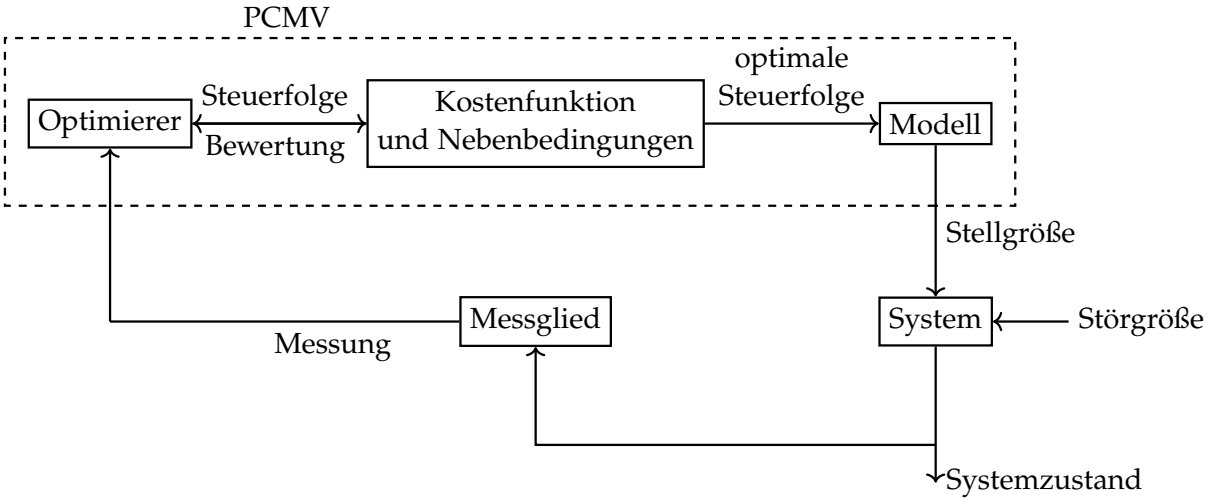


Abbildung 2.8.: Prädiktiver Regler mit modellbasierter Validierung.

Die prädiktiven Regler können das Streckenmodell auf unterschiedliche Weise verwenden. So gibt es zum einen die prädiktiven Regler mit modellbasierter Validierung und zum anderen die modellprädiktiven Regler.

2.2.1. Prädiktive Regler mit modellbasierter Validierung

Ein prädiktiver Regler mit modellbasierter Validierung (Englisch: „Predictive Controller with Model Validation“, kurz PCMV) beginnt mit der modellunabhängigen Exploration einer Kostenfunktion $\mathfrak{L}$, die der Beschreibung der Missionen (siehe Kapitel 5) oder einer Abweichung von einer Führungsgröße (vgl. [50]) entspricht (siehe Abbildung 2.8). Das Ziel der Exploration ist stets die Auffindung des Minimums, daher werden Optimierer oder Abtastverfahren verwendet. Klassische Abtastverfahren sind:

1. Zufällig gleich verteilte Stichproben
2. Gitterbasierte Stichproben
3. Latin Hypercube Design (vgl. [183])
4. Improved Latin Hypercube Design (vgl. [11])
5. Halton Sequenz (vgl. [56])

Für dynamische Stichprobengrößen gibt es die folgenden adaptiven Einfügeverfahren:

1. Kriging (vgl. [67] und [29])
2. Expected Improvement (vgl. [146] und [204])

Statt dieser Abtastverfahren werden auch abtastbasierte Optimierungsverfahren eingesetzt, die von einem Systemzustand $\mathbf{x} \in \mathcal{X}$ aus die Kostenfunktion $\mathfrak{L}$ explorieren und zum globalen Minimum konvergieren. In der mobilen Robotik werden diese Verfahren daher als *globale Pfadplaner* bezeichnet (vgl. [114]) und unterscheiden sich damit von den lokalen Pfadplanern, die nur zu lokalen Minima hin optimieren.
Der globale Pfadplanungsalgorithmus von Dijkstra arbeitet dabei graphbasiert und findet optimale Wege nur innerhalb einer beschränkten Knotenmenge, die einer Stichprobenanzahl in der Definitionsmenge $\mathbb{D}_{\mathfrak{L}}$ von $\mathfrak{L}$ entspricht (vgl. [5]). Der A*-Algorithmus arbeitet ebenfalls auf einer endlichen abzählbaren Stichprobenmenge, in dem der Definitionsraum $\mathbb{D}_{\mathfrak{L}}$ der Kostenfunktion $\mathfrak{L}$ mittels eines regulären dimensionsentsprechenden Gitters diskretisiert wird. Von der Startzelle an, in der sich der Systemzustand $\mathbf{x}$ befindet, wird das Gitter jeweils um die aktuell kostengünstigste Zelle herum evaluiert bis zum Erreichen des Minimums. Dieser Algorithmus verwendet dabei Baumstrukturen oder Prioritätslisten, die die Zellen nach ihren Kosten aufsteigend sortieren und einfügen (vgl. [102]).
Als weitere Optimierungsklasse funktionieren die „Rapidly Exploring Random Trees“ als eine Kombination aus Abtastverfahren und graphbasierter Optimierung.

2.2.1.1. Rapidly Exploring Random Trees

Die grundlegende Idee der „Rapidly Exploring Random Trees“ (kurz RRT) (vgl. [96]) ist, dass die Definitionsmenge $\mathbb{D}_{\mathfrak{L}} = \mathcal{X}$ sequenziell abgetastet wird, zum Beispiel mit Hilfe des Voronoi-Bias (vgl. [101]). Dies entspricht dem Ziehen einer Zufallsvariablen $\mathbf{x} \in \mathcal{X}$ aus einer gewählten Verteilung $\varsigma(\mathcal{X})$ (vgl. Algorithmus 2.1). Anschließend wird die Baustruktur $\mathcal{T}$ in die Richtung der neuen Stützstelle $\mathbf{x}$ erweitert, indem anhand einer Distanzmetrik d oder der Kostenfunktion $\mathfrak{L}$ der nächste Knoten $\mathbf{x}_{\bowtie}$ zum Stützpunkt gewählt wird (siehe Abbildung 2.9a). Von diesem Knoten und ggf. weiteren Nachbarknoten aus, wird ein neuer Knotenpunkt $\mathbf{x}_k$ bestimmt, der durch eine Kantenmenge $\mathcal{E}$ verbunden wird.

Definition 2.2.2: *Eine Kante $e \in \mathcal{E}$ wird definiert als Zustandsfolge im Raum $\mathcal{X}$, zum Beispiel als lineare Interpolation zwischen $\mathbf{x}_{\bowtie}$ und $\mathbf{x}$.*

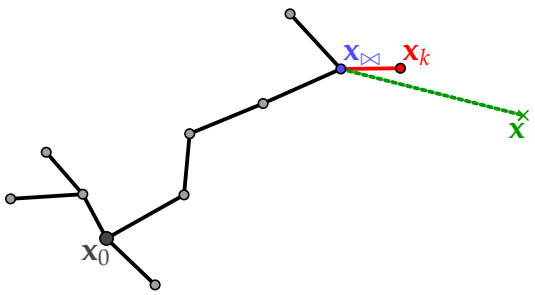
Eine Kante wird jedoch nur dann in $\mathcal{T}$ eingefügt, sofern alle Zustände der Folge im zulässigen Zustandsraum $\mathbb{X} \subseteq \mathcal{X}$ liegen und somit die Folge zulässig ist. Ist die Kantenmenge leer, so ist dieser Zustand momentan nicht erreichbar. Im Kontext der Pfadplanung entspricht dies Hindernissen auf Pfaden. Je nach Algorithmusvariation werden nicht erreichbare Zustände aus der Baumstruktur entfernt oder dazu verwendet die Verteilung $\varsigma(\mathcal{X})$ anzupassen (vgl. [71], [196] und [87]). Der Algorithmus endet nach einer vorgegebenen Iterationsanzahl K. Eine alternative Terminierungsbedingung ist das Erreichen eines zuvor lokalisierten globalen Minimums mit $\mathbf{x}_k \simeq \arg\min_{\mathbf{x}\in\mathcal{X}} \mathfrak{L}$. Das Erreichen kann entweder exakt gefordert werden oder durch eine Schranke bzw. Umgebung relaxiert werden. Letzteres wird vor allem für kontinuierliche Räume verwendet.
Der Vorteil dieses Verfahren liegt darin, dass sehr effizient Zustandsfolgen, in der Robotik als Pfade und in der Regelungstechnik als Trajektorie bezeichnet, vom initialen

Algorithmus 2.1.: RRT Aufbau (vgl. [96])

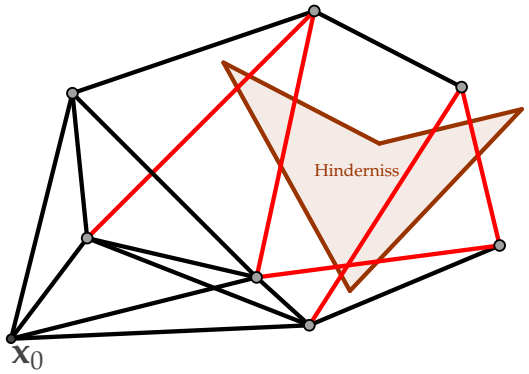
Voraussetzung: $\mathbf{x}_0 \in \mathbb{X}$: zulässiger initialer Zustand, $\mathcal{T}$: RRT-Graph

function BUILDRRT($\mathcal{T}, \mathbf{x}_0$)
  $\mathcal{T} \leftarrow \mathcal{T} \cup \{(\mathbf{x}_0, \emptyset)\}$
  for $k = 1$ **to** K **do**
    $\mathbf{x} \sim \varsigma(\mathcal{X})$ ▷ Stichprobe aus einer Verteilung ziehen bzw. abtasten
    extendRRT($\mathcal{T}, \mathbf{x}$)
  return $\mathcal{T}$

function EXTENDRRT($\mathcal{T}, \mathbf{x}$)
  $(\mathbf{x}_{\triangleright\!\!\triangleleft}, \mathcal{E}) \leftarrow \arg\min_{(\mathbf{x}', \mathcal{E}) \in \mathcal{T}} d(\mathbf{x}, \mathbf{x}')$ ▷ Distanzmetrik d
  $(\mathbf{x}_k, \mathcal{E}_k) \leftarrow$ createStep($(\mathbf{x}_{\triangleright\!\!\triangleleft}, \mathcal{E})$, $\mathbf{x}$)
  $\mathcal{T} \leftarrow \mathcal{T} \cup \{(\mathbf{x}_k, \mathcal{E}_k)\}$
  if $\mathcal{E}_k \neq \emptyset$ **then**
    if $\mathbf{x}_k \simeq \mathbf{x}$ **then**
      return *Reached*
    else
      return *Extended*
  else
    return *Obstacle*



(a) Visualisierung der „extendRRT“ Funktion. Der auf Basis einer Distanzmetrik d naheste Zustand $\mathbf{x}_{\triangleright\!\!\triangleleft}$ zu $\mathbf{x}$ wird bestimmt und von dort aus ein neuer Zustand $\mathbf{x}_k$ bestimmt, der den Graphen $\mathcal{T}$ erweitert.



(b) Visualisierung einer „Probablistic Roadmap“ (kurz PRM). Es werden zufällige Zustände aus einer Verteilung gezogen und mittels eines Verfahrens durch Kanten verbunden. Nicht zulässige Kanten sind rot eingefärbt und werden entfernt. Dies ergibt eine Art *zufällige Straßenkarte*, die mit zunehmender Zustandsanzahl global konvergiert (vgl. [86]).

Abbildung 2.9.: Darstellung der Gemeinsamkeiten von RRT und PRM. Beide Verfahren vernetzen einen Graphen auf ähnliche Weise durch eine Distanzmetrik und filtern unzulässige Kanten heraus.

Zustand $\mathbf{x}_0$ zu jedem Zustand $\mathbf{x}_k$ mit $k = 1, 2, \ldots, K$ berechnet werden können. Allerdings sind diese Trajektorien nicht zwangsläufig optimal, sodass RRT* bei jeder neuen Stichprobe den Graphen $\mathcal{T}$ neu vernetzt und dabei die Kanten optimiert (vgl. [71]). Geht die Stichprobenanzahl gegen unendlich, wird eine optimale modellunabhängige Trajektorie erzeugt, die den initialen Zustand mit dem globalen Minimum mit minimalen kumulierten Kosten verbindet. Die Laufzeit von RRT* ist jedoch höher als die des RRT-Algorithmus. Daher gibt es den „Real Time Rapidly Exploring Random Tree*“ (kurz RT-RRT*) Ansatz, um die Laufzeit zu reduzieren und somit die Aktualisierungsrate der Pfade für mobile Roboter zu erhöhen (vgl. [120]). Weitere Verfahren, wie zum Beispiel RRT+ (vgl. [196]), RRT-A* (vgl. [99]) und „Informed RRT*“ (vgl. [71]), versuchen die Konvergenzgeschwindigkeit durch die Kombination mit anderen Algorithmen oder durch die Information über die Position des globalen Minimums zu erhöhen. Dennoch funktionieren diese Verfahren nur in statischen Umgebungen oder in dynamischen Umgebungen mit großen Zeitkonstanten. Damit verbunden sind geringe Reaktionszeiten der Regler, die einem ausgeprägten Totzeitglied in der Regelungstechnik entsprechen und somit zur Reduktion der Robustheit bis hin zum Stabilitätsverlust führen können. „Probabilistic Roadmaps“ (kurz PRMs) sind sehr ähnlich zu RRT (siehe Abbildung 2.9). Hierbei wird ein Abtastverfahren (siehe Abschnitt 2.2.1) gewählt und anschließend wird eine Stichprobenmenge gezogen, deren Knoten bzw. Stichproben mit ihren nächsten Nachbarn anhand einer Distanzmetrik d verbunden werden. Die Anzahl an Verbindungen kann frei gewählt werden, wird jedoch ebenfalls durch die Zulässigkeit der Folgen beschränkt. Auf dieser „zufälligen Straßenkarte“ kann dann der Dijkstra-Algorithmus angewendet werden, um Pfaden zum Minimum zu bestimmen (vgl. [86]). Im Unterschied zu RRT arbeitet dieser Algorithmus nicht sequenziell. Parallel zu der Entwicklung der RRT-Graphen gibt es die „Expansive Space Trees“ (kurz EST, vgl. [65]), die auf ähnliche Art und Weise arbeiten. Der große Nachteil und Vorteil all dieser Verfahren ist die Regelstreckenunabhängigkeit. Daher muss nach der Generierung der im Zustandsraum zulässigen Trajektorien überprüft werden, ob diese von der Regelstrecke umgesetzt werden können und die entsprechenden Stellgrößen müssen bestimmt werden. Für diesen Berechnungsschritt wird ein Streckenmodell zur Validierung benötigt, weshalb dieser Regelungsansatz, trotz der modellunabhängigen Optimierer, der Klasse der modellbasierten Regelung angehört. Um den Aufwand durch die Validierung und die nachträgliche Bestimmung der Stellgröße mit Hilfe inverser Kinematik zu minimieren, wird die Kantenform in „Spline-based RRT“ angepasst (vgl. [201]), die Distanzmetrik mit dem Modell verschmolzen (vgl. [128]) oder das Modell bei der Wahl der Stichproben beachtet (vgl. [87]). Wird das Modell zur Auswahl des nächsten Zustandes verwendet, so wird dies als modellprädiktive Regelung bezeichnet (siehe Abschnitt 2.2.2).

2.2.1.2. Lokale Pfadplanung

In der mobilen Robotik dienen globale Pfadplaner in der Regel dazu Missionen mit beliebigen Agenten zu lösen, da die Beachtung der Systemdynamik und ihrer Beschränkungen die Komplexität des Problems deutlich erhöhen (vgl. [198] und [114]). Dazu werden die Missionsinformationen in der Kostenfunktion $\mathfrak{L}$ oder in einem Modell

abgebildet. Das Komplement zur globalen Pfadplanung in der Robotik ist die *lokale Pfadplanung*. Diese behandelt vornehmlich die Umwandlung der regelstreckenunabhängigen Trajektorien oder Zwischenzustände der globalen Pfadplaner in streckenabhängige Stellgrößen und Trajektorien unter Berücksichtigung der systemspezifischen Beschränkungen. „Dynamic Window Approach“ (kurz DWA) ist ein laufzeiteffizienter lokaler Planer. Dieser zieht Stichproben nur aus der Menge $\mathbb{U}$ der zulässigen Stellgrößen und prädiziert damit eine festgelegte Anzahl modellabhängiger Kurvensegmente über ein definiertes Zeitfenster. Aus dieser Trajektorienschar wird anschließend mit Hilfe einer Gütefunktion, die die Distanz zum Ziel, die Orientierung zum Ziel und die Distanz zu Hindernissen evaluiert, eine lokal optimale Trajektorie gewählt. Jedoch werden nur Kreissegmente verarbeitet und es wird nur ein Schritt in die Zukunft prädiziert. Danach werden die Stellgrößen konstant gehalten (vgl. [41]).
Eine andere Methode bildet „Time Elastic Band“ (kurz TEB). Diese besteht aus zwei Komponenten. Sie basiert auf einer initialen Trajektorie, die mit einem zeitoptimalen Gütemaß optimiert wird, indem die Zeit minimiert werden soll. Dabei werden Nebenbedingungen, z. B. Verbundenheit der optimierten Trajektorienstücke, Beschränkungen bezüglich der Stellgrößen und der Abstand zu Hindernissen als quadratische Kosten formuliert. Diese werden mit einem Levenberg-Markwardt-Optimierer minimiert. Dadurch sind ebenfalls Stellgrößenverletzungen sowie unverbundene Trajektorienstücke möglich. Dies ist bei DWA und „Control Particle Belief Propagation“ (kurz CPBP, siehe Abschnitt 2.4.2) nicht möglich, da diese Algorithmen die Kinematik zur Prädiktion verwenden. TEB verwendet das „Broyden–Fletcher–Goldfarb–Shanno“ (kurz BFGS) Verfahren und fordert somit die Existenz der ersten und zweiten Ableitung (vgl. [125]). Ferner nutzt es die dünne Besetztheit des Problems aus, um schneller zu konvergieren. Es wird allerdings stets nur eine Trajektorie optimiert. Die Optimalität dieser Methode bezieht sich ausschließlich auf die Zeit und verwendet mitunter suboptimale Approximationen. Ein weiteres Problem dieses Verfahrens bilden die initialen Trajektorien. Diese werden ebenfalls durch Stichproben erzeugt und in Homologieklassen unterteilt, welche ähnlich zu aber nicht so strikt wie Homotopieklassen sind. Anschließend wird jeweils eine Trajektorie aus jeder Klasse parallel optimiert. So kann es sein, dass die Trajektorie zwar zeitoptimal wird, jedoch die initiale Trajektorie suboptimal ist, weil ein kürzerer Pfad in einer anderen Homologieklasse existiert. Weiterhin werden zur Erzeugung der initialen Trajektorie die Zustandsstichproben ausschließlich vorwärts orientiert in Richtung eines gegebenen Ziels verbunden, sodass es einfache Unzulässigkeitskonfigurationen gibt, bei denen keine initialen Trajektorien vom Start zum Ziel erstellt werden können. Aufgrund dessen ist dieser Ansatz stark von der vorgelagerten globalen Pfadplanung abhängig, die durch die Konfiguration navigieren muss. Damit kann die lokale Planung auf realisierbare Trajektorienstücke verkleinert werden (vgl. [158]). Gleichwohl der genannten Nachteile erzielt TEB deutlich bessere Ergebnisse als DWA (vgl. [158]).
Die Weiterentwicklung des TEB-Verfahrens verwendet eine modellprädiktive Regelung statt des Levenberg-Markwardt-Algorithmus und einen quadratischen Kostenterm statt des zeitoptimalen Gütemaßes, sodass dieses nun konfigurierbar ist. Die Gradienten werden auf Basis eines Hypergraphen berechnet, der die Konvergenzgeschwindigkeit erhöht. Weiterhin werden harte Beschränkungen (Englisch: „Hard

Constraints“) ermöglicht, die die Laufzeit jedoch stark erhöhen. Auch eine asymptotischen Stabilität (vgl. [50]) kann garantiert werden, sofern eine Endkostenabschätzung, z. B. in Form eines Zielzustandes, bereitgestellt werden kann.
Allgemein verwenden lokale Pfadplaner hauptsächlich Vorwärtskinematiken und haben somit eine geringere Rechenzeit als bei der Verwendung inverser Kinematiken. Einfache aber laufzeiteffiziente Optimierungsheuristiken werden ebenfalls erforscht, erzielen jedoch oft nur eine geringe Regelungsgüte. Die Aufteilung in lokale und globale Pfadplaner bietet den Vorteil einer modularen Struktur und des einfachen Austausches der Regelungsalgorithmen (vgl. [114]). Dies ist auch gleichzeitig nachteilig, weil die globalen Pfadplaner missionsbasierte Trajektorien generieren, die von dem System bzw. dem lokalen Pfadplaner nicht umgesetzt werden können. Andersherum erzeugen die lokalen Pfadplaner ausführbare Trajektorien, die allerdings nicht unter Beachtung der Missionen und deren Beschränkungen berechnet werden. Durch den fehlenden Informationsaustausch können suboptimale oder unzulässige Trajektorien entstehen.

2.2.1.3. Evolutionäre Algorithmen

Die Verschmelzung von lokalem und globalem Pfadplaner ist bereits insbesondere für die Luftfahrt und die Regelung von Flugdrohnen erforscht worden (vgl. [34], [152], [177], [198], [26] und [76]), jedoch werden evolutionäre Algorithmen zur Optimierung verwendet. Diese Methoden sind gute Optimierer für allgemeine Probleme, deren Struktur unbekannt ist (Englisch: „Blackbox Optimization“). Deren Laufzeit ist allerdings für oberflächenbezogene Systeme in dynamischen Umgebungen mit im Vergleich zur Luftfahrt vielen Hindernissen nicht anwendbar (vgl. [90] und [197]), zumal die Struktur der Probleme und Systeme bekannt ist oder approximiert werden kann.
Weitere Verfahren werden im Abschnitt 2.5 direkt mit dem verwendeten CPBP bzw. „Advanced Control Particle Belief Propagation“ Verfahren verglichen.

2.2.2. Modellprädiktive Regelung

Bezugnehmend auf die Forschungsziele (siehe Abschnitt 1.2) wird eine modellprädikitve Regelung (Englisch: „Model Predictive Control“, kurz MPC) entwickelt, die die Trennung zwischen lokaler (siehe Abschnitt 2.2.1.2) und globaler Pfadplanung (siehe Abschnitt 2.2.1.1) auflöst und somit zulässige Trajektorien mit einer hohen Güte berechnet. Dafür wird versucht den Großteil der Missionskomplexität in dynamische Kostenfunktionen zu verschieben (siehe Kapitel 5). Ziel ist es, autonome Agenten in unbekannten Umgebungen unter Beachtung der Missions- bzw. Regelungsziele implizit durch die Angabe von begünstigenden und erschwerenden Faktoren zu regeln. Dies entspricht einer mehrkriteriellen, gemischt ganzzahligen, mehrdimensionalen Optimierung mit einer großen Anzahl linearer und nichtlinearer Nebenbedingungen, die modellspezifisch sind. Die Optimierungsumgebung ist hierbei veränderlich, nicht nur in Abhängigkeit von der Zeit (siehe Kapitel 5). Dieses Optimierungsproblem besitzt in der Regelungstechnik bekannte Strukturen und die Vielzahl der Nebenbedingungen besitzen spezielle Abhängigkeiten (vgl. [50]). Daraus folgt, dass ein hybrider Ansatz

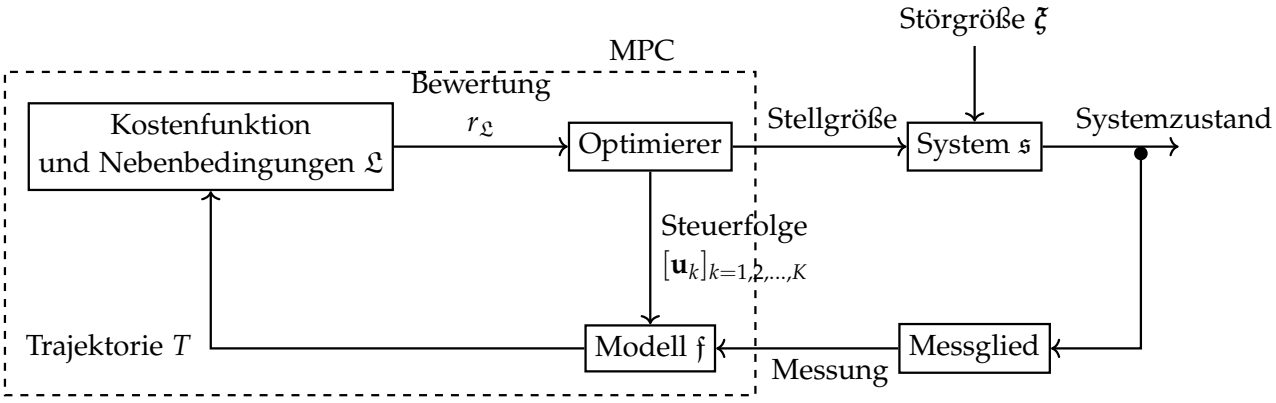


Abbildung 2.10.: Strukturbild einer modellprädiktiven Regelung.

aus Optimierung und Regelung angewendet wird. Diese Technik verbindet modellbasierte Optimierung mit einer Regelungsschleife (siehe Abbildung 2.10). Die Abbildung 2.10 stellt die Hauptkomponenten der Regelung dar. Das Stellglied, das System und das Messglied in Form von Sensoren wird in Kapitel 6 behandelt und hier zunächst als ideal bzw. identisch zum Modell angenommen. Zur Approximation des realen Systems wird die mathematische Systembeschreibung $\mathfrak{s}$ mit normalverteiltem Rauschen $\boldsymbol{\xi} \sim \mathcal{N}(0, \mathbf{Q})$ gestört. Die MPC berechnet mit Hilfe einer Modellfunktion $\mathfrak{f}$ und einer Stellgrößenfolge $[\mathbf{u}_k]_{k=1,2,\ldots,K}$ Systemzustände $\mathbf{x}_k \in \mathbb{X}_{\mathbf{x}_0}(t_k)$ mit $k = 1, 2, \ldots, K$ aus der Menge der zeitabhängigen zulässigen Zustände $\mathbb{X}_{\mathbf{x}_0}(t_k)$, die des Weiteren von einem initial Zustand $\mathbf{x}_0$ des Systems abhängen. Die Menge der zulässigen Zustände bildet eine Teilmenge des gesamten Zustandsraumes $\mathcal{X}$, somit gilt $\mathbb{X}_{\mathbf{x}_0}(t_k) \subseteq \mathcal{X}$. Die resultierende Trajektorie $T = [\mathbf{x}_k \mathbf{u}_k]_{k=1,2,\ldots,K}$, bestehend aus Zuständen und den zugehörigen Stellgrößen, wird durch Kosten- und Nebenbedingungsfunktionen $\mathfrak{L}$ bewertet werden. Aufgrund dieser Bewertung $r_{\mathfrak{L}}$ erzeugt ein Optimierungsalgorithmus eine neue Stellgrößenfolge $[\mathbf{u}_k]_{k=1,2,\ldots,K} \in \mathbb{U}_{\mathbb{X}_{\mathbf{x}_0}(t_k)}$ aus der Menge der zulässigen Stellgrößen, die vom jeweiligen Zustand abhängig ist. Ebenso ist die Menge der zulässigen Stellgrößen $\mathbb{U}_{\mathbb{X}_{\mathbf{x}_0}(t_k)}$ eine Teilmenge der Stellgrößenmenge $\mathcal{U}$; daher gilt $\mathbb{U}_{\mathbb{X}_{\mathbf{x}_0}(t_k)} \subseteq \mathcal{U}$. Mit Hilfe dieser Stellgrößenfolge und des Modells werden erneut Systemzustände prädiziert. Diese *innere Schleife* iteriert so lange, bis der Optimierungsalgorithmus zu einem Minimum $r_{\mathfrak{L}}^{\sim} = \min \mathfrak{L}$ konvergiert oder ein Zeitkontingent t_K erschöpft ist. Anschließend wird das erste Element der optimalen Steuerfolge $\tilde{\mathbf{u}}_1$ als Stellgröße mittels eines Stellglieds auf das reale System $\mathfrak{s}$ angewendet. Der ermittelte Messwert $\check{\mathbf{x}}_k$ des Systemzustandes wird zur Initialisierung des initialen Zustandes $\mathbf{x}_0$ und zur Verbesserung oder zur Anpassung des Modells verwendet. Daraufhin beginnt die *innere Schleife* erneut mit der Berechnung einer optimalen Stellgröße für das reale System. Somit ist der Regelkreis geschlossen. Für eine detaillierte Beschreibung der modellprädiktiven Regelung und ihrer Eigenschaften wird auf das Buch von Grüne und Pannek [50] verwiesen. In den folgenden Abschnitten wird auf die wesentlichen Bestandteile der Modellierung (vgl. Abschnitt 2.3) und Optimierung (vgl. Abschnitt 2.4.2) eingegangen. Die Bewertung, die das Missionsverhalten abbildet, wird aufgrund des Umfangs in Kapitel 5 erläutert und die Verwendung von Beobachtern zur Rekonstruktion des vollständigen

Systemzustandes **x** wird zusammen mit der Messung in Kapitel 6 behandelt.

2.3. Nichtlineare Modelle der Regelstrecke

Bei der Bestimmung der Übertragungsfunktion der Regelstrecke (siehe Kapitel 6), auch Auswahl des Streckenmodells genannt, ist stets zwischen Modellgenauigkeit und Modellkomplexität abzuwägen. In der Regel besitzen reale physikalische Systeme in der Umwelt eine hohe Komplexität und Nichtlinearität, die zu einer langen Berechnungszeit führen (vgl. [177]). Um die Berechnungszeit zu reduzieren, werden die Systemmodelle abstrahiert und reduziert in Form von Approximationen. Typische Approximationen sind Linearisierungen um den Arbeitspunkt des Systems herum, wodurch optimale und effiziente Regler mit reduzierter Laufzeit verwendet werden können (vgl. [3]). Jedoch nimmt die Regelgüte und die Stabilität des Regelkreises mit zunehmender Approximation ab, sodass stets anhand der geforderten Regelgüte und der Laufzeitschranke aufgrund der Systemdynamik und des Abtasttheorems (siehe Anhang C.2) abgewogen werden muss. Die in dieser Arbeit entwickelte MPC ist optimiert für folgende Oberflächenfahrzeugtypen:

1. Kettenfahrzeug und agile Schreitrobotik (siehe 2.3.1)
2. Schiffe mit starrer Welle oder Außenbordmotor (siehe 2.3.2)
3. Kraftfahrzeuge mit Ackermann-Lenkung (siehe 2.3.3)

Diese Fahrzeuge weisen allesamt ein nichtlineares und nichtholonomes Verhalten auf, außer den agilen Schreitroboter. Diese können auch vollständig holonom betrieben werden, wobei jedoch die seitliche Bewegungsgeschwindigkeit deutlich geringer ist als die der Hauptrichtungen vorwärts, rückwärts und Rotation. Die seitliche Bewegung dient bei diesen Systemen hauptsächlich der Stabilisierung durch Ausgleichsbewegungen und Gewichtsverlagerung. Weiterhin agieren alle Fahrzeugtypen auf Oberflächen und können sich in der dritten Dimension nur vernachlässigbar geringfügig bewegen. Damit lässt sich ein gemeinsames nichtlineares Modell identifizieren, welches von jeder Gattung weiter spezifiziert wird.

Allgemein lässt sich ein kontinuierlicher Zustandsübergang von $\overset{\circ}{\mathbf{x}}(t_0)$ zu $\overset{\circ}{\mathbf{x}}(t_0+t)$ über die Zeit t wie folgt definieren:

$$\overset{\circ}{\mathbf{x}}(t_0+t) = \int_{t_0}^{t_0+t} \overset{\circ}{\mathbf{f}}(\overset{\circ}{\mathbf{x}}(\tau), \overset{\circ}{\mathbf{u}}(\tau), \tau) d\tau \tag{2.3.1}$$

Dabei bezeichnet ein Kreis über den Variablen stets eine kontinuierliche Größe und eine Raute eine diskrete Größe. Mit Hilfe von $k = 1, 2, \ldots, K$ diskreten Zeitschritten $\overset{\diamond}{\Delta t}(k) = \overset{\diamond}{\Delta t}_k$ lässt sich die kontinuierliche Gleichung approximieren.

$$\overset{\diamond}{\mathbf{x}}(k) = \overset{\diamond}{\mathbf{f}}(\overset{\diamond}{\mathbf{x}}(k-1), \overset{\diamond}{\mathbf{u}}(k), \overset{\diamond}{\Delta t}(k)) = \overset{\diamond}{\mathbf{x}}_k \tag{2.3.2}$$

Dabei besteht ein Zustand $\overset{\diamond}{\mathbf{x}}$ aus drei Elementen, die zwei Positionskoordinaten in $\mathbb{R}$ entsprechen und einer Ausrichtung im Bogenmaß in $\Pi = [0, 2\pi)$. Die Stellgröße $\overset{\diamond}{\mathbf{u}} \in \mathbb{R}^2$ ist systemspezifisch und daher nicht weiter festgelegt.

$$\begin{bmatrix} \overset{\diamond}{\mathbf{x}}_{[1]} \\ \overset{\diamond}{\mathbf{x}}_{[2]} \\ \overset{\diamond}{\mathbf{x}}_{[3]} \end{bmatrix} = \overset{\diamond}{\mathbf{x}} \in \mathcal{X} = \mathbb{R} \times \mathbb{R} \times \Pi \tag{2.3.3}$$

$$\begin{bmatrix} \overset{\diamond}{\mathbf{u}}_{[1]} \\ \overset{\diamond}{\mathbf{u}}_{[2]} \end{bmatrix} = \overset{\diamond}{\mathbf{u}} \in \mathcal{U} = \mathbb{R} \times \mathbb{R} \tag{2.3.4}$$

$$\tag{2.3.5}$$

Jedoch sind sowohl die Stellgrößen vollständig als auch der Zustandsraum teilweise beschränkt, sodass eine Übertragungsfunktion $\overset{\diamond,\dashv}{\mathrm{f}}$ unter Berücksichtigung dieser Beschränkungen bestimmt wird. Beschränkungen werden stets mit $\dashv$ gekennzeichnet. Nun lässt sich eine Modellfunktionsvorlage angeben:

$$\begin{aligned} \overset{\diamond,\dashv}{\mathbf{x}}_k &= \overset{\diamond,\dashv}{\mathrm{f}}(\overset{\diamond,\dashv}{\mathbf{x}}_{k-1}, \overset{\diamond,\dashv}{\mathbf{u}}_k, \overset{\diamond}{\Delta t_k}) = \\ \overset{\diamond,\dashv}{\mathbf{x}}_k &= \overset{\diamond,\dashv}{\mathbf{x}}_{k-1} \oplus \overset{\diamond,\dashv}{\overset{\mapsto}{\mathbf{x}}}_k(\overset{\diamond,\dashv}{\mathbf{u}}_k) \cdot \overset{\diamond}{\Delta t_k} \end{aligned} \tag{2.3.6}$$

$\overset{\diamond,\dashv}{\mathbf{x}}_k$ beschreibt einen beschränkten Folgezustand ausgehend von $\overset{\diamond,\dashv}{\mathbf{x}}_{k-1}$ unter Applikation eines Zustandsinkrements $\overset{\diamond,\dashv}{\overset{\mapsto}{\mathbf{x}}}_k(\overset{\diamond,\dashv}{\mathbf{u}}_k)$, abhängig von der Stellgröße $\overset{\diamond,\dashv}{\mathbf{u}}_k$, über einen Zeitraum $\overset{\diamond}{\Delta t_k}$. Die Applikationsoperation $\oplus$ ist für die Größen aus $\mathbb{R}$ identisch zur Addition und für die Ausrichtung entspricht sie der Addition der Orientierungen mit anschließender Restwertbildung über 2π.

$$x \oplus y = x + y, \qquad \text{für } x, y \in \mathbb{R} \tag{2.3.7}$$

$$x \oplus y = (x + y)\%2\pi, \qquad \text{für } x, y \in \Pi \tag{2.3.8}$$

2.3.1. Kettenfahrzeuge und agile Schreitrobotik

Diese unbemannten Bodenfahrzeuge haben weniger Freiheitsgrade als Luftfahrzeuge, jedoch befinden sich in dem Gelände der betrachteten Fahrzeuge meistens deutlich mehr Hindernisse als im Luftraum. Des Weiteren sind die Möglichkeiten zur Umgehung statischer und dynamischer Hindernisse geringer, sodass die Planung erschwert wird. Es wird ein einfaches diskretes, nichtlineares Modell für nichtholonome Systeme angenommen, das die Fähigkeit auf der Stelle zu drehen abbildet (siehe Abbildungen

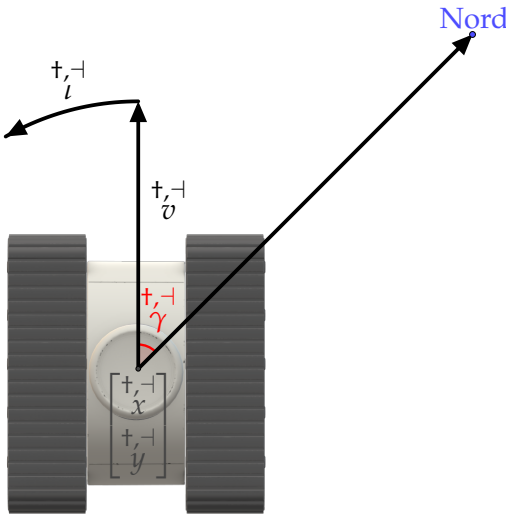


Abbildung 2.11.: Modell eines Kettenfahrzeugs (vgl. [141]).

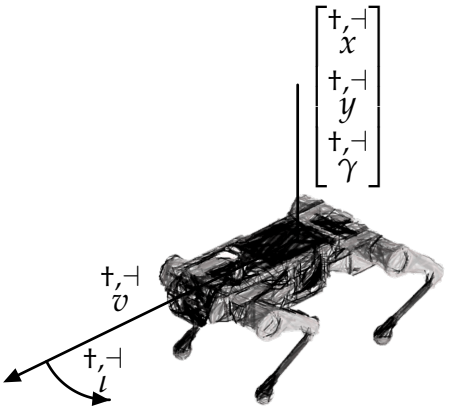


Abbildung 2.12.: Modell eines Schreitroboters (vgl. [141]).

2.11 und 2.12).

$$\overset{\diamond,\dashv}{\mathbf{x}}_k = \overset{\diamond,\dashv}{\mathbf{f}}\left(\overset{\diamond,\dashv}{\mathbf{x}}_{k-1}, \overset{\diamond,\dashv}{\mathbf{u}}_k, \overset{\diamond}{\Delta t}_k\right) = \tag{2.3.9}$$

$$\underbrace{\begin{bmatrix} \overset{\dagger,\dashv}{x}_k \\ \overset{\dagger,\dashv}{y}_k \\ \overset{\dagger,\dashv}{\gamma}_k \end{bmatrix}}_{\overset{\diamond,\dashv}{\mathbf{x}}_k} = \underbrace{\begin{bmatrix} \overset{\dagger,\dashv}{x}_{k-1} \\ \overset{\dagger,\dashv}{y}_{k-1} \\ \overset{\dagger,\dashv}{\gamma}_{k-1} \end{bmatrix}}_{\overset{\diamond,\dashv}{\mathbf{x}}_{k-1}} \oplus \underbrace{\begin{bmatrix} \overset{\dagger,\dashv}{v}_k \cdot \sin(\overset{\dagger,\dashv}{\gamma}_{k-1}) \\ \overset{\dagger,\dashv}{v}_k \cdot \cos(\overset{\dagger,\dashv}{\gamma}_{k-1}) \\ \overset{\dagger,\dashv}{\iota}_k \end{bmatrix}}_{\overset{\mapsto}{\overset{\diamond,\dashv}{\mathbf{x}}}_k} \cdot \overset{\diamond}{\Delta t}_k$$

$$\text{mit } \overset{\diamond,\dashv}{\mathbf{u}}_k = \begin{bmatrix} \overset{\dagger,\dashv}{v}_k \\ \overset{\dagger,\dashv}{\iota}_k \end{bmatrix} \tag{2.3.10}$$

$\overset{\dagger,\dashv}{x}_k \in \mathbb{R}$ und $\overset{\dagger,\dashv}{y}_k \in \mathbb{R}$ seien die Koordinaten bzw. die Positionen in der Ebene und $\overset{\dagger,\dashv}{\gamma}_k \in \Pi$ sei der Winkel zu einem Meridian bzw. zur x-Achse. Dabei deutet † darauf hin, dass die Größen als Koordinaten eines Referenzsystems zu verstehen sind. Die Begrenztheit der jeweiligen Größen wird immer mit $\dashv$ gekennzeichnet. Weiterhin bezeichne $\overset{\dagger,\dashv}{v}_k \in \mathbb{R}$ die beschränkte Vortriebsgeschwindigkeit und $\overset{\dagger,\dashv}{\iota}_k \in \mathbb{R}$ die ebenfalls beschränkte Rotationsgeschwindigkeit. Dieses Modell erster Ordnung ist allgemein gehalten, um möglichst viele Fahrzeuge dieser Gattungen abzudecken (siehe Abbildungen 2.11 und 2.12). Die Modellierung der Kinematik einzelner Fahrzeuge würde über kaskadierte Verfahren, die die Stellgrößen des Reglers als Führungsgröße annehmen, erfolgen. Zudem muss eine höhere Modellkomplexität stets gegen die damit verbundenen steigenden Berechnungszeiten abgewogen werden, da eine suboptimale Reglung ausreichend ist.

2.3.1.1. Aglie Schreitrobotik

Agile Schreitroboter regeln grundlegend unterschiedlich zu Schreitrobotern, die auf Schrittplanung aufbauen. Letztere versuchen stets in einem stabilen Zustand zu sein und berechnen eine Transition von einem stabilen Zustand in den Nächsten. Dieses Reglerverhalten weist lange Laufzeiten auf und eignet sich daher nur für geringe Geschwindigkeiten. Dem gegenüber besitzen agile Schreitroboter eine gewisse Instabilität als Präferenz, denn die Annahme einer konstanten Fallbewegung ermöglicht die Approximation kinematischer partieller Differentialgleichungen und mündet in kontinuierlichen Ausgleichsbewegungen. Stillstehen entspricht bei dieser Regelung dem Fallen in undefinierte Richtungen, wodurch die Komplexität zur Lösung erhöht wird. Die Lösung der approximierten Gleichungen vereinfacht sich stark bei konstanter Bewegungsgeschwindigkeit in eine konstante Richtung, sodass die Modellgleichung verwendet werden kann.

Abbildung 2.13.: 3D-Modell eines Frachtschiffes.

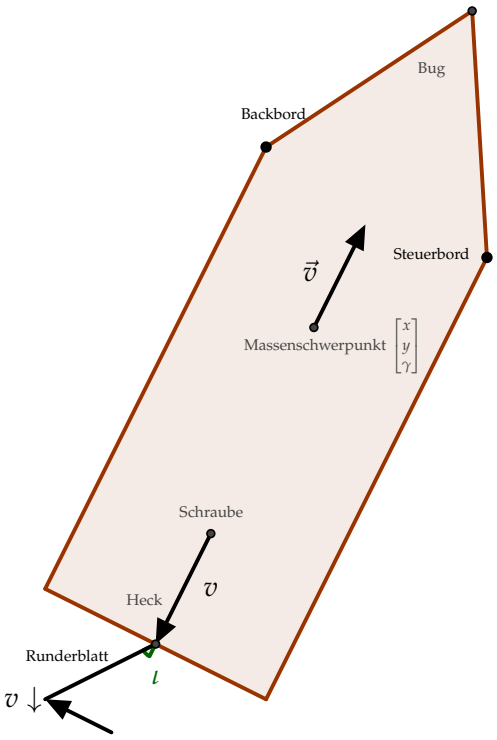


Abbildung 2.14.: Modell eines Schiffes mit starrer Welle.

2.3.2. Wasserfahrzeuge

Das Modell der Gattung der Wasserfahrzeuge deckt Schiffe mit Außenbordantrieb und Schiffe mit Ruder und starrer Welle ab (siehe Abbildung 2.14).

$$\overset{\diamond,\dashv}{\mathbf{x}}_k = \overset{\diamond,\dashv}{\mathfrak{f}}(\overset{\diamond,\dashv}{\mathbf{x}}_{k-1}, \overset{\diamond,\dashv}{\mathbf{u}}_k, \overset{\diamond}{\Delta t}_k) = \tag{2.3.11}$$

$$\underbrace{\begin{bmatrix} \overset{\dagger,\dashv}{x}_k \\ \overset{\dagger,\dashv}{y}_k \\ \overset{\dagger,\dashv}{\gamma}_k \end{bmatrix}}_{\overset{\diamond,\dashv}{\mathbf{x}}_k} = \underbrace{\begin{bmatrix} \overset{\dagger,\dashv}{x}_{k-1} \\ \overset{\dagger,\dashv}{y}_{k-1} \\ \overset{\dagger,\dashv}{\gamma}_{k-1} \end{bmatrix}}_{\overset{\diamond,\dashv}{\mathbf{x}}_{k-1}} \oplus \underbrace{\begin{bmatrix} \overset{\dagger,\dashv}{\vec{v}}_k \cdot \sin(\overset{\dagger,\dashv}{\gamma}_{k-1}) \\ \overset{\dagger,\dashv}{\vec{v}}_k \cdot \cos(\overset{\dagger,\dashv}{\gamma}_{k-1}) \\ \overset{\dagger,\dashv}{v}\downarrow_k \cdot \frac{1}{r_{\mapsto \odot_m}} \end{bmatrix}}_{\overset{\mapsto}{\overset{\diamond,\dashv}{\mathbf{x}}}_k} \cdot \overset{\diamond}{\Delta t}_k$$

$$\text{mit } \overset{\diamond,\dashv}{\mathbf{u}}_k = \begin{bmatrix} \overset{\dagger,\dashv}{v}_k \\ \overset{\dagger,\dashv}{\iota}_k \end{bmatrix}, \ \overset{\dagger,\dashv}{\vec{v}}_k = \overset{\dagger,\dashv}{v}_k \cdot \cos(\overset{\dagger,\dashv}{\iota}_k), \ \overset{\dagger,\dashv}{v}\downarrow_k = \overset{\dagger,\dashv}{v}_k \cdot \sin(\overset{\dagger,\dashv}{\iota}_k) \tag{2.3.12}$$

$\overset{\dagger,\dashv}{x}_k \in \mathbb{R}$ und $\overset{\dagger,\dashv}{y}_k \in \mathbb{R}$ seien die Koordinaten bzw. die Positionen in der Ebene und $\overset{\dagger,\dashv}{\gamma}_k \in \Pi$ sei der Winkel zu einem Meridian bzw. zur x-Achse. Ferner bezeichne $\overset{\dagger,\dashv}{v}_k \in \mathbb{R}$ erneut die beschränkte Vortriebsgeschwindigkeit. Auch gilt, dass $\dagger$ eine Koordinaten eines Referenzsystems bedeutet und $\dashv$ ebenfalls die Begrenztheit der jeweiligen Größen kennzeichnet. $\overset{\dagger,\dashv}{\iota} \in \mathbb{R}$ bezeichnet bei diesem Modell allerdings den beschränkten Lenkwinkel des Ruderblattes bzw. den Anstellwinkel des Außenbordmotors. Die Vortriebsgeschwindigkeit $\overset{\dagger,\dashv}{v}_k$ teilt sich abhängig vom Lenkwinkel $\overset{\dagger,\dashv}{\iota}_k$ in einen in die aktuelle Richtung $\overset{\dagger,\dashv}{\gamma}_k$ vortreibenden Anteil $\overset{\dagger,\dashv}{\vec{v}}_k$ und einen quertreibenden Anteil $\overset{\dagger,\dashv}{v}\downarrow_k$. Der quertreibende Anteil bewirkt eine Rotation abhängig vom Abstand $r_{\mapsto \odot_m}$ des Ruders von dem Massenschwerpunkt $\odot_m$ des Schiffes, welcher gleichzeitig den Drehpunkt und die Position des Schiffes beschreibt (siehe Abbildung 2.14). Der wesentliche Unterschied zum vorherigen Modell der Kettenfahrzeuge (siehe Abschnitt 2.3.1) besteht darin, dass der Antrieb nicht am Massenschwerpunkt ansetzt und dass die Rotation durch die Lenkung nicht unabhängig vom Vortrieb geschehen kann. Dadurch sind die Orientierung und die Position verschränkt, was zu einer erhöhten Lösungskomplexität führt. Dadurch neigen diese Systeme eher zur Instabilität.

2.3.3. Ackermannlenkung für Landfahrzeuge

Kraftfahrzeuge mit Ackermann-Lenkung (vgl. [189] und [205]) verfügen über eine starre Antriebsachse und eine Lenkachse. Gemeinsam mit dem Modell der Wasserfahrzeuge und im Gegensatz zu den Kettenfahrzeugen ist diese Gattung nicht in der Lage auf der Stelle zu wenden.

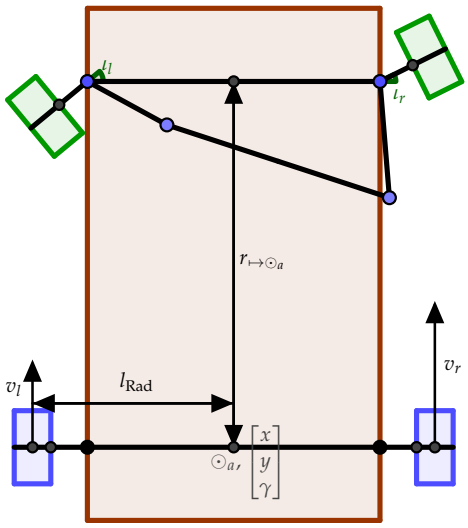


Abbildung 2.15.: Modell eines Kraftfahrtzeuges mit Vorderradlenkung und differentiellem Heckantrieb. Die Vortriebsgeschwindigkeit v ermittelt sich aus dem Mittelwert der einzelnen Radgeschwindigkeiten (vgl. [157]).

$$v = \frac{1}{2}(v_r + v_l) \tag{2.3.13}$$

$$r = \frac{r_{\mapsto\odot_a}}{\tan(\iota)} \tag{2.3.14}$$

$$\iota_l = \arctan\left(\frac{r_{\mapsto\odot_a}}{r - l_{\text{Rad}}}\right) \tag{2.3.15}$$

$$\iota_r = \arctan\left(\frac{r_{\mapsto\odot_a}}{r + l_{\text{Rad}}}\right) \tag{2.3.16}$$

$$\overset{\diamond,\dashv}{\mathbf{x}}_k = \overset{\diamond,\dashv}{\mathbf{f}}\left(\overset{\diamond,\dashv}{\mathbf{x}}_{k-1}, \overset{\diamond,\dashv}{\mathbf{u}}_k, \overset{\diamond}{\Delta t}_k\right) = \tag{2.3.17}$$

$$\underbrace{\begin{bmatrix} \overset{\dagger,\dashv}{x}_k \\ \overset{\dagger,\dashv}{y}_k \\ \overset{\dagger,\dashv}{\gamma}_k \end{bmatrix}}_{\overset{\diamond,\dashv}{\mathbf{x}}_k} = \underbrace{\begin{bmatrix} \overset{\dagger,\dashv}{x}_{k-1} \\ \overset{\dagger,\dashv}{y}_{k-1} \\ \overset{\dagger,\dashv}{\gamma}_{k-1} \end{bmatrix}}_{\overset{\diamond,\dashv}{\mathbf{x}}_{k-1}} \oplus \underbrace{\begin{bmatrix} \overset{\dagger,\dashv}{v}_k \cdot \sin(\overset{\dagger,\dashv}{\gamma}_{k-1}) \\ \overset{\dagger,\dashv}{v}_k \cdot \cos(\overset{\dagger,\dashv}{\gamma}_{k-1}) \\ \overset{\dagger,\dashv}{v}_k \cdot \frac{\tan(\overset{\dagger,\dashv}{\iota}_k)}{r_{\mapsto\odot_a}} \end{bmatrix}}_{\overset{\mapsto}{\overset{\diamond,\dashv}{\mathbf{x}}_k}} \cdot \overset{\diamond}{\Delta t}_k$$

$$\text{mit } \overset{\diamond,\dashv}{\mathbf{u}}_k = \begin{bmatrix} \overset{\dagger,\dashv}{v}_k \\ \overset{\dagger,\dashv}{\iota}_k \end{bmatrix} \tag{2.3.18}$$

Auch bei diesem Modell seien $\overset{\dagger,\dashv}{x}_k \in \mathbb{R}$ und $\overset{\dagger,\dashv}{y}_k \in \mathbb{R}$ die Koordinaten in der Ebene und $\overset{\dagger,\dashv}{\gamma}_k \in \Pi$ sei der Winkel zu einem Meridian. Erneut bildet $\overset{\dagger,\dashv}{v}_k \in \mathbb{R}$ die beschränkte Vortriebsgeschwindigkeit und $\iota_k \in \mathbb{R}$ den Lenkwinkel aus der Fahrzeugflucht heraus ab. † bedeutet, dass die Größen als Koordinaten eines Referenzsystems zu verstehen sind. Die Begrenztheit der jeweiligen Größen wird mit $\dashv$ gekennzeichnet. Dieser Lenkwinkel wird mittels Gleichungen 2.3.16 und 2.3.15 in die jeweiligen Radauslenkungen der Ackermannlenkung übersetzt. Ebenfalls wird die Vortriebsgeschwindigkeit durch ein Differential auf die hinteren Antriebsräder mittels Gleichung 2.3.13 aufgeteilt. Das Rotationsinkrement $\overset{\dagger,\dashv}{v}_k \cdot \frac{\tan(\overset{\dagger,\dashv}{\iota}_k)}{r_{\mapsto\odot_a}}$ ist von dem konstanten Achsabstand $r_{\mapsto\odot_a}$ zwischen der Lenkachse und der Antriebsachse $\odot_a$ abhängig (siehe Abbildung 2.15). Dieses Systems weist ebenfalls eine Verschränkung der Position mit der Ausrichtung bzw.

Orientierung auf. Allerdings ist der Achsabstand im Allgemeinen um Größenordnungen geringer als der Abstand des Ruderblatts vom Massenschwerpunkt eines Schiffes und die Wirkung der Lenkung ist anders als bei Schiffen. Daher ist die Komplexität des Systemmodells zwischen dem der Kettenfahrzeuge und der Schiffe einzuordnen.

2.4. Optimierung

Dieser Abschnitt präsentiert das aus den vorherigen Bausteinen resultierende Optimalsteuerungsproblem (vgl. [50]) sowie die untersuchten Verfahren zur Lösung dieses. Weiterhin werden alternative Lösungsmöglichkeiten aufgezeigt, die aufgrund der Anforderungen an die Missionen und somit an die Kostenfunktion nur sehr eingeschränkt nutzbar sind. Die Optimierungsverfahren werden detailliert erörtert und im folgenden Unterkapitel 2.5 empirisch untersucht.

2.4.1. Optimalsteuerungsproblem

Aufbauend auf der MPC (siehe Abschnitt 2.2.2) und einem Modell (siehe Unterkapitel 2.3) lässt sich das vollständige diskrete Minimierungsproblem für einen Agenten bzw. Roboter $[r_k]_{k=1,2,\dots,K} = \mathfrak{r} \in \mathfrak{R}$ eines Schwarmes bzw. Menge an Agenten $\mathfrak{R} = [\mathcal{R}_k]_{k=1,2,\dots,K}$ aufstellen. Diese Problem wird in der Regelungstechnik als Optimalsteuerungsproblem definiert (vgl. [107]), weil die Lösung des Problems eine optimale Steuerfolge erzeugt, die auf das System gegeben werden kann.

$$\min_{[\overset{\diamond,\dashv^{r_k}}{\mathbf{u}}{}_k]_{k=1,2,\dots,K}} \mathfrak{L}^{\mathfrak{r}}(T^{\mathfrak{r}}_{\overset{\diamond,\dashv^{\mathfrak{r}}}{\mathbf{x}}{}_0}, \mathfrak{R}, \mathfrak{t}, \Delta\mathfrak{t}) = \sum_{k=1}^{K} L^{r_k}(\overset{\diamond,\dashv^{\mathfrak{r}}}{\mathbf{x}}{}_0, \overset{\diamond,\dashv^{r_k}}{\mathbf{x}}{}_k, \overset{\diamond,\dashv^{r_k}}{\mathbf{u}}{}_k, \mathcal{R}_k, t_k, \overset{\diamond}{\Delta t}_k) \tag{2.4.1}$$

mit Nebenbedingungen

$$K \in \mathbb{N} \tag{2.4.2}$$

$$\overset{\diamond,\dashv^{r_k}}{\mathbf{x}}{}_0 \in \mathbb{X}(t_0) \subseteq \mathcal{X} \tag{2.4.3}$$

$$t_0 \in \mathbb{R}_0^+ \tag{2.4.4}$$

$$\overset{\diamond,\dashv^{r_k}}{\mathbf{x}}{}_k \in \mathbb{X}_{\overset{\diamond,\dashv^{\mathfrak{r}}}{\mathbf{x}}{}_0}(t_k) \subseteq \mathcal{X} \tag{2.4.5}$$

$$\overset{\diamond,\dashv^{r_k}}{\mathbf{u}}{}_k \in \mathbb{U}_{\mathbb{X}_{\overset{\diamond,\dashv^{\mathfrak{r}}}{\mathbf{x}}{}_0}(t_k)} \subseteq \mathcal{U} \tag{2.4.6}$$

$$[r_k]_{k=1,2,\dots,K} = \mathfrak{r} \in \mathfrak{R} = [\mathcal{R}_k]_{k=1,2,\dots,K} \tag{2.4.7}$$

$$[{}^{\diamond,\dashv^{r_k}}\mathbf{x}_k \; {}^{\diamond,\dashv^{r_k}}\mathbf{u}_k]_{k=1,2,\dots,K} = T^{\mathfrak{r}} \in (\mathcal{X} \times \mathcal{U})^K \tag{2.4.8}$$

$$[t_k]_{k=1,2,\dots,K} = \mathfrak{t} \subseteq \mathbb{R}_0^{+K} \tag{2.4.9}$$

$$[\overset{\diamond}{\Delta} t_k]_{k=1,2,\dots,K} = \Delta\mathfrak{t} \subseteq \mathbb{R}_0^{+K} \tag{2.4.10}$$

$$\forall k, j \in \{1, 2, \dots, K\}. \forall t_k, t_j \in \mathfrak{t}.\; k < j \Leftrightarrow t_k < t_j \tag{2.4.11}$$

$$0 \leq \overset{\diamond}{\Delta} t_k = t_k - t_{k-1} \tag{2.4.12}$$

$${}^{\diamond,\dashv^{r_k}}\mathbf{x}_k = {}^{\diamond,\dashv}\mathfrak{f}({}^{\diamond,\dashv^{r_k}}\mathbf{x}_{k-1}, {}^{\diamond,\dashv^{r_k}}\mathbf{u}_k, \overset{\diamond}{\Delta} t_k) \tag{2.4.13}$$

Sei $\mathfrak{L}^{\mathfrak{r}}$ die Gesamtkostenfunktion eines Agenten $\mathfrak{r}$, welche sich aus der Summe über den Prädiktionshorizont $k = 1, 2, \dots, K$ einzelner Kostenfunktionsaggregationen L^{r_k} (siehe Kapitel 5) für einen Agentenzeitreihenpunkt r_k zum diskreten Zeitpunkt t_k mit einem Zeitschritt $\overset{\diamond}{\Delta} t_k$ ergibt. Die Gesamtkostenfunktion $\mathfrak{L}^{\mathfrak{r}}$ ist ein Abbild einer Trajektorie $T^{\mathfrak{r}}_{{}^{\diamond,\dashv^{\mathfrak{r}}}\mathbf{x}_0}$ eines Agenten $\mathfrak{r}$ abhängig von seinem initialen Zustand ${}^{\diamond,\dashv^{\mathfrak{r}}}\mathbf{x}_0$, einer Agentenzeitreihenmenge $\mathfrak{R}$ basierend auf einer Zeitreihe $\mathfrak{t}$ sowie der zugrundeliegenden Zeitschrittfolge $\Delta\mathfrak{t}$. Eine Trajektorie ergibt sich aus einer Kombination aus einem zulässigen Zustand ${}^{\diamond,\dashv^{r_k}}\mathbf{x}_k$ aus der Menge der zeitdiskreten zulässigen Zustände $\mathbb{X}_{{}^{\diamond,\dashv^{\mathfrak{r}}}\mathbf{x}_0}(t_k)$ abhängig von der Zeit t_k und dem initialen System ${}^{\diamond,\dashv^{\mathfrak{r}}}\mathbf{x}_0$ und einer zulässigen Stellgröße ${}^{\diamond,\dashv^{r_k}}\mathbf{u}_k \in \mathbb{U}_{\mathbb{X}_{{}^{\diamond,\dashv^{\mathfrak{r}}}\mathbf{x}_0}(t_k)}$, die von einem vorherigen Systemzustand ${}^{\diamond,\dashv^{r_k}}\mathbf{x}_{k-1}$ zu dem Systemzustand führt. Weiterhin bedeuten die Nebenbedingungen 2.4.11 und 2.4.12 in Kombination mit der Definitionsmenge $\mathbb{R}_0^+$, dass die Zeitreihe stets positiv und monoton steigend ist. Zudem ist somit die Zeitdifferenz zwischen vorherigem und nachfolgendem Zeitschritt nicht negativ. Die letzte Nebenbedingung 2.4.13 erzwingt die Zusammenhängigkeit einer Trajektorie, sodass sie realisierbar ist. Jedoch ist stets die Abweichung zwischen einer Modellfunktion bzw. der Zustandsübertragungsfunktion $\mathfrak{f}$ und der realen Systemübertragungsfunktion $\mathfrak{s}$ zu beachten. Entscheidend bei der Lösung des Problems ist, dass ausschließlich die Stellgrößenfolgen $[{}^{\diamond,\dashv}\mathbf{u}_k]_{k=1,2,\dots,K}$, auch Steuerfolge genannt, gewählt werden kann. Alle anderen Parameter hängen entweder indirekt von dieser ab oder können nicht von der Regelung beeinflusst werden. Dennoch sind dies keine Konstanten. Manche Optimierungsverfahren können ihr Diskretisierungsintervall durch die Modifikation der Zeitschrittfolge $\Delta\mathfrak{t}$ und damit der Planungszeit $\mathfrak{t}$ anpassen (siehe Abschnitt 2.4.3).

2.4.2. Control Particle Belief Propagation

Dieser Abschnitt beruht auf der Methode „Control Particle Belief Propagation“ (kurz CPBP) aus dem Artikel Hämäläinen, Rajamäki und Liu [57] und der Betrachtung dieser in der Arbeit von Puzicha [141]. CPBP ist ein universeller modellprädiktiver Regler. Dieser vorwärts simulierende Optimierer basiert auf der „Particle Belief Propagation“

(PBP, vgl. [68]) und ist im Vergleich zu anderen Optimierungsalgorithmen robust gegenüber Unstetigkeiten und Multimodalität der Kostenfunktionen. Die Methode der stichprobenbasierten „Belief Propagation“ (kurz BP) erlaubt eine effiziente Zerlegung eines hoch-dimensionalen Problems innerhalb des Optimierers. Allerdings entsteht durch die stichprobenbasierte Methode Rauschen im Ausgangssignal des Optimierers, welches der besten ermittelten Stellgrößenfolge entspricht. Dieses Rauschen ist ebenfalls am Systemausgang messbar. Es äußert sich in einer stetig leicht veränderten Trajektorie nach jedem Optimierungsprozess. CPBP verzichtet auf jegliche Vorberechnungen zur Erstellung von Verhaltensregeln, Referenzdaten und Zustandsautomaten, die die Handlungen in simplere Handlungssequenzen unterteilen. Somit bildet es einen komplementären Ansatz zu anderen physikalischen Echtzeitsystemen, die auf vorausberechneten oder aufgenommenen Daten, erlernten Verhaltensregeln oder spezialisierten Zustandsautomaten zur Zerlegung komplexer Handlung beruhen.
Diese Systeme benötigen signifikanten Berechnungsaufwand vor der eigentlichen Ausführung (Englisch: „Offline Effort“) und führen zu einem weniger robusten Echtzeitverhalten in unbekannten und ungelernten Situationen und Umgebungen (vgl. [57] und [7]). Stattdessen wird ein Algorithmus genutzt, der ein adaptives Echtzeitverhalten aus einfachen Kostenfunktionen extrahiert. Dadurch verlagert sich jedoch ein Teil der Berechnungskomplexität in die Modellierung, Parametrierung und Parametrisierung der Kostenfunktionen (siehe Kapitel 5). Daraus ergibt sich der wesentliche Unterschied zur ursprünglichen Formulierung, der bereits in der Formulierung der Potenzialfunktionen (siehe Gleichung 2.4.14) liegt. Es wird explizit die Trennung der Zustands- und Stellgrößenkosten vermieden, sodass eine größere Funktionsfreiheit für die Kostenfunktionen bzw. Missionen ermöglicht wird. Die restliche Formalisierung des Algorithmus bleibt nahezu identisch, wird aber in einer laufzeitoptimierten Variante implementiert. Entscheidende Weiterentwicklungen werden in der erweiterten Variante „Advanced Control Particle Belief Propagation“ (siehe Abschnitt 2.4.3) erläutert.
Zunächst wird die grundlegende Idee des CPBP-Algorithmus erklärt und anschließend werden detailliert die einzelnen Teilaspekte erläutert. Der Algorithmus führt zunächst eine Trajektorienplanung und -glättung durch. Daraufhin werden die gewonnen Informationen für den nächsten Aufruf aufbereitet, um eine höhere Konvergenzgeschwindigkeit aufgrund zeitlicher Kausalitäten und Ähnlichkeiten zu erzielen. Als Ähnlichkeiten sind vor allem Stetigkeiten physikalischer Systeme in der Umwelt (siehe Kapitel 6) zu verstehen. CPBP umfasst drei Hauptroutinen:

1. Erkunde die zukünftige Entwicklung des Systems bis zu einem Planungshorizont K mittels N informierten zufälligen stochastischen Irrfahrten (Englisch: „Guided Random Walks“). Dazu werden Stellgrößenvektoren aus einer Verteilung β über dem Stellgrößenraum gezogen und für die Simulation der Dynamik bzw. Übertragungsfunktion $\mathfrak{f}$ verwendet, um N zugehörige Trajektorie $T^{(n)} \in \mathcal{T}$ mit $n = 1, 2, \ldots, N$ zu erhalten.

2. Glätte die optimale Trajektorie $\tilde{T}$ aus dem vorherigen Schritt durch rückwärts gewandte rekursive lokale Verfeinerung der Stellgrößen.

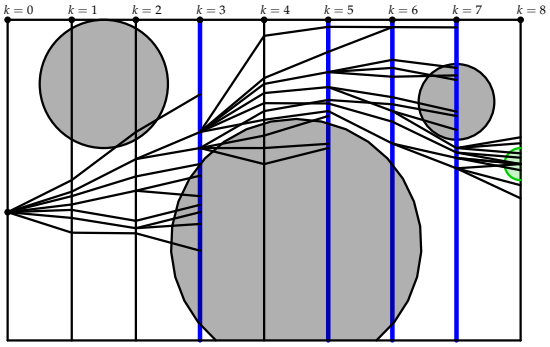


Abbildung 2.16.: Schritt 1 [141] (vgl. [57, S. 2]). Entwicklung der informierten stochastischen Irrfahrt bis zu einem Prädiktionszeitpunkt k, bei dem die Kosten der einzelnen Trajektorienstücke stark divergieren. Dies führt zu einer Umverteilung, markiert mittels blauer Linien zu Zeitschritten $k = 3, 5, 6, 7$.

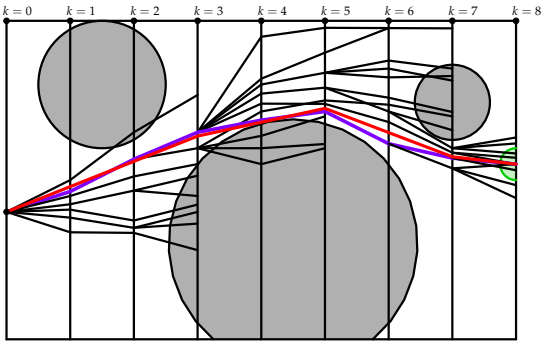


Abbildung 2.17.: Schritt 2 [141] (vgl. [57, S. 2]). Die Irrfahrt mit den geringsten Gesamtkosten $\tilde{r_{\mathfrak{L}}}$, violett hervorgehoben, wird rückwärtig geglätt und resultiert in der roten optimalen Trajektorie $\tilde{T}$. Diese reicht von der Startposition bei $k = 0$ bis hin zu der grünen Zielregion bei $k = 8$.

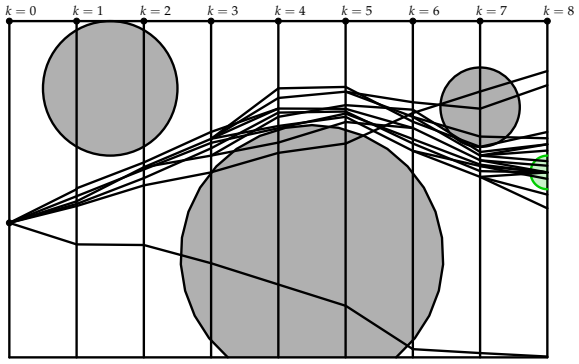


Abbildung 2.18.: Schritt 3 [141] (vgl. [57, S. 2]). Die Informationen über die optimale Trajektorie $\tilde{T}$ werden als „Warmstart"-Information an die nächste Iteration weitergegeben. Dies erhöht die Informiertheit der Irrfahrten, sodass diese geringere Kosten aufweisen.

3. Wende die optimale Stellgrößenfolge $[\tilde{\overset{\diamond}{\mathbf{u}}}_k]_{k=1,2,\ldots,K}$ auf das System $\mathfrak{s}$ an und übergebe die Information über die prädizierte Entwicklung für den nächsten Aufruf.

Die Kostenfunktion $\mathfrak{L}$ wird für jede Trajektorie $T^{(n)}$ bei jedem Prädiktionsschritt k ausgewertet. Falls eine Vielzahl der Trajektorien $\mathcal{T}$ hohe Gesamtkosten aufweisen, so wird eine Umverteilung angewendet; sie ist in Abbildung 2.16 als blaue Linie zu erkennen. Die Umverteilung ordnet die Rechenkapazität neu zu, indem Trajektorien mit sehr hohen Kosten vernachlässigt und die Günstigen vervielfältigt werden. Die prädiktive Simulation und die rückwärtige Glättung der Trajektorie mit minimalen Kosten bilden eine Iteration. Die erhaltene optimale Trajektorie $\tilde{T}$ wird für die folgenden informierten zufälligen Irrfahrten verwendet, um die Konvergenz zu beschleunigen.
Abbildungen 2.16, 2.17 und 2.18 veranschaulichen das Verfahren, bei dem ein Objekt sich von links nach rechts bewegen soll, sodass k sowohl die Zeit als auch die vertikale

Position beschreibt. Die Zielregion ist grün gefärbt. Abbildung 2.16 zeigt die baumartige Graphstruktur der im vorwärts gerichteten Durchlauf erzeugten Trajektorienstücke. Die optimalen Stücke werden im zweiten Schritt (siehe Abbildung 2.17) in Violett und die durch rückwärtige lokale Verfeinerung generierte Trajektorie $\tilde{T}$ in Rot dargestellt. Die Informationen über die optimale Trajektorie $\tilde{T}$ wird als „Warmstart"-Information an die nächste Iteration weitergegeben. Dies erhöht die Informiertheit der Irrfahrten, sodass die Trajektorienstücke des folgenden Algorithmusaufrufs in Abbildung 2.18 geringere Kosten aufweisen.

2.4.2.1. Einführung der Notation

$\wp$ bezeichne eine Situation zu einem Zeitpunkt, an dem Prädiktionen über die zeitliche Entwicklung eines Systems $\mathfrak{s}$ aufgrund unterschiedlicher Stellgrößen mit Hilfe einer Übertragungsfunktion $\mathfrak{f}$ getätigt werden können. Des Weiteren sei k eine Zeitpunktindexdifferenz, sodass $a_{k,\wp}$ eine Größe a zum Zeitpunkt $t_{\wp+k}$ aus der Sicht des Zeitpunktes $t_\wp$ sei. Somit bezieht sich $k < 0$ auf die Vergangenheit, $k > 0$ auf die Zukunft in Form einer Prädiktion und $k = 0$ auf die Gegenwart der zeitlichen Situation von $\wp$. Die Abkürzung a_k wird auf die aktuelle zeitliche Situation $\wp = 0$ bezogen verwendet. Eine Gruppe von Stichproben der Größe a indiziert mit n, wird mit $a^{(n)}$ angegeben. Aufgrund der Umverteilung wird eine Historienfunktion $h(n, k, -l)$ eingeführt, die Stichproben mit Index n zum Zeitpunkt k auf den zugehörigen Index derselben Trajektorie $T^{(n)}$ zum Zeitpunkt $k - l$ abbildet. Dabei bildet $\mathbf{x}_0 = \mathbf{x}_\wp$ stets den rekursiven Beginn der Historienfunktion. Zur erhöhten Leserlichkeit werden in diesem Abschnitt die Stellgrößen ${}^{\diamond,\dashv}\mathbf{u}_k$ mit $\mathbf{u}_k$ und die Zustände ${}^{\diamond,\dashv}\mathbf{x}_k$ mit $\mathbf{x}_k$ abgekürzt.

2.4.2.2. Unterschied zur Regelung mit expliziter Führungsgröße

Um die Güte bzw. Optimalität der Regelung zu bewerten, muss eine Kostenfunktion $\mathfrak{L}$ in Abhängigkeit des Systemzustandes und der Stellgröße angegeben werden. Im Unterschied zur Regelung mit expliziten Führungsgrößen, gibt es bei dem verwendeten Optimierungsproblem (siehe Abschnitt 2.4.1) weder einen bekannten Zielzustand $\mathbf{x}^{\text{ref}}$ noch eine Referenztrajektorie $[\mathbf{x}_k^{\text{ref}}]_{k=1,2,\ldots,K}$, mit Hilfe derer ein Abstand zu einem bekannten Optimum bewertet werden kann. Somit kann die Kostenfunktion deutlich freier, sogar unstetig, gewählt werden. Andererseits ist die Berechnung der optimalen Stellgrößenfolge zur Minimierung der Kosten aufgrund von Nichtlinearitäten und Unstetigkeiten für beliebige Kostenfunktionen $\mathfrak{L}$ und Übertragungsfunktionen $\mathfrak{f}$ nur mit sehr wenigen Verfahren lösbar. Daher verarbeitet und vergleicht CPBP nur Stichproben von Stützstellen dieser Funktionen, die als ein Markov-Netzwerk aufgefasst werden können.

2.4.2.3. Beschreibung als Markov-Netzwerk

Sei $\mathbf{z}_k = [\mathbf{x}_k, \mathbf{u}_k]^T = \left(T_{[k]}\right)^T$ der Optimierungsvektor zum Prädiktionszeitpunkt k. Nachfolgend wird die Minimierung der Kosten $\mathfrak{L}$ in die Maximierung der Wahrscheinlichkeitsdichtefunktion $\mathcal{P}(\mathbf{z})$ der vollständigen Trajektorie $\mathbf{z} = [\mathbf{z}_k]_{k=1,2,\ldots,K} = T^T$ über-

führt. Dabei sei $K \in \mathbb{N}$ der Prädiktionshorizont. Die Maximierung wird durch die Umwandlung der Kostenfunktionen in Wahrscheinlichkeitsdichten mittels Exponentialfunktionen und durch approximierte Abtastung von $\mathcal{P}(\mathbf{z})$ erreicht; dies entspricht der Ziehung von Stichproben im Bereich der maximalen Dichte.
Falls ausschließlich Zustands- und Stellgrößenkosten $\mathfrak{L}$ betrachtet werden, so kann $\mathcal{P}'(\mathbf{z})$ wie folgt beschrieben werden:

$$\begin{aligned}\mathcal{P}'(\mathbf{z}) &= \frac{1}{Z}\prod_{k=1}^{K}\exp\{-\frac{1}{2}(\mathfrak{L}(T,\dots)+c(\mathbf{u}_k,k))\}\\ &= \frac{1}{Z}\prod_{k=1}^{K}\exp\{-\frac{1}{2}(\mathfrak{L}(\mathbf{z}^T,\dots)+c(\mathbf{u}_k,k))\}\\ &= \frac{1}{Z}\prod_{k=1}^{K}\alpha_k(\mathbf{z}_k)\beta_k(\mathbf{u}_k)\\ &= \frac{1}{Z}\prod_{k=1}^{K}\psi_k(\mathbf{z}_k)\end{aligned} \tag{2.4.14}$$

Dabei entspricht Z einer Normalisierungskonstanten und α_k und β_k bezeichnen Kostenpotenzialfunktionen und Stellgrößenpotenzialfunktionen, sodass $\int_{-\infty}^{\infty}\frac{1}{Z_{\alpha_k}}\alpha_k(\mathbf{z}_k)d\mathbf{z}_k = 1$ und $\int_{-\infty}^{\infty}\frac{1}{Z_{\beta_k}}\beta_k(\mathbf{z}_k)d\mathbf{z}_k = 1$ gilt. Daraus folgt:

$$\int_{-\infty}^{\infty}\mathcal{P}'(\mathbf{z}) = \int_{-\infty}^{\infty}\prod_{k=1}^{K}\frac{1}{Z_{\alpha_k}}\alpha_k(\mathbf{z}_k)\frac{1}{Z_{\beta_k}}\beta_k(\mathbf{u}_k)d\mathbf{z} = 1 \tag{2.4.15}$$

Die Möglichkeit, Kostenfunktionen in exponentielle Wahrscheinlichkeitsdichten zu transformieren, wird in der Veröffentlichung E. Todorov [31] gezeigt. Die Potenzialfunktionen entsprechen Normalverteilungen, falls die Kostenfunktionen quadratisch sind, und lassen sich zu $\psi_k(\mathbf{z}_k)$ zusammenfassen. Im Gegensatz zum Verfahren in der Veröffentlichung von Hämäläinen, Rajamäki und Liu [57] wird auf die explizite Separierung der Stellgrößen- und Zustandskosten verzichtet, da dies nicht notwendig ist und somit mehr Modellierungsfreiheit ermöglicht. Ferner wird β_k aufgrund physikalischer Systemeigenschaften gewählt (siehe Abschnitt 2.4.2.11).
Die Gleichung 2.4.14 nimmt implizit an, dass die Folge von $\mathbf{z}_k$ stets eine gültige Trajektorie T im Produkt des Zustands- und Stellgrößenraumes bildet. Jedoch werden $\mathbf{z}_k$ als unabhängige zufällige Variablen aufgefasst, daher muss die Gleichung 2.4.14 dahingehend erweitert werden, dass die Wahrscheinlichkeit der Verknüpfung eines Trajektorienstücks $\mathbf{z}_k$ mit seinem adjazenten Vorgänger $\mathbf{z}_{k-1}$ und Nachfolger $\mathbf{z}_{k+1}$ in die Gesamtwahrscheinlichkeit einfließt:

$$\begin{aligned}\mathcal{P}(\mathbf{z}) &= \frac{1}{Z}\left(\prod_{k=1}^{K}\psi_k(\mathbf{z}_k)\right)\left(\prod_{k=1}^{K}\Psi_{\text{fwd}}(\mathbf{z}_{k-1},\mathbf{z}_k)\right)\left(\prod_{k=0}^{K-1}\Psi_{\text{bwd}}(\mathbf{z}_{k+1},\mathbf{z}_k)\right)\\ &= \frac{1}{Z}\prod_{k=1}^{K}\psi_k(\mathbf{z}_k)\Psi_{\text{fwd}}(\mathbf{z}_{k-1},\mathbf{z}_k)\Psi_{\text{bwd}}(\mathbf{z}_{k+1},\mathbf{z}_k)\end{aligned} \tag{2.4.16}$$

Seien Ψ_{fwd} und Ψ_{bwd} die vorwärts und rückwärts gerichteten Übergangspotenziale. Sie werden mit Hilfe der Gleichung 2.4.17 definiert:

$$\begin{aligned}\Psi_{\text{fwd}}(\mathbf{z}_{k-1}, \mathbf{z}_k) &= \Psi_{\text{bwd}}(\mathbf{z}_k, \mathbf{z}_{k+1}) \\ &= \mathfrak{N}(\mathbf{x}_k; \mathfrak{f}(\mathbf{x}_{k-1}, \mathbf{u}_k, \dots), \mathbf{Q})\end{aligned} \tag{2.4.17}$$

Hier bezeichne $\mathfrak{N}(\mathbf{x}; \boldsymbol{\mu}, \mathbf{C}) = \exp\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\mathbf{C}^{-1}(\mathbf{x}-\boldsymbol{\mu})\}$ eine nicht normalisierte Gaußfunktion von $\mathbf{x}$ mit Mittelwert $\boldsymbol{\mu}$ und Kovarianzmatrix $\mathbf{C}$. Die gaußschen Normalisierungen werden an dieser Stelle vernachlässigt, da sie in Z vereint werden können. Als Besonderheit müssen Potenziale für den Fall $\mathbf{Q} = \mathbf{0}$ ebenfalls ausgewertet werden können, jedoch wird zunächst Folgendes angenommen:

$$\lim_{\mathbf{Q}\to\mathbf{0}} \Psi_{\text{fwd}}(\mathbf{z}_{k-1}, \mathbf{u}_k) = \begin{cases} 1 & \text{, falls } \mathbf{x}_k = \mathfrak{f}(\mathbf{x}_{k-1}, \mathbf{u}_k, \dots) \\ 0 & \text{, sonst} \end{cases} \tag{2.4.18}$$

Die Gleichung 2.4.16 entspricht einer simplen Form eines Markov-Netzwerks. Dabei sind die einzelnen Variablen $\mathbf{z}_k$ bedingt abhängig von ihren Nachbarn $\mathbf{z}_{k-1}$ und $\mathbf{z}_{k+1}$ und von allen anderen Variablen bedingt unabhängig. Folglich kann diese Marokov-Netzwerkstruktur als Wahrscheinlichkeitsgraph betrachtet werden (siehe Abbildung 2.19). Es korrespondiert jeder Knoten des Graphmodells mit genau einem $\mathbf{z}_k$ eines Prädiktionsschrittes k und die Kanten bezeichnen die Abhängigkeiten zwischen den Variablen. Die Umformung in ein ungerichtetes Graphmodell (siehe Gleichung 2.4.19) ermöglicht die nun folgende Anwendung der BP für Markov-Netzwerke (Englisch: „Markov Random Field“, vgl. [68]):
Sei $G = (\mathcal{V}, \mathcal{E})$ ein ungerichteter Graph bestehend aus der Knotenmenge
$\mathcal{V} = \{1, 2, \dots, K\}$ und der Kantenmenge $\mathcal{E}$. Ferner sei Γ_k als die Menge der Nachbarknoten l eines Knoten $k \in \mathcal{V}$ des Graphen G, sodass $\{k, l\} \in \mathcal{E}$ gilt. In einem Wahrscheinlichkeitsmodell wird jedem Zustand k eine Zufallsvariable $\acute{\mathbf{Z}}_k$ einer Trajektorienmenge $\mathbb{T}_k$ zugewiesen. Dabei wird angenommen, dass für jeden Knoten k und jede Kante $\{k, l\}$ zugehörige Potenzialfunktionen ψ_k und $\Psi_{k,l}$ existieren, die eine Wahrscheinlichkeitsdichteverteilung erzeugen:

$$\mathcal{P}(\mathbf{z}) = \frac{1}{Z}\left(\prod_{k=1}^{K} \psi_k(\mathbf{z}_k)\right)\left(\prod_{\{k,l\}\in\mathcal{E}} \Psi_{k,l}(\mathbf{z}_k, \mathbf{z}_l)\right) \tag{2.4.19}$$

Sei $\mathbf{z}$ ein Ergebnis aller K Zufallsexperimente und folglich $\mathbf{z}_k$ das Ergebnis eines Zufallsexperimentes für die Zufallsvariable $\acute{\mathbf{Z}}_k$. Des Weiteren wird Z als skalare Normalisierung der Dichte $\mathcal{P}$ gewählt. Diese wird auch als Partitionierungsfunktion bezeichnet. Die Gleichung 2.4.19 definiert somit ein paarweises Markov-Netzwerk, das in Abbildung 2.19 abgebildet ist (vgl. [68]). Werden die Kanten nun in vorwärts und rückwärts gerichtete Kanten aufgeteilt, so entspricht dies der Gleichung 2.4.16. Dabei ist die „Richtung“ der Kanten nicht als solche eines gerichteten Graphen zu verstehen, da die Wahrscheinlichkeiten sich wechselseitig beeinflussen, sondern als temporale Ordnung der Knoten k über K. Dies entspricht dem Planungshorizont der Optimierung. Folglich bedeutet vorwärts gerichtet ein Fortschreiten mit den Zeitschritten k, welche mittels

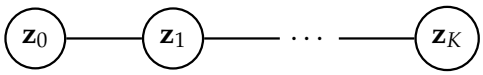


Abbildung 2.19.: Markov-Netzwerk, dessen Knoten die Zustands- und Stellgrößenvariablen eines Prädiktionszeitschrittes bis zur Horizontlänge K darstellen (vgl. [57, S. 4], [141] und [68])

t_k und Δt_k definiert sind (siehe Abschnitte 2.2.2 und 2.4.1), und rückwärts gerichtet einen Rückschritt. Der rückwirkende Informationsfluss in Form der Gesamtkosten der Trajektorie ist wichtig, da zunächst günstig erscheinende Trajektorienstücke in der Gesamtbetrachtung eine schlechte Wahl darstellen können.

2.4.2.4. Particle Belief Propagation

Basierend auf der graphischen Formulierung als Markov-Netzwerk kann untersucht werden, wie die PBP als allgemeine stichprobenbasierte BP angewendet wird (vgl. [68]). Der Hauptgrund für die Verwendung besteht in der Vereinigung zweier umfassender Verfahren. Einerseits wird das globale Abtasten zur Erkundung multimodaler Optimierungsumgebungen und andererseits die dynamische Programmierung zur Behandlung der hohen Optimierungsdimensionen angewendet. PBP wertet Fragmente $\mathcal{P}_k(\mathbf{z}_k)$ der Verteilungsfunktion aus statt die höher dimensionale gesamte Verteilungsfunktion $\mathcal{P}(\mathbf{z})$ zu verarbeiten. Dadurch wird auf Trajektorienstücken $\mathbf{z}_k$ gearbeitet und nicht auf der gesamten Trajektorie $T = \mathbf{z}^T$. Diese Stücke sind nicht direkt von Interesse, bieten jedoch die Möglichkeit, dass durch die Schätzung dieser Verteilungsfunktionsfragmente mit Hilfe der BP eine adäquate Menge an vollständigen Trajektorien erzeugt werden kann.
Äquivalent zu anderen BP Methoden versucht PBP die Glaubwürdigkeit (Englisch: „Beliefs“) $\mathcal{B}_k(\mathbf{z}_k)$ der Variablen k des Graphen zu berechnen. Hierbei ist die Glaubwürdigkeit proportional zur Fragmentwahrscheinlichkeitsdichtefunktion $\mathcal{P}_k(\mathbf{z}_k)$, falls der Graph zyklenfrei ist. Als Konsequenz beruht die Glaubwürdigkeit des Knoten k auf von anderen Knoten gesendeten Nachrichten $m_{l\to k}(\mathbf{z}_k)$ (vgl. [68]):

$$\mathcal{B}_k(\mathbf{z}_k) = \psi_k(\mathbf{z}_k) \prod_{l\in\Gamma_k} m_{l\to k}(\mathbf{z}_k) \tag{2.4.20}$$

$$m_{l\to k}(\mathbf{z}_k) = \sum_{\mathbf{z}_l\in\mathbb{T}_l} \Psi_{l,k}(\mathbf{z}_l, \mathbf{z}_k)\psi_l(\mathbf{z}_l) \prod_{j\in\Gamma_l\setminus k} m_{j\to l}(\mathbf{z}_l) \tag{2.4.21}$$

Γ_l beschreibt die Menge der Nachbarknoten von l und $\mathbb{T}_l$ sei der Raum von $\mathbf{z}_l$. Die Nachrichten $m_{l\to k}$ von Knoten l zu Knoten k können als nicht normalisierte Wahrscheinlichkeitsdichtefunktion der Zielknotenvariable $\mathbf{z}_k$ aufgefasst werden. Das Potenzial $\psi_k(\mathbf{z}_k)$ repräsentiert einen Beleg für $\mathbf{z}_k$ basierend auf den Kostenfunktionen $\mathfrak{L}$ und $c(\dots)$, welcher mittels der Nachrichten im Graphen bekannt gemacht wird. Die Glaubwürdigkeit $\mathcal{B}_k(\mathbf{z}_k)$ bildet sich aus dem Produkt der direkten und der vermittelten Belege.
Die Nachrichten und die Glaubwürdigkeit der Gleichungen 2.4.20 und 2.4.21 werden mittels N Stichproben $\acute{q}_k^{(n)} \sim q(\mathbf{z}_k^{(n)})$ geschätzt. Sei $n = 1, 2, \dots, N$ der Stichprobenindex und $q(\mathbf{z}_k^{(n)})$ eine beliebig gewählte Dichteverteilung sowie $q_k^{(n)}$ die Realisierung

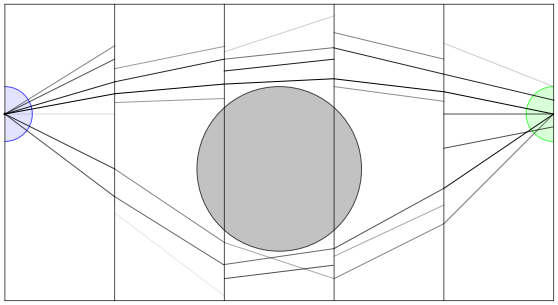


Abbildung 2.20.: Direkte Anwendung der „Particle Belief Propagation“ (vgl. [57, S. 4] und [141]). Die blaue Region entspricht dem initialen Zustand $\mathbf{x}_0$ und die grüne Region bildet den Zielzustand ab. Der graue Kreis stellt ein Hindernis bzw. einen unzulässigen Bereich dar. Die Trajektorienstücke werden abhängig von ihrer Glaubwürdigkeit dunkler gefärbt, je höher diese ist.

der Stichprobe bzw. Zufallsvariable $q_k^{(n)}$. Die Schätzung wird mit $\hat{m}$ verdeutlicht. Wird die Potenzialfunktion ψ_k durch die Realisierung $q_k^{(n)}$ dividiert, so ergeben sich eine gewichtete Stichprobennachricht und -glaubwürdigkeit (vgl. [68]):

$$\hat{m}_{l\to k}(\mathbf{z}_k^{(n)}) = \frac{1}{N}\sum_{m=1}^{N} \Psi_{l,k}(\mathbf{z}_l^{(m)}, \mathbf{z}_k^{(n)}) \frac{\psi_l(\mathbf{z}_l^{(m)})}{q_l^{(m)}} \prod_{j\in\Gamma_l\setminus k} \hat{m}_{j\to l}(\mathbf{z}_l^{(n)}) \tag{2.4.22}$$

$$\hat{\mathcal{B}}_k(\mathbf{z}_k^{(n)}) = \frac{\psi_k(\mathbf{z}_k^{(n)})}{q_k^{(n)}} \prod_{l\in\Gamma_k} \hat{m}_{l\to k}(\mathbf{z}_k^{(n)}) \tag{2.4.23}$$

N sei die Größe der Stichprobe. Die Stichprobenglaubwürdigkeit $\hat{\mathcal{B}}_k(\mathbf{z}_k^{(n)})$ repräsentiert die Fragmentwahrscheinlichkeitsdichtefunktion des Trajektorienstücks $\mathbf{z}_k$, das durch einen Simulationsschritt Δt_k mit der Stellgröße $\mathbf{u}_k^{(n)}$ im Zustand $\mathbf{x}_k^{(n)}$ endend erzeugt wird. Wenn gleich dies keine vollständige optimale Zustands- oder Stellgrößentrajektorie generiert, so kann die Glaubwürdigkeit dennoch in Abbildung 2.20 visualisiert werden. Es gilt in Abbildung 2.20, je dunkler die Färbung ist, desto wahrscheinlicher ist das zugehörige Trajektorienstück. Dabei kann die Kovarianz $\mathbf{Q}$ zur Feinjustierung verwendet werden; mit geringem Übergangsrauschen werden die geringeren Übergangspotenziale zwischen Segmenten aufgrund der Maschinenpräzision zu Null ausgewertet. Konträr dazu bewirkt ein zu großes Übertragungsrauschen eine Mittelwertbildung der Glaubwürdigkeiten. Dies würde sich in Abbildung 2.20 in einem Verschwimmen der Fragmente äußern, sodass die Extrema weniger deutlich separiert wären.
Ferner zeigt die Abbildung 2.20, dass die Grundform von PBP bereits fähig, ist die multimodale Fragmentverteilung, die mit verschiedenen Pfaden um Hindernisse herum korrespondiert, zu approximieren. Somit liegen die korrekten Informationen für das Optimalsteuerungsproblem (siehe Abschnitt 2.4.1) vor, müssen jedoch noch weiter extrahiert werden.

2.4.2.5. Methode

Basierend auf dem PBP-Algorithmus fügt der CPBP-Algorithmus verschiedene Neuerungen für die Optimierung komplexer physikalischer Systeme hinzu:

1. Eine Auswahl an Vorschlägen, die die Stichprobenziehung physikalisch realisierbarer Trajektorien ermöglicht
2. Ein adaptives Stichprobenziehen zur Anpassung zwischen lokaler und globaler Suche
3. Eine rückwärtige lokale Verfeinerung der Trajektorien
4. Eine Vermeidung von Null-Wert-Nachrichten $\hat{m}_{l\to k} = 0$
5. Eine Übermittlung der Informationen nicht nur innerhalb des Graphen aus Abbildung 2.19, sondern auch zwischen Algorithmusaufrufen, inklusive verschiedener Kontrollprioren zur Glättung der Trajektorien und zur Unterstützung der Konvergenz
6. Behandlung des Sonderfalls der Kovarianzmatrix $\mathbf{Q} = \mathbf{0}$

Das offenkundigste Problem des PBP Algorithmus ist, dass die Trajektorienstücke $\mathbf{z}_k$ nicht verbunden sind; somit können sie nicht direkt zu einer physikalisch realisierbaren Gesamttrajektorie kombiniert werden. Außerdem kann der Startpunkt eines Trajektorienstücks innerhalb eines unzulässigen Bereiches liegen. Somit sollten die Vorschläge so gewählt werden, dass sie mögliche realisierbare Trajektorien ergeben, für die $\mathcal{P}(\mathbf{z})$ ausgewertet kann. Die beste ausgewählte Trajektorie $\tilde{T}$ kann als Lösung sowohl zur Regelung als auch als initiale Lösung, Warmstart genannt, des nächsten Algorithmusaufrufs genutzt werden. Die berechneten Glaubwürdigkeiten beeinflussen dabei die informierten Irrfahrten (Englisch: „Guided Random Walks“) des nächsten Aufrufs.

2.4.2.6. Auswahl der Dichtefunktion

Unter Berücksichtigung der gewichteten Erkundung sollte die ausgewählte Wahrscheinlichkeitsdichtefunktion der Trajektorien möglichst nah an die tatsächliche Dichtefunktion heran kommen. Dies wird durch eine Menge $\mathcal{T}$ zufälliger Irrfahrten (Englisch: „Random Walks“) erzielt, die im Startzustand $\mathbf{x}_0$ starten und deren Stellgrößen als Realisierung einer Zufallsvariablen aus der Dichteverteilung, dem so genannten Potenzial, $\acute{\mathbf{u}}_k^{(n)} \sim \beta_k(\mathbf{u}_k^{(n)})$ gezogen werden. Die zugehörigen Zustände werden durch die Simulation der Systemdynamik bestimmt, $\mathbf{x}_k^{(n)} = \mathfrak{f}(\mathbf{x}_{k-1}^{(h(n,k,-1))}, \mathbf{u}_k^{(n)})$. Die Historienfunktion $h(\dots)$ ist erforderlich, da die Stichprobenindizes durch eine mögliche Umverteilung nach einem Prädiktionsschritt k verändert sein können. Durch die vollständige Bestimmtheit eines Zustands $\mathbf{x}_k^{(n)}$ durch die zugehörige Stellgröße $\mathbf{u}_k^{(n)}$ hängt die Dichtefunktion $q(\mathbf{z}_k^{(n)})$ ausschließlich von der Stellgröße $\mathbf{u}_k^{(n)}$ ab. Folglich wird q definiert:

$$q(\mathbf{z}_k^{(n)}) = \beta_k(\mathbf{u}_k^{(n)}) \tag{2.4.24}$$

Die gewählte Dichtefunktion garantiert die Verbindung der Trajektorienstücke $\mathbf{z}_k$ und vereinfacht die Fragmentglaubwürdigkeit und Fragmentnachrichten der Gleichungen 2.4.22 und 2.4.23. Die Wahl impliziert, dass die Gleichungen ausschließlich auf den Potenzialfunktionen beruhen. Aufgrund der Kettenstruktur des resultierenden Graphen lassen sich die Nachrichten in vorwärts und rückwärts gerichtete Nachrichten $\hat{m}_{\text{fwd}}$ und $\hat{m}_{\text{bwd}}$ unterteilen:

$$\hat{\mathcal{B}}_k(\mathbf{z}_k^{(n)}) = \alpha_k(\mathbf{z}_k^{(n)})\hat{m}_{\text{fwd}}(\mathbf{z}_k^{(n)})\hat{m}_{\text{bwd}}(\mathbf{z}_k^{(n)}) \tag{2.4.25}$$

$$\hat{m}_{\text{fwd}}(\mathbf{z}_k^{(n)}) = \frac{1}{N}\sum_{j=1}^{N}\Psi_{\text{fwd}}(\mathbf{z}_{k-1}^{(j)}, \mathbf{z}_k^{(n)})\alpha_{k-1}(\mathbf{z}_{k-1}^{(j)})\hat{m}_{\text{fwd}}(\mathbf{z}_{k-1}^{(j)}) \tag{2.4.26}$$

$$\hat{m}_{\text{bwd}}(\mathbf{z}_k^{(n)}) = \frac{1}{N}\sum_{j=1}^{N}\Psi_{\text{bwd}}(\mathbf{z}_{k+1}^{(j)}, \mathbf{z}_k^{(n)})\alpha_{k+1}(\mathbf{z}_{k+1}^{(j)})\hat{m}_{\text{bwd}}(\mathbf{z}_{k+1}^{(j)}) \tag{2.4.27}$$

Bei der Algorithmusimplementierung können die vorwärts gerichteten Nachrichten in einem vorwärts gerichteten Durchlauf des Graphen aktualisiert werden, gefolgt von einem rückwärtigen Durchlauf, der die rückwärts gerichteten Nachrichten und die Glaubwürdigkeiten aktualisiert.

2.4.2.7. Umverteilung

Während des vorwärts gerichteten Durchlaufs können die vorwärts gerichteten Nachrichten mittels einer rekursiven Formulierung durch die vorwärts gerichtete Glaubwürdigkeit ausgedrückt werden:

$$\hat{\mathcal{B}}_{\text{fwd}}(\mathbf{z}_k^{(n)}) = \alpha_k(\mathbf{z}_k^{(n)})\hat{m}_{\text{fwd}}(\mathbf{z}_k^{(n)}) \tag{2.4.28}$$

$$\hat{m}_{\text{fwd}}(\mathbf{z}_k^{(n)}) = \frac{1}{N}\sum_{j=1}^{N}\Psi_{\text{fwd}}(\mathbf{z}_{k-1}^{(j)}, \mathbf{z}_k^{(n)})\hat{\mathcal{B}}_{\text{fwd}}(\mathbf{z}_{k-1}^{(j)}) \tag{2.4.29}$$

$\hat{\mathcal{B}}_{\text{fwd}}(\mathbf{z}_k^{(n)})$ ist proportional zur vollständigen Glaubwürdigkeit $\hat{\mathcal{B}}_k(\mathbf{z}_k^{(n)})$. Falls der rückwärtige Durchlauf nicht viel Information hinzufügt, kann die Konvergenz beschleunigt werden, indem Pfade mit geringer vorwärts gerichtetenr Glaubwürdigkeit beendet und Gebiete um Pfade mit hoher Glaubwürdigkeit herum verstärkt erkundet werden. Dies entspricht einer Umverteilung in Partikelfiltern (vgl. [175]); dabei entspricht die vorwärts gerichtete Glaubwürdigkeit der Stichproben- beziehungsweise Partikelgewichtung in der Partikelfilterung. Zu Beginn eines Prädiktionsschrittes k wird die effektive Partikelanzahl N_{eff} des vorherigen Schrittes berechnet:

$$N_{\text{eff}} = \frac{1}{\sum_{n=1}^{N}\left(\frac{\hat{\mathcal{B}}_{\text{fwd}}(\mathbf{z}_{k-1}^{(n)})}{\sum_{j=1}^{N}\hat{\mathcal{B}}_{\text{fwd}}(\mathbf{z}_{k-1}^{(j)})}\right)^2} \tag{2.4.30}$$

Der Nenner entspricht der Summe der quadrierten normalisierten Glaubwürdigkeit. Eine geringe Anzahl N_{eff} verdeutlicht, dass die Glaubwürdigkeit beinahe aller, bis auf

ein paar, Trajektorienstücke gering ist. Falls $N_{\text{eff}} < N_{\text{eff}}^{\min}$ gilt, so muss eine neue Wahrscheinlichkeitsdichtefunktion $\mathcal{P}$ erzeugt werden, sodass $\mathcal{P} \sim \hat{\mathcal{B}}_{\text{fwd}}(\mathbf{z}_{k-1}^{(n)})$ gilt. Für jede Stichprobe n wird zufällig ein Index w aus der Dichteverteilung $\mathcal{P}(\cdot)$ gezogen und die Historienfunktion h angepasst, sodass w auf n im vorherigen Prädiktionsschritt $k-1$ abgebildet wird (siehe 2.4.2.1). Dadurch werden Trajektorien mit geringer Glaubwürdigkeit beendet und diejenigen mit hoher Glaubwürdigkeit vervielfältigt. Analog zur Partikelfilterung, bei der die Stichprobengewichtung nach der Umverteilung konstant gehalten wird, wird die Glaubwürdigkeit $\hat{\mathcal{B}}_{\text{fwd}}(\mathbf{z}_{k-1}^{(j)})$ zum Zeitpunkt der Umverteilung aus der vorwärts gerichteten Nachrichtengleichung 2.4.29 entfernt.
Des Weiteren entspricht $N_{\text{eff}}^{\min} < \frac{1}{N}$ keiner Umverteilung, wodurch jede Trajektorie bis zum Prädiktionshorizont $k = K$ simuliert wird, auch wenn sie unzulässige Bereiche schneidet. Wohingegen $N_{\text{eff}}^{\min} > 1$ eine Umverteilung nach jedem Prädiktionsschritt k erwirkt, sodass ggf. Trajektorien mit anfangs höheren Kosten, allerdings mit geringen Gesamtkosten, frühzeitig beendet werden.
Abbildung 2.16 veranschaulicht die Stichprobenziehung der Fragmente $\mathbf{z}_k$ mit dem vorgeschlagenen Verfahren und die Umverteilung. Dies resultiert in einer baumartigen Graphstruktur der Trajektorien. Durch „Nachbereitung“ (Englisch: „Backtracking“) der Trajektorien $T^{(n)}$, die den Horizont $k = K$ erreichen, können alle N Zustands- und Stellgrößenfolgen vollständig erstellt werden. Die vollständige Verteilungsdichte $\mathcal{P}(\mathbf{z})$ kann nun für alle Trajektorien $T^{(n)}$ ausgewertet werden. Anschließend wird die beste Trajektorie $\tilde{T}$ oder eine lokale Verfeinerung der Trajektorienmenge $\mathcal{T} = \{T^{(1)}, T^{(2)}, \ldots, T^{(N)}\}$ angewendet.

2.4.2.8. Lokale Verfeinerung

Weil die Simulation der Übertragungsfunktion ƒ sehr rechenintensiv sein kann, wird angestrebt, so viele Informationen wie möglich aus der Stichprobe zu extrahieren. Das ist der Grund für das Hinzufügen einer lokalen Verfeinerung (siehe Algorithmus 2.2) bei dem rückwärtigen Durchlauf durch die Baumstruktur:

Algorithmus 2.2.: Lokale Verfeinerung im rückwärtigen Durchlauf durch die Baumstruktur (vgl. [57, S. 5] und [141])

1: **function** LOKALE VERFEINERUNG
2: Initialisiere $\hat{\mathbf{x}}_K \leftarrow \frac{\sum_n^N \alpha(\mathbf{z}_K^{(n)}) \mathbf{x}_K^{(n)}}{\sum_n^N \alpha(\mathbf{z}_K^{(n)})}$
3: **for** $k = K-1, K-2, .., 1$ **do**
4: Berechne das gaußsche Ausgleichsmodell des konkatenierten Vektors $[\mathbf{x}_k^{(n)}, \mathbf{u}_k^{(n)}, \mathbf{x}_{k+1}^{(n)}]$ gewichtet mit $\hat{\mathcal{B}}_{\text{fwd}}(\mathbf{z}_k^{(n)})$
5: $\hat{\mathbf{x}}_k, \hat{\mathbf{u}}_k \leftarrow \mathbb{E}[\mathbf{x}_k, \mathbf{u}_k | \hat{\mathbf{x}}_{k+1} = \hat{\mathbf{x}}_{k+1}]$ basierend auf dem Ausgleichsmodell

In der ersten Zeile des Algorithmus 2.2 wird $\hat{\mathbf{x}}_K$ mit dem Mittel der mit den Potenzialen gewichteten Endzustände der Trajektorien initialisiert. Dies ist eine einfache Schätzung des optimalen Endzustandes $\tilde{\mathbf{x}}_K$. Innerhalb der Schleife wird das gaußsche Ausgleichsmodell bzw. der Parameterschätzer aus der Stichprobe $[\mathbf{x}_k^{(n)}, \mathbf{u}_k^{(n)}, \mathbf{x}_{k+1}^{(n)}]$ im vorwärts

gerichteten Durchlauf erzeugt und mit der vorwärts gerichteten Glaubwürdigkeit gewichtet. Die folgenden Größen $\hat{\mathbf{x}}_k$ und $\hat{\mathbf{u}}_k$ werden mit Hilfe des Erwartungswertes des gaußschen Ausgleichsmodells geschätzt.
Basierend auf der Gewichtung maximiert der Erwartungswert sowohl die vorwärts gerichtete Glaubwürdigkeit $\hat{\mathcal{B}}_{\text{fwd}}$ als auch die Wahrscheinlichkeit, dass die Simulation den Zustand $\hat{\mathbf{x}}_k$ mit der Stellgröße $\hat{\mathbf{u}}_k$ in $\hat{\mathbf{x}}_{k+1}$ überführt. Dies entspricht der Minimierung einer Distanzmetrik $\|\hat{\mathbf{x}}_{k+1} - \mathfrak{f}(\hat{\mathbf{x}}_k, \hat{\mathbf{u}}_k, \dots)\|$. Da der Zustandsraum hohe Dimensionen aufweisen und es gegebenenfalls nur eine geringe Trajektorienanzahl geben kann, wird der Erwartungswert in regularisierter Form berechnet:

$$\mathbb{E}[\mathbf{x}_k, \mathbf{u}_k | \mathbf{x}_{k+1}] = \boldsymbol{\mu}_{(\mathbf{x}_k, \mathbf{u}_k)} + \mathbf{C}_{(\mathbf{x}_k, \mathbf{u}_k), \mathbf{x}_{k+1}} (\mathbf{C}_{\mathbf{x}_{k+1}, \mathbf{x}_{k+1}} + \lambda \mathbf{I})^{-1} (\hat{\mathbf{x}}_{k+1} - \boldsymbol{\mu}_{\mathbf{x}_{k+1}}) \tag{2.4.31}$$

Sei $\boldsymbol{\mu}_a$ der Mittelwert einer Zufallsvariable a und $\mathbf{C}_{a,b}$ bezeichne die Kovarianzmatrix der Zufallsvariablen a und b. λ bezeichnet einen Regularisierungsparameter.

2.4.2.9. Vermeidung von Null-Potenzialen und Null-Wert-Nachrichten

Aufgrund der Rechnergenauigkeit können Kostenpotenziale α_k, die hohe Kosten $\mathfrak{L}$ aufweisen, in komplexen Situationen leicht zu Null ausgewertet werden. Zur Vermeidung dieses Problems werden die Kosten so skaliert, dass die minimalen Kosten eines Prädiktionsschrittes k einen Schwellwert $\mathfrak{L}_{\min} = 20$ nicht überschreiten. Dadurch wird das Kostenpotenzial abgeflacht und Extrema sind weniger ausgeprägt.

$$\rho = \frac{\mathfrak{L}_{\min}}{\max(\mathfrak{L}_{\min}, \min\limits_{n=1,2,\dots,N} (\mathfrak{L}_k(\mathbf{z}_k^{(n)})))} \tag{2.4.32}$$

$$\alpha_k(\mathbf{z}_k^{(n)}) = \exp\{-0{,}5\rho \mathfrak{L}_k(\mathbf{z}_k^{(n)})\} \tag{2.4.33}$$

2.4.2.10. Ausführung über mehrere Aufrufe hinweg

Zur Informationsvermittlung zwischen den Aufrufen wird das Markov-Netzwerk um gerichtete Kanten erweitert, sodass es die Prädiktionsschritte und Aufrufe umfasst (siehe Abbildung 2.21). Die gerichteten Kanten bilden den kausalen und zeitlichen Informationsfluss von Aufruf $\wp - 1$ zu Aufruf $\wp$ ab; die Information kann ausschließlich vom vorherigen zum aktuellen Aufruf fließen. Dabei wird der Aufruftakt so festgelegt, das er der Zeitdifferenz von $\Delta t_1 = t_1 - t_0$ entspricht. Folglich wird die Information $\mathbf{z}_{k+1,\wp-1}$ für den Warmstart um einen Zeitschritt nach $\mathbf{z}_{k,\wp}$ verschoben. Der letzte Knoten des Aufrufs $\wp$ wird durch Extrapolation des Knoten $\mathbf{z}_{K-1,\wp}$ erzeugt. Die zentralen Eigenschaften des Markov-Netzwerks bleiben bei dieser Erweiterung erhalten, so zum Beispiel die Zyklenfreiheit.
Ferner werden folgende Operationen zwischen den Aufrufen durchgeführt:

- Für den Aufruf $\wp$ wird die erste Stellgröße $\tilde{\mathbf{u}}_{1,\wp-1}$ der besten Trajektorie $\tilde{T}_{\wp-1}$ des vorwärts gerichteten Durchlauf des vorherigen Aufrufs $\wp - 1$ angewendet, um den initialen Zustand $\mathbf{x}_{0,\wp} = \mathfrak{f}(\mathbf{x}_{0,\wp-1}, \tilde{\mathbf{u}}_{1,\wp-1})$ zu definieren. Somit entspricht eine

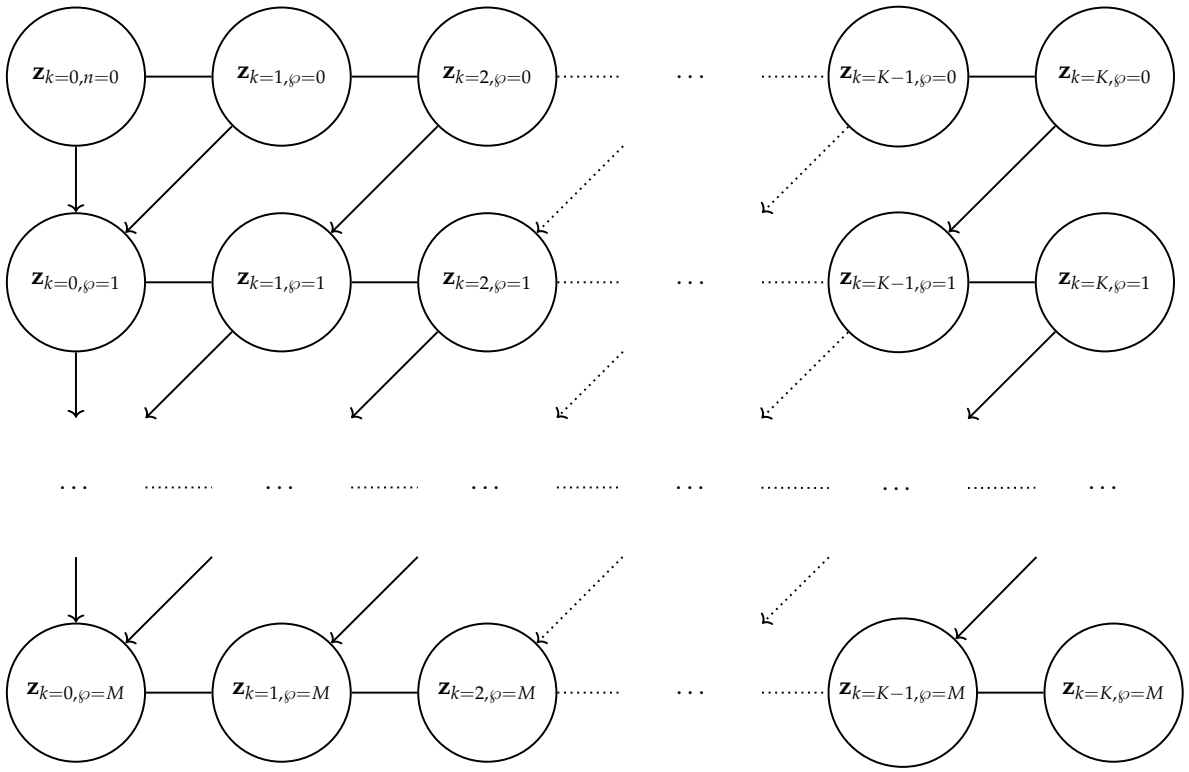


Abbildung 2.21.: Modell des Markov-Netzwerks über M Aufrufe mit Prädiktionshorizontlänge K. (vgl. [57, S. 6]) und [141]

Nachricht von $\mathbf{z}_{0,\wp-1}$ nach $\mathbf{z}_{0,\wp}$ einer Delta-Distribution, in der Regelungstechnik als „Impulsfunktion“ bezeichnet, im Zustandsraum.

- Des Weiteren werden die beste Trajektorie $\tilde{T}_{\wp-1}$ und die Ergebnisse des Algorithmus 2.2 als Warmstart für die Optimierung verwendet, wodurch bereits zwei Elemente der begrenzten Stichprobenmenge N besetzt werden. Die übrigen Stichproben werden anhand der Informationen des vorherigen Aufrufs $\wp - 1$ gezogen. Dies entspricht einer Nachricht vom Knoten $\mathbf{z}_{k+1,\wp-1}$ nach $\mathbf{z}_{k,\wp}$.

Jedoch ist zu beachten, dass die Ergebnisse des Algorithmus 2.2 nicht direkt übernommen werden können, da das berechnete gaußsche Ausgleichsmodell nicht exakt ist. Kleine Fehler in der geschätzten Stellgröße können abhängig vom Modell bereits große Abweichungen in der Zustandsfolge induzieren. Das Gaußsche Ausgleichsmodell beruht nicht auf lokaler Taylor-Expansion, welche starke Schätzfehler bei Unstetigkeiten und Multimodalität der Kostenfunktion $\mathfrak{L}$ und der Systemdynamik $\mathfrak{s}$ aufweist. Stattdessen werden einfachere Stichprobenschätzungen für die Mittelwerte und Kovarianzen verwendet und die Ergebnisse der Übertragungsfunktion $\mathfrak{f}$ des vorwärts gerichteten Durchlaufs wiederverwertet. Als Konsequenz sind die Schätzungen zwar stets ungenau, jedoch wird dies dadurch kompensiert, dass die geschätzte Stellgrößenfolge nicht vollständig auf das System $\mathfrak{s}$ gegeben wird; sondern diese Stellgrößenfolge wird als initialer Vorschlag für den nächsten Aufruf $\wp$ verwendet. Falls dieser initiale Vorschlag eine kostengünstige Trajektorie liefert, so werden sich die anderen informierten Irrfahrten (Englisch: „Informed Random Walks“) dieser Lösung annähern

und die Übertragungsgleichung $\mathfrak{f}$ erfüllen. Ferner wird oft nur das erste Element einer Stellgrößenfolge auf das System $\mathfrak{s}$ gegeben.

2.4.2.11. Regelungstechnische Adaption der Optimierung

Physikalische Bewegungen erfolgen ernergiegewahr und unter Beachtung der Trägheit, sodass die meisten Kostenfunktionen Bewegungen mit hohen Geschwindigkeiten, Momenten, Beschleunigungen und Rucken mit hohen Kosten bestrafen, um sie so zu vermeiden. Zudem wird für die Umgebung ein kontinuierliches Änderungsverhalten einer realen physischen Welt zwischen den Aufrufen angenommen, sodass folgende Stellgrößenpotenzialfunktion β_k angewendet wird:

$$\begin{aligned}\beta_k(\mathbf{u}_{k,\wp}^{(n)}) =& \mathcal{N}(\mathbf{u}_{k,\wp}^{(n)}; 0, \sigma_0^2\mathbf{C}_u)\cdot \\ & \mathcal{N}(\mathbf{u}_{k,\wp}^{(n)}; \mathbf{u}_{k-1,\wp}^{(h(i,k,-1))}, \sigma_1^2\mathbf{C}_u)\cdot \\ & \mathcal{N}(\mathbf{u}_{k,\wp}^{(n)}; 2\mathbf{u}_{k-1,\wp}^{(h(i,k,-1))} - \mathbf{u}_{k-2,n}^{(h(i,k,-2))}, \sigma_2^2\mathbf{C}_u)\cdot \\ & \mathcal{N}(\mathbf{u}_{k,\wp}^{(n)}; \boldsymbol{\mu}_{k,\wp}^{(n)}, \mathbf{C}_e)\cdot \\ & \mathcal{P}(\mathbf{u}_{k,\wp}^{(n)}|\mathbf{z}_{:,\wp-1}^{(:)})\end{aligned} \tag{2.4.34}$$

Bei der Berechnung wird das Produkt der Realisierung der Zufallszahlen der Dichtefunktionen gebildet. Mit $\mathbf{z}_{:,\wp-1}^{(:)}$ wird die mögliche Abhängigkeit zu mehren vorwärts gerichteten Prädiktionsschritten und Stichproben ausgedrückt. Die ersten drei Normalverteilungen der Gleichung 2.4.34 minimieren den quadratischen Fehler der Stellgröße und dessen ersten beiden Ableitungen. Die erste Ableitung der Geschwindigkeit definiert die Beschleunigung und die zweite Ableitung den Ruck. Als physikalischer Ruck ist die Ableitung der Beschleunigung definiert. $\mathbf{C}_e$ und $\mathbf{C}_u$ seien Diagonalkovarianzmatrizen, die in Verbindung mit σ_0, σ_1 und σ_2 als Parameter dient. Mit Hilfe der Sigmas σ_0, σ_1 und σ_2 kann eine relativer Gewichtung innerhalb der Minimierung vorgenommen werden. Falls die Kovarianzmatrix ungleich der Einheitsmatrix gewählt wird, so entspricht dies einer Gewichtung der einzelnen Stellglieder zueinander. Zum Beispiel kann die Minimierung der Rotationsgeschwindigkeitsänderung besonders gewichten werden, um möglichst glatte Trajektorien zu erhalten. Zur Verdeutlichung der zweiten und dritten Normalverteilung wird die Ableitung auf Basis der finiten Rückwärtsdifferenzen gebildet:

$$\Delta\mathbf{u}_{k,\wp} = \frac{\mathbf{u}_{k,\wp} - \mathbf{u}_{k-1,\wp}}{\Delta t_k} = 0 \tag{2.4.35}$$

$\Delta t_k = t_k - t_{k-1}$ sei die Aufrufzeitdifferenz. Zur Minimierung der Ableitung wird die Normalverteilung über $\mathbf{u}_{k,\wp}$ so gewählt, dass sie die höchste Wahrscheinlichkeit an der Stelle $\mathbf{u}_{k-1,\wp}$ aufweist. Analog dazu wird für die zweite Ableitung $\mathbf{u}_{k,\wp}$ wie folgt gewählt:

$$\Delta^2 \mathbf{u}_{k,\wp} = \frac{\frac{\mathbf{u}_{k,\wp}-\mathbf{u}_{k-1,\wp}}{\Delta t_k} - \frac{\mathbf{u}_{k-1,\wp}-\mathbf{u}_{k-2,\wp}}{\Delta t_{k-1}}}{\frac{1}{2}\Delta t_k + \frac{1}{2}\Delta t_{k-1}} = 0 \tag{2.4.36}$$
$$\approxeq_{(\Delta t_k \approxeq \Delta t_{k-1})} \frac{\mathbf{u}_{k,\wp} - 2\mathbf{u}_{k-1,\wp} + \mathbf{u}_{k-2,\wp}}{\Delta t_k^2} = 0$$

Für eine annähernd äquidistante Zerlegung der Zeitfolge t mit $(\Delta t_k \approxeq \Delta t_{k-1})$ kann die zweite Ableitung entsprechend vereinfacht werden. Daraus folgt die Wahl der höchsten Wahrscheinlichkeit für $\mathbf{u}_{k,n} = 2\mathbf{u}_{k-1,n} - \mathbf{u}_{k-2,n}$.
Der vierte Term spezifiziert eine modellspezifische normalverteilte empfohlene Stellgröße, zum Beispiel um ein System in einem Gleichgewichtszustand zu halten. Zuletzt bleibt die Repräsentation der übermittelten Information des vorherigen Aufrufs $\wp - 1$. Analog zur vorwärts und rückwärts gerichteten Nachrichtenübermittlung kann eine Nachricht von Knoten $\mathbf{z}_{k+1,\wp-1}$ zu Knoten $\mathbf{z}_{k,\wp}$ mit folgendem Übergangspotenzial beschrieben werden:

$$\Psi(\mathbf{z}_{k+1,\wp}, \mathbf{z}_{k,\wp}) = \mathfrak{N}(\mathbf{x}_{k-1,\wp}; \mathbf{x}_{k,\wp-1}, \mathbf{Q})\mathcal{N}(\mathbf{u}_{k,\wp}; \mathbf{u}_{k+1,\wp-1}, \sigma_m^2 \mathbf{C}_u) \tag{2.4.37}$$

Dadurch wird die Erkundung der Umgebung um Zustände und Stellgrößen mit einer zuvor hohen Glaubwürdigkeit verstärkt. σ_m beschreibt einen wählbaren Mutationsfaktor. Bei einer begrenzten Stichprobenanzahl N ist es jedoch empfehlenswert, vorherige Informationen für die Ziehung guter Stichproben zu verwenden, statt die Glaubwürdigkeit bereits ausgewerteter Stichproben zu vernachlässigen. Daher wird statt der Nachrichtengleichung 2.4.23 ein normalverteiltes Mischmodell (Englisch: „Gaussian Mixture Model“, kurz GMM) erstellt.

$$\mathcal{P}(\mathbf{u}_{k,\wp}^{(n)} | \mathbf{z}_{:,\wp-1}^{(:)}) \sim \sum_{j=1}^{N} w^{(j)} \mathcal{N}(\mathbf{u}_{k,\wp}^{(n)}; \mathbf{u}_{k+1,\wp-1}^{(n)}, \sigma_m^2 \mathbf{C}_u) \tag{2.4.38}$$

$$\text{mit } w^{(j)} = \hat{\mathcal{B}}_{k+1,\wp-1}^{(j)}(\mathbf{z}_{k+1,\wp-1}^{(j)}) \mathfrak{N}(\mathbf{x}_{k-1,\wp}^{(n)}; \mathbf{x}_{k,\wp-1}^{(j)}, \mathbf{Q}) \tag{2.4.39}$$

Die Gleichung 2.4.39 bewirkt, dass auf Grundlage der Glaubwürdigkeit des vorherigen Aufrufs $\hat{\mathcal{B}}_{k+1,\wp-1}^{(j)}(\mathbf{z}_{k+1,\wp-1}^{(j)})$ und des Abstandes $\|\mathbf{x}_{k-1,\wp}^{(n)} - \mathbf{x}_{k,\wp-1}^{(j)}\|$ des vorherigen Zustandes $\mathbf{x}_{k-1,\wp}^{(n)}$ von der Trajektorie $\mathbf{x}_{k,\wp-1}^{(j)}$ des vorherigen Aufrufs $\wp - 1$ die Gewichte $w^{(j)}$ des GMM angepasst werden. Dies geschieht, wenn die Stellgrößenstichprobe $\mathbf{u}_{k,\wp}^{(n)}$ erzeugt wird, um von einem Zustand $\mathbf{x}_{k-1,\wp}^{(n)}$ aus zu prädizieren. Zur Reduktion der Justierungsgparameter wird die gleiche Matrix $\mathbf{Q}$ für die Übergänge zwischen den Prädiktionsschritten und den Aufrufen genutzt, obwohl auch andere Kovarianzmatrizen möglich sind.
Um die Konvergenz zu einem lokalen Minimum zu verhindern, werden $0 < N_{\text{rand}} < N$ Stichproben der Stichprobenanzahl N für die Ziehung ohne den Term $\mathcal{P}(\mathbf{u}_{k,\wp}^{(n)} | \mathbf{z}_{:,\wp-1}^{(:)})$ bestimmt. Somit entspricht N_{rand} einem Mutationsfaktor zur Anpassung zwischen Exploration und lokaler Optimierung.

Beispiel 2.4.1: *In den Experimenten wird* $10\,\% \leq \frac{N_{rand}}{N} \leq 20\,\%$ *und* $\frac{N_{eff}^{min}}{N} = 50\,\%$ *gewählt.*

2.4.2.12. Null-Varianz-Übergänge

Die Anpassung der Kovarianzmatrix $\mathbf{Q}$ kann aufgrund der hohen Dimension des Zustandsraumes aufwändig sein. Insofern kann versucht werden $\mathbf{Q} = \mathbf{0}$ zu setzen. Gilt dies, so werden die Übergangspotenziale der Gleichungen 2.4.26 und 2.4.27 zu Eins, falls n und j zur gleichen Trajektorie gehören, sonst zu Null ausgewertet. Dadurch wird die Berechnung der Nachrichten vereinfacht:

$$\hat{m}_{\text{fwd}}(\mathbf{z}_k^{(n)}) = \alpha_{k-1}(\mathbf{z}_{k-1}^{(h(n,k,-1))})\hat{m}_{\text{fwd}}(\mathbf{z}_{k-1}^{(h(n,k,-1))}) \tag{2.4.40}$$

$$\hat{m}_{\text{bwd}}(\mathbf{z}_k^{(n)}) = \alpha_{k+1}(\mathbf{z}_{k+1}^{(h(n,k,1))})\hat{m}_{\text{bwd}}(\mathbf{z}_{k+1}^{(h(n,k,1))}) \tag{2.4.41}$$

Der Startwert der Rekursion beginnt am initialen Zustand $\mathbf{x}_{\wp} = \mathbf{x}_0$ mit der Glaubwürdigkeit und Wahrscheinlichkeit 1, da dies dem real gemessen Zustand entspricht. Dieses Verfahren reduziert die Nachrichtenvermittlung auf eine einfache „Nachbereitung“ (Englisch: „Backtracking“) der erzeugten Trajektorie und bietet daher keinen zusätzlichen Informationsgewinn. Allerdings kombiniert die lokale Verfeinerung Informationen verschiedener Trajektorien durch das Gaußsche Ausgleichsmodell, wodurch ein ähnlicher Glättungseffekt erzeugt wird. Beide Verfahren beschleunigen die Konvergenz und können unabhängig voneinander verwendet werden. Der entscheidende Unterschied zwischen rückwärts gerichteter Nachrichtenvermittlung und der lokalen Verfeinerung ist, dass ersteres ausschließlich die Glaubwürdigkeit anpasst. Dahingegen kann die Verfeinerung neue Trajektorien außerhalb der bekannten Stichprobenzustände und -stellgrößen erzeugen. Für die Auswahl von $\mathbf{Q}$ sollte beachtet werden, dass die Verwendung der Gleichungen 2.4.40 und 2.4.41 statt 2.4.26 und 2.4.27 die Möglichkeit nimmt, dass benachbarte Trajektorien aufgrund der Summierung ihre Glaubwürdigkeit gegenseitig verstärken.
Abbildung 2.22 veranschaulicht, wie wichtig die Summierung der Dichtefunktionen der Normalverteilung der Gleichung 2.4.27 innerhalb der dynamischen Programmierung von PBP ist. Dies impliziert die Wiederverwendbarkeit der Berechnungsergebnisse bereits beendeter Trajektorien; mit $\mathbf{Q} = \mathbf{0}$ werden jedoch die Erkenntnisse aus frühzeitig beendeten Trajektorien vernachlässigt.
Falls eine Trajektorie durch die Umverteilung im vorwärts gerichteten Durchlauf frühzeitig beendet wird, so können die Zustände und die Stellgrößen dieser dennoch die Glaubwürdigkeit der anderen Trajektorien im rückwärtigen Durchlauf erhöhen. Das Gaußsche Übertragungspotenzial mit $\mathbf{Q} = \mathbf{0}$ wird in Abbildung 2.22 verdeutlicht. Dabei würde die Berücksichtigung der zuvor beendeten und somit vernachlässigten Trajektorie zu einem Pfad zur Zielregion führen. Diese Region wird durch einen grünen Kreis in Abbildung 2.22 illustriert. Zudem würde die zusätzliche Glaubwürdigkeit die Stichprobenziehung des Folgeaufrufs mit Hilfe der Gleichung 2.4.39 beeinflussen. Als Konsequenz wird der Algorithmus für $\mathbf{Q} = \mathbf{0}$ spezialisiert, indem nicht das GMM der Gleichung 2.4.38 verwendet wird, denn die Gewichte beinahe aller Stichproben werden bei diesem Spezialfall zu Null ausgewertet. Die Anpassung wählt zufällig für

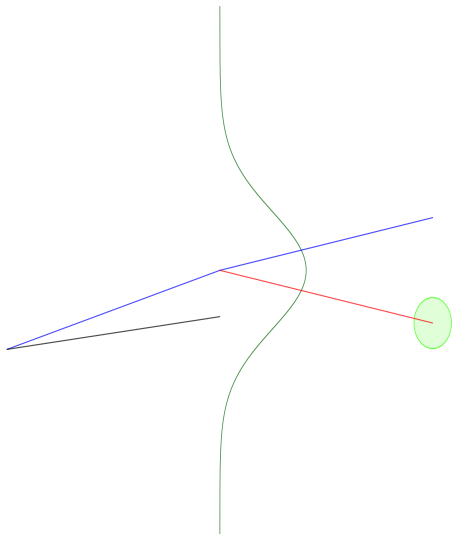


Abbildung 2.22.: Einfluss Gaußscher Übergangspotenziale (vgl. [57, S. 7] und [141]). Die blaue Trajektorie entsteht mit $\mathbf{Q} = \mathbf{0}$ und vernachlässigt die Informationen aus der schwarzen zuvor beendeten Trajektorie; wird diese jedoch beachtet, entsteht die blaurote Trajektorie, die die grüne Zielregion erreicht. Die dunkelgrüne Kurve bildet die korrespondierende Normalverteilung bzw. Dichtefunktion mit $\mathbf{Q} = \mathbf{0}$ ab.

jede Stichprobensteuerfolge $\mathbf{u}^{(n)}_{k=1,2,\dots,K,\wp}$ eine vorherige Trajektorie $\mathbf{u}^{(j)}_{k=1,2,\dots,K,\wp-1}$ mit $\mathcal{P}(\cdot) \sim \hat{\mathcal{B}}^{(j)}_K$ und ersetzt Gleichung 2.4.38 mit 2.4.34:

$$\mathcal{P}(\mathbf{u}^{(n)}_{k,\wp}|\mathbf{z}^{(:)}_{:,\wp-1}) = \mathcal{N}(\mathbf{u}^{(n)}_{k,\wp}; \mathbf{u}^{(j)}_{k+1,\wp-1}, \sigma_m^2 \mathbf{C}_u) \tag{2.4.42}$$

Diese Gleichung 2.4.42 vermittelt die vorherige Glaubwürdigkeit zwischen den Aufrufen ähnlich wie das GMM. Aufgrund von Gleichung 2.4.40 und 2.4.41 gilt für $\mathbf{Q} = \mathbf{0}$ dann $\hat{\mathcal{B}}^{(j)}_K = \prod_{k=1}^{K} \alpha_k(\mathbf{z}^{(j)}_{k,\wp-1})$. Dies ist proportional zur gewichteten Wahrscheinlichkeit der ganzen Trajektorie.
Da die Zustandsfolge im Folgeaufruf aufgrund eines veränderten Startzustandes abweichen kann, ist die Benutzung der Stellgrößenfolge unabhängig von den Zuständen inkorrekt. Allerdings sind die Abweichungen des tatsächlichen neuen initialen Zustandes und des Startzustandes der zeitlich verschobenen vorherigen Trajektorie minimal, weil aufgrund der Umverteilung die vorherigen Trajektorien einen Baum mit der Wurzel $k = 0$ bilden. Viele Trajektorien folgen zu Beginn der besten Trajektorie, die schlussendlich den Startzustand des Folgeaufrufs bildet. Es ist zu beachten, dass die vorherigen Trajektorienindizes j der Stichprobendatenstruktur übermittelt werden, sodass die vervielfältigten Trajektorien der Umverteilung, die Pfade der vorherigen Trajektorien nutzen. Nach der Abhandlung über die Informationsvermittlung zwischen den Aufrufen werden nun die Auswirkungen der Normalverteilungen der Gleichung 2.4.34 beschrieben.
Abbildung 2.23 verdeutlicht wie das Produkt der Realisierungen die Stichprobenziehung für $\mathbf{u}^{(n)}_{k,\wp}$ auf Grundlage der Dichtefunktionen geschieht. Die vorherigen Stellgrößen $\mathbf{u}^{(n)}_{k-3}$, $\mathbf{u}^{(n)}_{k-2}$, $\dots$, $\mathbf{u}^{(n)}_{k}$ werden als schwarze Trajektorie dargestellt. Die blauen und

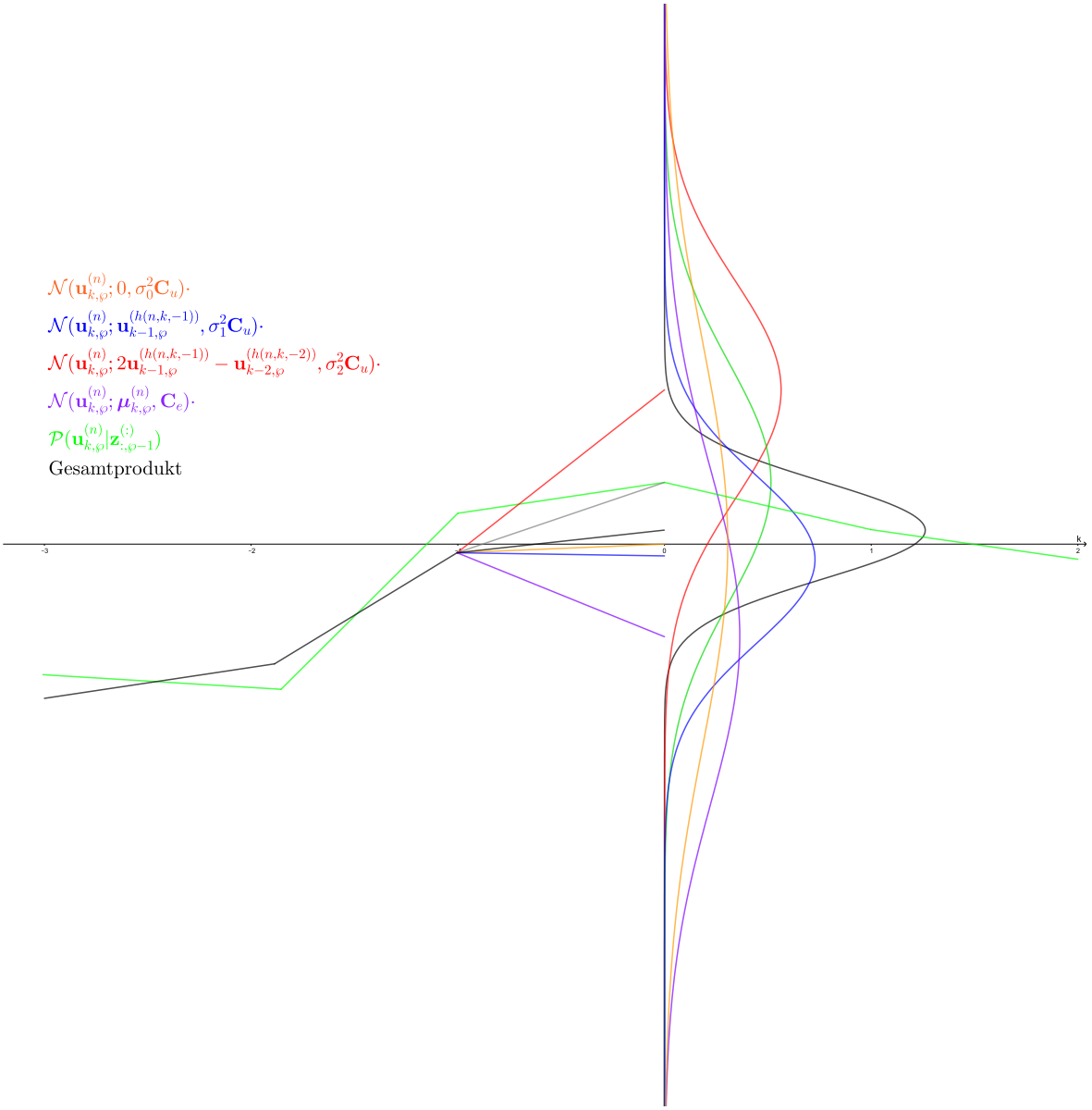


Abbildung 2.23.: Stichprobenziehung für $\mathbf{u}_{k,\wp}^{(n)}$ aus der resultierenden Gesamtverteilung (vgl. [57, S. 8] und [141]). Das orange Segment zeigt die Regelung zur Ruhestellgröße Null. Das blaue und rote Segment verdeutlicht die Minimierung der ersten und zweiten Ableitung. Das Segment in Violett verdeutlicht die empfohle modellabhängige Stellgröße. Die grüne Trajektorie bildet die Stellgrößenfolge des vorherigen Aufrufs $\wp$. Die resultierende Stellgrößenfolge wird in Schwarz dargestellt.

roten Segmente zeigen das Minimum der ersten und zweiten Ableitung der Stellgröße. Das violette Segment zeigt die empfohlene Stellgröße an und das orange Segment illustriert die Regelung auf Null. Weiterhin veranschaulicht die grüne Trajektorie die Stellgrößenfolge des vorherigen Aufrufs $\wp - 1$. Das Produkt bildet schließlich die Gesamtdichteverteilung für die Ziehung einer Stichprobe $\mathbf{u}_{k,\wp}^{(n)}$.

2.4.2.13. Verbindungen zu anderen Techniken

Die „Particle Filter Model Predictive Control“ (kurz PF-MPC) Methode prädiziert ebenfalls Stellgrößen und benutzt Umverteilungsstrategien (vgl. [100] und [175]). Es gilt, dass CPBP mit $\mathbf{Q} = \mathbf{0}$ den vorwärts gerichteten Durchlauf identisch zu PF-MPC vollführt. Jedoch besitzt PF-MPC weder einen rückwärtigen Durchlauf bzw. Nachbearbeitung noch eine lokale Verfeinerung. Beide Methoden weisen Ähnlichkeiten und Verknüpfungen zu RRT (siehe Abschnitt 2.2.1.1) oder zu „Fast Marching Tree“ (kurz FMT) auf. Letzteres wird bereits für einfache Regelungsaufgaben genutzt (vgl. [57, S. 8]) Der Unterschied besteht aber darin, dass diese Methoden im Zustandsraum und CPBP im Stellgrößenraum arbeiten. So generiert RRT einen zufälligen Zustand und sucht daraufhin eine Stellgröße, die einen Zustand erzeugt, der möglichst nah an den zufälligen Zustand heran kommt. Jedoch ist die Auswahl dieser Stellgröße für hohe Dimensionen, Nichtlinearitäten und Unstetigkeiten schwierig, denn manche inverse Kinematikabbildungen sind nicht eindeutig. Daher erkundet CPBP den Stellgrößenraum und simuliert daraus mit Hilfe der Übertragungsfunktion $\mathfrak{f}$ die Zustände (vgl. [57, S. 8]). CPBP entspricht einer Kombination der mathematischen Optimierungsmethoden „vollständige Diskretisierung“ (Englisch: „Full Collocation“) und dem „Einfachschießverfahren“ (Englisch: „Single Shooting“). Analog dazu versuchen RRT und FMT indirekt auf den „Schlupfvariablen“ (Englisch: „Slack Variable“) des „Mehrfachschießverfahrens“ (Englisch: „Multiple Shooting“) zu optimieren.

2.4.3. Erweiterung des „Control Particle Belief Propagation“-Algorithmus

Die algorithmische Erweiterung des „Control Particle Belief Propagation“-Algorithmus (kurz CPBP, siehe Unterkapitel 2.4.2) zum nun „Advanced Control Particle Belief Propagation“-Algorithmus (kurz ACPBP) genannten Verfahren besteht aus vier Teilen. Erstens ermöglicht die Umverteilung der äquidistanten Zeitschritte $\mathfrak{t}$ bzw. $\Delta\mathfrak{t}$ in eine exponentielle Folge eine bessere Wahrnehmung von unzulässigen Bereichen in der Nähe des initialen Zustandes $\overset{\diamond,\dashv}{\mathbf{x}}_0$ bei gleichzeitiger Wahrung der Horizontlänge K und der zeitlichen Horizontlänge $t_K - t_0$ (siehe Abschnitt 2.4.3.1). Zweitens wird durch die explizite Verwendung von vordefinierten modellabhängigen Trajektorien $T^{\mathfrak{f}}$ das Rauschen der Trajektorienlösungen des Optimierers auf einfachen, aber häufig verwendeten Trajektoriensegmenten vermieden (siehe Abschnitt 2.4.3.2). Die dritte Erweiterung verbessert die Konvergenz des Optimierers bei einfachen Kostenfunktionen $\mathfrak{L}$ wie der Mission der Wegpunkterreichung oder Bahnverfolgung (siehe Abschnitt 2.4.3.3). Zuletzt wird eine Totzeitkompensation integriert, die die Zustandsprädiktion verbessert (siehe Abschnitt 2.4.3.4).

2.4.3.1. Exponentielle Verteilung der Zeitfolge

CPBP basiert in seiner ursprünglichen Form auf einer äquidistanten Zerlegung des zeitlichen Horizontes $t_K - t_0$ und somit der Zeitfolge $\mathfrak{t} = [t_k]_{k=1,2,\dots,K}$. Folglich gilt für CPBP:

$$\forall k = 1, 2, \dots, K. \; \overset{\diamond}{\Delta} t_k = \text{konstant} \qquad (2.4.43)$$
$$\text{mit } [\overset{\diamond}{\Delta} t_k]_{k=1,2,\dots,K} = \Delta \mathfrak{t}$$

Diese Beschränkung vereinfacht den Informationsaustausch und die zeitliche Verschiebung der Trajektorie um einen Zeitschritt Δt_k (siehe Abbildung 2.21). Weiterhin vereinfacht sich ebenfalls die Berechnung der Ableitungen (siehe Gleichung 2.4.36). Jedoch gehen mit dieser Verteilung auch einige Nachteile einher, wenn dynamische unbekannte Umgebungen, Roboterschwärme und weitere Unsicherheiten durch Modellschätzungen und Kostenapproximationen berücksichtigt werden. Typischerweise sinkt die Präzision der Prädiktion je weiter sie in die Zukunft reicht, weil sich die Fehler über die Zeit akkumulieren. Zur Lösung dieses Problems werden die prädizierten Kosten invers zum zeitlichen Fortschritt k gewichtet, z. B. diskontiert (siehe Abschnitt 2.1.2), (vgl. [155]). Allerdings reduziert dies nur den Informationsgehalt des Kostenwertes, weil zwar der Einfluss von Stützstellen in fernerer Zukunft korrelierend mit ihrer zunehmenden Unsicherheit sinkt, aber es existiert kein Informationszugewinn im nahen zeitlichen Umfeld des Systems. Somit wird eine exponentielle Zeitreihenverteilung postuliert.

$$\overset{\diamond}{\Delta} t_k = \overset{\diamond}{\Delta} t_0 \cdot t_{\text{Schritt}}^{k-1} \qquad (2.4.44)$$

Der Vorteil dieser Verteilung ist, dass der zeitliche Nahbereich des Systems $\mathfrak{s}$ mit Hilfe des Modells $\mathfrak{f}$ deutlich feiner abgetastet wird als zeitlich entferntere Prädiktionen. Weiterhin bleibt das Verhältnis zwischen zwei aufeinander folgenden Prädiktionen k und $k+1$ stets konstant, sodass der relative Fehler durch die Verschiebung der Trajektorie um einen Zeitschritt (siehe Abschnitt 2.4.2.10) konstant ist. Zudem ist eine Gewichtung der Kosten sowie die Auswahl des zugehörigen Parameters nicht mehr nötig, da die Stützstellen mit großer zeitlicher Entfernung seltener vorkommen, sofern der maximale zeitliche Horizont identisch gewählt wird. Die Kosten sind somit indirekt exponentiell gewichtet ohne explizit den Informationsgehalt zu reduzieren. Trotz der Umverteilung behält der Algorithmus seinen Planungshorizont bei, um missionsfortschrittsverändernde Zustände erkennen zu können und lokale Minima zu umgehen (siehe Abbildung 2.24).

2.4.3.2. Modellabhängige Trajektorien

Einer der größten Nachteile und Vorteile des CPBP-Algorithmus ist seine zufällige Stellgrößenwahl anhand von angepassten Verteilungen. Dies resultiert in stets sich leicht variierenden Trajektorien für die stets selbe Kostenfunktion. Weiterhin nimmt die Wahrscheinlichkeit für eine über den Planungshorizont K konstante Stellgröße mit der Größe von K ab. Dies bedeutet für die in Abschnitt 2.3 vorgestellten Modelle, dass die Wahrscheinlichkeit für geradlinige Trajektorien oder das Stillstehen des Systems

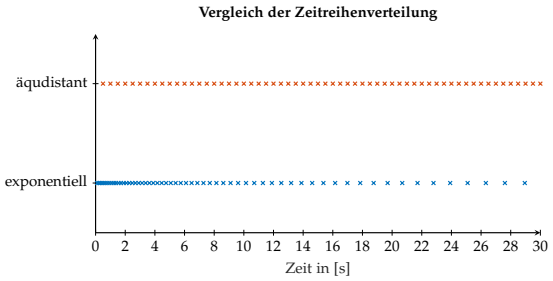


Abbildung 2.24.: Vergleichende Darstellung der Zeitreihenverteilung für $\overset{\diamond}{\Delta t}_k = 0{,}5\,\mathrm{s}$ für die äquidistante Zerlegung und $\overset{\diamond}{\Delta t}_0 = 0{,}1\,\mathrm{s}$ sowie $t_{\text{Schritt}} = 1{,}045$ für die exponentielle Zerlegung. Dies resultiert in einem Zeitverschiebungsfehler von 4,5 %. Es gilt $K = 60$.

sehr unwahrscheinlich ist. Daher werden explizit folgende Trajektorien als initialer Lösungsvorschlag zur Menge der informierten Irrfahrten $\mathcal{T}$ hinzugefügt:

$$T^{(1)} = [\mathbf{x}_k, [\overset{\dagger,\dashv}{v}_k, 0]^T]_{k=1,2,\ldots,K} \tag{2.4.45}$$

$$T^{(2)} = [\mathbf{x}_k, [0,0]^T]_{k=1,2,\ldots,K} \tag{2.4.46}$$

Die Trajektorie in Gleichung 2.4.45 beschreibt die Bewegung des Systems $\mathfrak{s}$ mit einer Vortriebsgeschwindigkeit $\overset{\dagger,\dashv}{v}_k$. Dies entspricht z. B. dem geradlinigen Fahren eines Transportschiffes mit seiner maximalen bzw. einstellbar wirtschaftlichsten Geschwindigkeit. Die zweite Trajektorie in Gleichung 2.4.46 beschreibt das Stillstehen bzw. einen Notstopp der beschriebenen Systeme. Diese Trajektorien werden im realen Einsatz oft verwendet, werden allerdings von CPBP nur mit einer sehr geringen Wahrscheinlichkeit ausgewählt. Der Algorithmus approximiert diese Trajektorien recht gut, jedoch beanspruchen dauerhafte Stellgrößensprünge Antriebsaktoren ohne Notwendigkeit (siehe Kapitel 6). Aufgrund der Informationsübertragung über Optimierungsaufrufe hinweg (siehe Abschnitt 2.4.2.10) und der Minimierung der Beschleunigung sowie des Rucks, tastet der Algorithmus mit hoher Wahrscheinlichkeit die Umgebung der vormals optimalen Trajektorie ab, welche die Trajektorie mit maximaler Vortriebsgeschwindigkeit sein könnte (siehe Gleichung 2.4.45). Folglich kann die Wahl der zweiten Trajektorie als Notstopp interpretiert werden, da diese Lösung explizit bei jedem Aufruf geprüft wird und den Algorithmus anschließend dazu bewegt um den Stillstand herum das Systemverhalten zu erkunden. Als weiterer Vorteil dieser modellabhängigen expliziten Trajektorien kann angeführt werden, dass die Umverteilung innerhalb des CPBP-Algorithmus es ermöglicht, dass Teilstücke der expliziten Trajektorien für andere Lösungstrajektorien verwendet werden. Nachteilig ist, dass die maximale Anzahl N der innerhalb eines Optimierungszyklus evaluierten Trajektorien durch die Laufzeitschranke des Systems $\mathfrak{s}$ (siehe Kapitel 6) beschränkt ist und durch die expliziten Trajektorien reduziert wird.

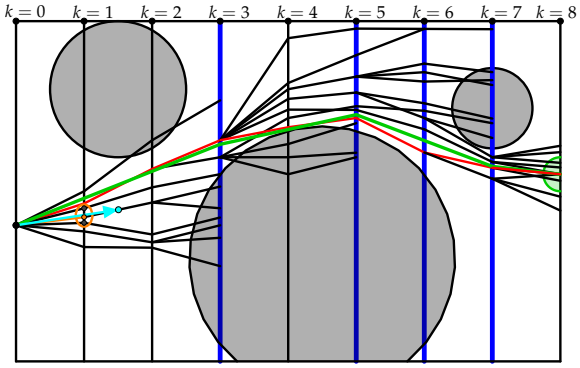


Abbildung 2.25.: Visualisierung des simulierten Gradientenabstiegs.

2.4.3.3. Simulierter Gradientenabstieg

Die dritte Erweiterung bildet ein Verfahren, ähnlich zu einem Gradientenverfahren, welches eine vollständig eigenständige Trajektorie T^g ermittelt. Dazu wird ein lokales quadratisches Einfügeverfahren (Englisch: „Infill-Criteria“) in der Umgebung der Stichprobe mit den geringsten Kosten verwendet, um den optimalen Zustand $\hat{\mathbf{x}}_k$ (siehe cyanfarbener Punkt in Abbildung 2.25) im Zeitschritt k zu schätzen. Anschließend wird die optimale Stellgröße rückwärtig approximiert, um den gewünschten Zustand von $\mathbf{x}_{k-1}$ aus zu erreichen. Für jede Iteration $k = 1, 2, \ldots, K$ wird die beste Stichprobe und ihr Zustand $\tilde{\mathbf{x}}_k$ bestimmt. Dies ist durch den orangefarbenen Punkt in Abbildung 2.25 markiert. Nach der Erzeugung aller N Stichproben wird das Minimum und seine beiden lokalen Nachbarn (siehe Gleichung 2.4.48 und 2.4.49) ausgewählt, um eine quadratische Funktion an diese Stützstellen anzupassen. Die Orientierung $\overset{\dagger}{\gamma}$ wird vorerst ignoriert, da sie bei komplexen Missionsfunktionen (siehe Kapitel 5) nur selten verwendet wird. Diese Vereinfachung wird mit $\dashv \overset{\dagger}{\gamma}$ gekennzeichnet.

$$\tilde{\mathbf{x}}_k = \min_{i \in N} \mathfrak{L}(\mathbf{x}_k^{(i)}, \mathbf{u}_k^{(i)}, \Delta_k) \tag{2.4.47}$$

$$\tilde{\mathbf{x}}_k^1 = \min_{\mathbf{x}_k \in \{\mathbf{x}_k^{(1)}, \mathbf{x}_k^{(2)}, \ldots, \mathbf{x}_k^{(N)}\} \setminus \tilde{\mathbf{x}}_k} \left\| \mathbf{x}_k - \tilde{\mathbf{x}}_k \right\|_{\dashv \overset{\dagger}{\gamma}} \tag{2.4.48}$$

$$\tilde{\mathbf{x}}_k^2 = \min_{\mathbf{x}_k \in \{\mathbf{x}_k^{(1)}, \mathbf{x}_k^{(2)}, \ldots, \mathbf{x}_k^{(N)}\} \setminus \{\tilde{\mathbf{x}}_k, \tilde{\mathbf{x}}_k^1\}} \left\| \mathbf{x}_k - \tilde{\mathbf{x}}_k \right\|_{\dashv \overset{\dagger}{\gamma}} \tag{2.4.49}$$

Das Lösen der linearen Gleichungen des Problems der kleinsten Quadrate ist der zeitaufwändigste Anteil. Falls die sich ergebende quadratische Funktion, die in Gleichung (2.4.50) dargestellt ist, ein Minimum besitzt, wird dieses als Anziehungspunkt andernfalls als ein abstoßender Zustand interpretiert. Letzteres kann nur dann auftreten, wenn das Minimum nicht zwischen seinen Nachbarn liegt, sondern einen Randfall bildet. Die meisten der verwendeten Kostenfunktionen (siehe Kapitel 5) basieren zumindest teilweise auf linearen oder quadratischen Kostenfunktionen oder können durch diese lokal approximiert werden. Folglich werden quadratische Funktionen als Einfügeverfahren für die parameterlose modellbasierte Optimierung angewendet. Dies ist eine typische Strategie für die Optimierung von komplexen und zeitaufwändigen Kostenfunktionen (vgl. [204]). Darüber hinaus kann die Lösung der Extremwerte

auf der Grundlage der Ableitungen erster und zweiter Ordnung, dargelegt in Gleichung (2.4.51), analytisch bestimmt werden:

$$\hat{\overset{\dagger}{x}}_k = \min_{x \in \mathbb{R}} \hat{\mathfrak{L}}^x_{\tilde{\mathbf{x}}_k, \tilde{\mathbf{x}}^1_k, \tilde{\mathbf{x}}^2_k}(x) = a \cdot x^2 + b \cdot x + c \tag{2.4.50}$$

$$0 = \frac{\partial \hat{\mathfrak{L}}^x_{\tilde{\mathbf{x}}_k, \tilde{\mathbf{x}}^1_k, \tilde{\mathbf{x}}^2_k}(x)}{\partial x} \Leftrightarrow x = \frac{-b}{2a} \tag{2.4.51}$$

Die Lösung für die Koordinate $\hat{\overset{\dagger}{y}}$ erfolgt analog. Um das bestimmte Extremum $\hat{\mathbf{x}}_k$, zunächst als Minimum angenommen, zu erreichen, muss ein weiteres Optimierungsproblem gelöst werden. Der nächste Zustand $\mathbf{x}^g_k$ der Trajektorie T^g der Gradientenlösung sollte möglichst nah bei der zuvor bestimmten Lösung liegen:

$$\min \|\mathbf{x}^g_k - \hat{\mathbf{x}}_k\|_{\dashv^\dagger_\gamma} \tag{2.4.52}$$

Diese Forderung wird zu einer vollständigen Lagrange-Kostenfunktion erweitert, um eine analytische Lösung zu ermitteln und einige Parameter zur Anpassung der Lösung an reale Hardware zu bieten. Die folgenden Gleichungen stellen die vollständigen Kuhn-Tucker-Bedingungen für das Kettenfahrzeugmodell dar:

$$\begin{aligned}
&\min_{\mathbf{u}^g_k = \begin{bmatrix} \overset{\dagger g}{v}_k \\ \overset{\dagger g}{\iota}_k \end{bmatrix} \in \mathcal{U}} \quad l^{\overset{\dagger g}{v}_k}_k(\mathbf{u}^g_k) = \kappa_1 \|\mathbf{x}^g_k - \hat{\mathbf{x}}_k\|^2_{\dashv^\dagger_\gamma} + \kappa_2 (\overset{\dagger g}{v}_k)^2 + \kappa_3 (\overset{\dagger g}{\iota}_k)^2 \\
&\underbrace{+\mu_1(\overset{\dagger,\dashv}{v}_- - \overset{\dagger g}{v}_k) + \mu_2(\overset{\dagger g}{v}_k - \overset{\dagger,\dashv}{v}_+) + \mu_3(\overset{\dagger,\dashv}{\iota}_- - \overset{\dagger g}{\iota}_k) + \mu_4(\overset{\dagger g}{\iota}_k - \overset{\dagger,\dashv}{\iota}_+)}_{R} \qquad (2.4.53) \\
&= \kappa_1 \left\| \begin{bmatrix} \overset{\dagger g}{x}_{k-1} - \hat{\overset{\dagger}{x}}_k \\ \overset{\dagger g}{y}_{k-1} - \hat{\overset{\dagger}{y}}_k \\ \overset{\dagger g}{\gamma}_{k-1} \end{bmatrix} + \begin{bmatrix} \overset{\dagger g}{v}_k \cos(\overset{\dagger g}{\gamma}_{k-1}) \\ \overset{\dagger g}{v}_k \sin(\overset{\dagger g}{\gamma}_{k-1}) \\ \overset{\dagger g}{\iota}_k \end{bmatrix} \Delta t_k \right\|^2_{\dashv^\dagger_\gamma} + \cdots \\
&= \kappa_1(p_1^2 + p_2^2 + p_3^2) - 2\kappa_1 \Delta t_k \overset{\dagger g}{v}_k (p_1 \cos(\overset{\dagger g}{\gamma}_{k-1}) + p_2 \sin(\overset{\dagger g}{\gamma}_{k-1})) \\
&+\kappa_1 \Delta t_k^2 (\overset{\dagger g}{v}_k)^2 - 2\kappa_1 \Delta t_k^2 p_3 (\overset{\dagger g}{\iota}_k)^2 + \kappa_1 \Delta t_k^2 (\overset{\dagger g}{\iota}_k)^2 + R + \kappa_2 (\overset{\dagger g}{v}_k)^2 + \kappa_3 (\overset{\dagger g}{\iota}_k)^2 \qquad (2.4.54) \\
&\text{mit} \begin{bmatrix} \overset{\dagger g}{x}_{k-1} - \hat{\overset{\dagger}{x}}_k \\ \overset{\dagger g}{y}_{k-1} - \hat{\overset{\dagger}{y}}_k \\ \overset{\dagger g}{\gamma}_{k-1} \end{bmatrix} = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} \qquad (2.4.55)
\end{aligned}$$

$\overset{\dagger,\dashv}{v}_-$, $\overset{\dagger,\dashv}{v}_+$, $\overset{\dagger,\dashv}{\iota}_-$ und $\overset{\dagger,\dashv}{\iota}_+$ seien die maximalen bzw. minimalen Stellgrößen. κ_2 und κ_3 steuern die Verwendung der Stellgrößen, um Stellgrößensprünge zu dämpfen. Zur

Bestimmung der optimalen Stellgröße wird die notwendige Bedingung für $\overset{\dagger^g}{v}_k$ geprüft:

$$0 = \frac{\partial l_k^{\overset{\dagger^g}{v}_k}}{\partial \overset{\dagger^g}{v}_k} \Leftrightarrow_{\mu_1,\mu_2=0} \frac{2\kappa_1\Delta_k(p_1\cos(\overset{\dagger^g}{\gamma}_{k-1}) + p_2\sin(\overset{\dagger^g}{\gamma}_{k-1}))}{2\kappa_1\Delta t_k^2 + 2\kappa_2} = \overset{\dagger^g}{v}_k$$

$$\Rightarrow \overset{\dagger^g}{v}_k = \begin{cases} \overset{\dagger}{v}_- \\ \frac{\kappa_1\Delta t_k}{\kappa_1\Delta t_k^2+\kappa_2}(p_1\cos(\overset{\dagger^g}{\gamma}_{k-1}) + p_2\sin(\overset{\dagger^g}{\gamma}_{k-1})) \\ \overset{\dagger}{v}_+ \end{cases} \tag{2.4.56}$$

Basierend auf $\overset{\dagger^g}{v}_k$ kann der Zustand $\mathbf{x}_k^g$ vollständig bestimmt werden, weil dieser unabhängig von $\overset{\dagger^g}{\iota}_k$ ist. Offensichtlich ist die Optimierung der Rotationsgeschwindigkeit $\overset{\dagger^g}{\iota}_k$ für jedes der Modelle deutlich schwieriger, weil sie oftmals keinen direkten Einfluss auf die Positionskoordinaten $\mathbf{x}_{[1],k}$ und $\mathbf{x}_{[2],k}$ des folgenden Zustands $\mathbf{x}_k$ hat. Daher wird angenommen, dass die Vortriebsgeschwindigkeit annähernd konstant bleibt, da CPBP versucht sowohl die Beschleunigung als auch den Ruck zu minimieren. Somit wird die Rotation dahingehend optimiert, dass $\mathbf{x}_{k+1}^g$ eine optimale Ausrichtung zum Zustand $\hat{\mathbf{x}}$ aufweist. Dies gilt für kleine Zeitschritte Δt_k mit der Annahme:

$$\overset{\dagger}{v}_{k-1} \approx \overset{\dagger}{v}_k \approx \overset{\dagger}{v}_{k+1} \tag{2.4.57}$$

$$\overset{\dagger}{\gamma}_{k-1} \approx \overset{\dagger}{\gamma}_k \approx \overset{\dagger}{\gamma}_{k+1} \tag{2.4.58}$$

Diese Annahme gilt in vielen Situationen, in denen die Agenten sich zu einer Zielposition begeben, wobei $\hat{\mathbf{x}}_k$ der Zielposition entspricht. Somit ist das Ziel dieser Erweiterung die Verbesserung der Trajektorien für einfachere Missionen auf Basis quadratischer Funktionen, wie z. B. die Streckenfahrten.

$$\min_{\overset{\dagger^g}{\iota}_k} l_k^{\overset{\dagger^g}{\iota}_k}(\overset{\dagger^g}{v}_k, \overset{\dagger^g}{\iota}_k) = \kappa_1\|\mathbf{x}_{k+1}^g - \hat{\mathbf{x}}_k\|^2_{\dashv\overset{\dagger}{\gamma}} + \kappa_2(\overset{\dagger^g}{v}_k)^2 + \kappa_3(\overset{\dagger^g}{\iota}_k)^2 \tag{2.4.59}$$

$$+\mu_1(\overset{\dagger,\dashv}{v}_- - \overset{\dagger^g}{v}_k) + \mu_2(\overset{\dagger^g}{v}_k - \overset{\dagger,\dashv}{v}_+)$$

$$+\mu_3(\overset{\dagger,\dashv}{\iota}_- - \overset{\dagger^g}{\iota}_k) + \mu_4(\overset{\dagger^g}{\iota}_k - \overset{\dagger,\dashv}{\iota}_+)$$

$$= \kappa_1 \left\| \begin{bmatrix} \overset{\dagger^g}{x}_k - \hat{\overset{\dagger}{x}}_k \\ \overset{\dagger^g}{y}_k - \hat{\overset{\dagger}{y}}_k \\ \cdot \end{bmatrix} + \begin{bmatrix} \overset{\dagger^g}{v}_k\cos(\overset{\dagger^g}{\gamma}_{k-1} + \overset{\dagger^g}{\iota}_k\Delta t_k) \\ \overset{\dagger^g}{v}_k\sin(\overset{\dagger^g}{\gamma}_{k-1} + \overset{\dagger^g}{\iota}_k\Delta t_k) \\ \cdot \end{bmatrix} \Delta t_{k+1} \right\|^2_{\dashv\overset{\dagger}{\gamma}} + \cdots$$

$$\kappa_1(p_1^2 + p_2^2) + 2\kappa_1\Delta t_{k+1}(\overset{\dagger^g}{v}_k)^2 - 2\kappa_1\overset{\dagger^g}{v}_k\Delta t_{k+1}(p_1\cos(\overset{\dagger^g}{\gamma}_{k-1} + \overset{\dagger^g}{\iota}_k\Delta t_k) +$$

$$p_2\sin(\overset{\dagger^g}{\gamma}_{k-1} + \overset{\dagger^g}{\iota}_k\Delta t_k)) + R + \kappa_2(\overset{\dagger^g}{v}_k)^2 + \kappa_3(\overset{\dagger^g}{\iota}_k)^2 \tag{2.4.60}$$

$$0 = \frac{\partial l_k^{\overset{\dagger}{\iota}_k^{\mathcal{g}}}}{\partial \overset{\dagger}{\iota}_k^{\mathcal{g}}} = \kappa_1 \overset{\dagger}{v}_k^{\mathcal{g}} \Delta t_{k+1} \Delta t_k (-p_1 \sin(\overset{\dagger}{\gamma}_{k-1}^{\mathcal{g}} + \overset{\dagger}{\iota}_k^{\mathcal{g}} \Delta t_k) + p_2 \cos(\overset{\dagger}{\gamma}_{k-1}^{\mathcal{g}} + \overset{\dagger}{\iota}_k^{\mathcal{g}} \Delta t_k)) + 2\kappa_3 \overset{\dagger}{\iota}_k^{\mathcal{g}} \tag{2.4.61}$$

Zur Vereinfachung der Lösung und um den Lösungsraum nicht weiter einzuschränken wird $\kappa_3 = 0$ gesetzt, wodurch eine eindeutige analytische Lösung entsteht:

$$\Rightarrow_{\kappa_3=0} \overset{\dagger}{\iota}_k^{\mathcal{g}} = \begin{cases} \overset{\dagger,\dashv}{\iota}_- \\ -\frac{\varphi}{\Delta t_k} \\ \overset{\dagger,\dashv}{\iota}_+ \end{cases} \tag{2.4.62}$$

$$\text{mit } \varphi = \arctan\left(\frac{p_1 \cos(\overset{\dagger}{\gamma}_{k-1}^{\mathcal{g}}) + p_2 \sin(\overset{\dagger}{\gamma}_{k-1}^{\mathcal{g}})}{p_1 \sin(\overset{\dagger}{\gamma}_{k-1}^{\mathcal{g}}) - p_2 \cos(\overset{\dagger}{\gamma}_{k-1}^{\mathcal{g}})}\right)$$

$$\text{und } p_1 = (\overset{\dagger}{x}_k^{\mathcal{g}} - \hat{\overset{\dagger}{x}}) \text{ und } p_2 = (\overset{\dagger}{y}_k^{\mathcal{g}} - \hat{\overset{\dagger}{y}})$$

Für abstoßende, also maximierende, Funktionen, werden p_1 und p_2 invertiert, wodurch das System in die entgegengesetzte Richtung gesteuert wird.

2.4.3.4. Totzeitkompensation

Als letzte Erweiterung wird die Totzeit, die aufgrund der Berechnungsdauer entsteht, kompensiert. Dazu ermittelt der Algorithmus seine eigene Laufzeit Δt^{tot}. Anschließend wird der neue initiale Zustand $\hat{\mathbf{x}}_0 \approx \mathbf{x}_{\wp,1} = \mathbf{x}_{\wp+1,0}$, der vom Messglied des Systems $\mathfrak{s}$ gemessen wird, bestimmt. Solange noch keine neue Stellgröße berechnet ist, wirkt die aktuelle Stellgröße $\mathbf{u}_{\wp,1}$ auf das System $\mathfrak{s}$, sodass $\hat{\mathbf{x}}_0$ nicht mehr dem Systemzustand zum Zeitpunkt der Stellgrößenanwendung entspricht. Folglich wird der initiale Zustand approximiert, um die Güte zu erhöhen:

$$\mathbf{x}_0 = \overset{\diamond}{\mathrm{f}}(\hat{\mathbf{x}}_0, \mathbf{u}_{\wp,1}, \Delta t^{\text{tot}}) \tag{2.4.63}$$

Gilt jedoch $\Delta t^{\text{tot}} > \Delta t_1$, so werden mehrere Stellgrößen appliziert bis zur Bestimmung einer neuen Stellgrößenfolge $[\mathbf{u}_{\wp+1,k}]_{k=1,2,\ldots,K}$. Die Anzahl der benötigten Stellgrößen der Folge und der Modellfunktionsaufrufe kann dabei wie folgt bestimmt werden:

$$\exists k^* \in \{1,2,\ldots,K\}.\ \nexists k' \in \{1,2,\ldots,K\}.k' > k^* \wedge \Delta t^{\text{tot}} > \sum_{k=1}^{k^*} \Delta t_k \wedge \Delta t^{\text{tot}} > \sum_{k=1}^{k'} \Delta t_k \tag{2.4.64}$$

Sei k^* jene Anzahl. So kann $\mathbf{x}_0$ rekursiv berechnet werden, bis zum Schluss eine Restzeitdifferenz $\Delta t^* = \Delta t^{\text{tot}} - \sum_{k=1}^{k^*} \Delta t_k$ verrechnet werden muss, um $\mathbf{x}_0$ möglichst genau zu approximieren. Die Totzeit ist algorithmus- und systemabhängig (siehe Kapitel 6).

2.4.4. Weitere gradientenfreie Optimierer

Es existieren generell andere stützstellenbasierte Optimierungsverfahren, mitunter Cobyla (vgl. [138]), Compass Search (vgl. [63]), der genetische Optimierer DE1220 (vgl. [15]) und die Partikelschwarmoptimierung (vgl. [73]). Jedoch nutzen sie aufgrund ihrer Generalität nicht die Systemstruktur des Systems $\mathfrak{s}$ aus, um den Suchraum und damit die Konvergenzzeit zu reduzieren. Dies wird im Abschnitt 2.5.1 genauer untersucht.

2.4.5. Gradientenbasierte Optimierungsverfahren

Der größte Nachteil der zufallsbasierten Optimierungsverfahren, wie z. B. ACPBP oder Partikelschwarmoptimierung (Englisch: „Particle Swarm Optimization“, kurz PSO, vgl. [26]), ist das Rauschen der Ausgangsgrößen, das sich in stetig leicht veränderten Trajektorien äußert. Demgegenüber bieten andere numerische Lösungsverfahren für das Optimalsteuerungsproblem der modellprädiktiven Regelung beweisbar optimale und systemstabilisierende Trajektorien (vgl. [155] und [160]). Diese Verfahren, deren Beweise und deren Erweiterungen sind in der Doktorarbeit Rösmann [155] umfassend erläutert. Jedoch gelten für diese Verfahren sowie für ihre Beweise Stetigkeitsannahmen über die Kostenfunktion, sodass die Freiheit der Missionsgestaltung (siehe Kapitel 5) stark eingeschränkt ist und daher nicht die Anforderungen aus Kapitel 1 erfüllen können (vgl. [155]).

2.5. Analyse der Optimierungsverfahren

Zur Bewertung der Funktionsfähigkeit des ACPBP-Algorithmus existieren formale Beweisansätze (siehe Anhang C.5), die für die in dieser Arbeit untersuchten Szenarien jedoch weder analytisch noch numerisch berechenbar sind. Daher wird die Funktionsfähigkeit und Qualität der Optimierung experimentell bewertet. Dazu werden verschiedene Vergleichsexperimente durchgeführt.

2.5.1. Gradientenfreie Optimierer

Um die erwartete Verbesserung von ACPBP sowie die Gesamtqualität der Optimierung zu analysieren, werden verschiedene gradientenfreie Algorithmen in fünf verschiedenen Umgebungen unter Anwendung verschiedener Missionen getestet. Es handelt sich um die folgenden Optimierungsalgorithmen, die im Kontext der Arbeit weit verbreitet sind. „Cobyla“, Kompasssuche (Englisch: „Compass Search“, kurz CS), der genetische Optimierer DE1220 und die PSO werden von PAGMO [15], einer parallelen Optimierungsbibliothek der europäischen Weltraumorganisation (Englisch: „European Space Agency“, kurz ESA), bereitgestellt. Die fünf Umgebungen (siehe Umgebung B.2.1 bis B.2.5) sind im Detail im Anhang B.2 beschrieben und bilden einen Querschnitt durch alle Komplexitätsbereiche beginnend mit einer leeren Umgebung mit einem Roboter, über eine komplexere Welt mit statischen und dynamischen Hindernissen, wie z. B. Zügen, unterschiedlichen Bodengegebenheiten, die das Fahrverhalten

Tabelle 2.1.: Experimentauswahl zur Bestimmung der empirischen Optimalität. Die Weltennummern sind im Anhang B.2 und die Missionsnummern im Kapitel 5 aufgeführt.

Simulation	Umgebung	Mission	Stichprobenanzahl	Messfrequenz
1	1 (B.2.1)	1 (5.9)	1060	1 Hz
2	1 (B.2.1)	2 (5.4)	1060	1 Hz
3	2 (B.2.2)	1 (5.9)	1060	1 Hz
4	2 (B.2.2)	2 (5.4)	1060	1 Hz
5	3 (B.2.3)	3 (5.16)	1060	1 Hz
6	3 (B.2.3)	4 (5.5)	1060	1 Hz
7	4 (B.2.4)	5, 1 (5.10, 5.9)	1060	1 Hz
8	5 (B.2.5)	6 (5.11)	1060	1 Hz
-	-	-	8480	$\sum$

beeinflussen, und Störsendern für das aufgespannte Netzwerk zur Kommunikation (siehe Kapitel 3), bis hin zu einer hoch komplexen Umgebung mit 84 dynamischen und statischen Hindernissen, die sich durch einen Schwarm mit 30 Agenten bewegen. In diesen Umgebungen werden folgende Missionskombinationen ausgeführt, die im Kapitel 5 detailliert beschrieben sind:

- Kreisförmige Pfadfolgeregelung (Mission 1, siehe Abschnitt 5.9)
- Zufällige Erkundung des Gebietes (Mission 2, siehe Abschnitt 5.4)
- Aufbau eines Liniennetzwerks (Mission 3, siehe Abschnitt 5.16)
- Aufbau eines mobilen ad hoc Netzwerks (Mission 4, siehe Abschnitt 5.5)
- Bildung einer Formationslinie (Mission 5) bei gleichzeitiger kreisförmiger Pfadfolgeregelung (Mission 1, siehe Abschnitt 5.10 und 5.9)
- Verbringung von Paketen zu Zielorten als Schwarm (Mission 6, siehe Abschnitt 5.11)

Die optimale Referenzlösung wird als die Trajektorie $\tilde{T}$ mit den geringsten Kosten $\tilde{r_{\mathfrak{L}}}$ aller Algorithmen definiert, da das tatsächliche Optimum unbekannt ist. Um eine fundierte Aussage über die Brauchbarkeit der Algorithmen treffen zu können, wird nicht nur die relative Wahrscheinlichkeit einer optimalen Lösung, sondern auch die Zuverlässigkeit der ermittelten Kosten untersucht. Dafür wird für ein gegebenes $\omega > 0$ die Zuverlässigkeitsabschätzung R der relativen Kosten $\frac{\mathfrak{L}^{\text{CPBP}}}{\mathfrak{L}^{\text{Optimum}}} \leq \omega$ bestimmt (vgl. [18]):

$$R = \mathfrak{P}\{\frac{\mathfrak{L}^{\text{CPBP}}}{\mathfrak{L}^{\text{Optimum}}} \leq \omega\} \tag{2.5.1}$$

Auf der Grundlage des Monte-Carlo-Ansatzes kann die empirische Zuverlässigkeit mit N unabhängigen, identisch verteilten Stichproben bestimmt werden.

$$\hat{R}_N = \frac{1}{N}\sum_{i=1}^{M}\Gamma\left(\frac{\mathfrak{L}^{\text{CPBP}}}{\mathfrak{L}^{\text{Optimum}}} \leq \omega\right) \text{ und } \lim_{N\to\infty}\hat{R}_N \to R \tag{2.5.2}$$

$\Gamma(\cdot) \in \{0,1\}$ bezeichnet die Indikatorfunktion und wird zu 1 ausgewertet, falls die Aussage wahr ist. Mit Hilfe der Hoeffending-Ungleichung (vgl. [18]) kann die Obergrenze des empirischen Fehlers festgelegt werden:

$$\mathfrak{P}^N\{|\hat{R}_N - R| \geq \epsilon_H\} \leq 2 \cdot \exp\{-2N\epsilon_H^2\} \tag{2.5.3}$$

Sei $\epsilon_H \in (0,1)$ eine a priori Genauigkeit und $\delta \in (0,1)$ mit $2\exp\{-2N\epsilon_H^2\} \leq \delta$ eine Konfidenzintervallbreite. Die additive Chernoff-Schranke liefert anhand dessen die erforderliche Anzahl von Stichproben:

$$N \geq \frac{1}{2\epsilon_H^2}\log\frac{2}{\delta} \tag{2.5.4}$$

Experiment 2.5.1: *Für jeden der 8 Simulationsläufe (siehe Tabelle 2.1) wird* $\epsilon_H = 0{,}05$ *und* $\delta = 0{,}01$ *gewählt, was zu* $N = 1060$ *Stichproben führt. Jede 1 s wird eine Stichprobe generiert, indem alle Algorithmen für denselben Zustand* $\mathbf{x}_0$ *zur aktuellen Simulationszeit aufgerufen werden, indem die Simulation pausiert wird, um vergleichbare Ergebnisse zu erzielen.*

Dies führt zu $N' = 8480$ Stichproben und bei gleichem δ zu einem $\epsilon'_H = 0{,}02$ insgesamt. Folglich gelten die getroffenen Aussagen mit einer Wahrscheinlichkeit von 98 % bei einer Konfidenz von 99 %.

2.5.1.1. Relative Verbesserung

Tabelle 2.2 beschreibt die relative Verbesserung der Kosten. Das heißt, welche Mehrkosten durch den Einsatz von CPBP im Vergleich zu den anderen Algorithmen erzeugt werden. Der relative Anteil gibt an, wie oft der jeweilige Algorithmus die optimale Lösung findet. Warm- und Kaltstart geben an, ob der jeweilige Algorithmus die zuvor berechnete beste Lösung des CPBP-Algorithmus als initiale Lösung für die Optimierung verwendet. Die gewählte Parametrisierung für CPBP, ACPBP und die Vergleichsalgorithmen sind im Anhang B.3 beschrieben. Die Vergleichsalgorithmen erhalten jeweils ein Kontingent von 1200 Kostenfunktionsaufrufen. Aus Tabelle 2.2 geht deutlich hervor, dass CPBP der schnellste Algorithmus ist, der gute Ergebnisse liefert. Allerdings berechnen alle Algorithmen außer CS auch bei Kaltstartbedingungen Trajektorien, die im Mittel bis zu 16,55 % weniger Kosten erzeugen. Anhand der relativen Anteile ist allerdings ebenfalls ersichtlich, dass CPBP zu 15,3 % eine Trajektorie findet, die nicht weiter verbessert werden kann und somit optimal ist. Falls die Vergleichsalgorithmen keinen initiale Lösung erhalten, so berechnet CPBP in 84,72 % die beste Lösung. Dass die relative Verbesserung dennoch so hoch ist, bezeugt, dass, falls die Trajektorien nicht optimal sind, sie deutlich verbessert werden können. Diese Behauptung wird mit den Daten der Tabelle 2.3 belegt. Obgleich der relative Anteil von ACPBP an den optimalen

Tabelle 2.2.: Optimierung mittels CPBP.

Algorithmus	Relative Verbesserung		Relativer Anteil		Max.
	Warmstart	Kaltstart	Warm	Kalt	Laufzeit
Cobyla	16,46 %	8,16 %	29,76 %	0,33 %	858 ms
CS	7,52 %	0,0 %	0,82 %	0,0 %	68 ms
DE1220	16,49 %	16,55 %	23,56 %	9,50 %	476 ms
PSO	16,35 %	15,17 %	30,56 %	5,45 %	443 ms
CPBP	-	-	15,3 %	84,72 %	279 ms

Tabelle 2.3.: Optimierung mittels ACPBP.

Algorithmus	Relative Verbesserung		Relativer Anteil		Max.
	Warmstart	Kaltstart	Warm	Kalt	Laufzeit
Cobyla	2,78 %	1,88 %	22,24 %	0,16 %	701 ms
CS	2,38 %	0,0 %	4,66 %	0,0 %	55 ms
DE1220	2,95 %	2,85 %	26,02 %	9,35 %	378 ms
PSO	3,01 %	2,74 %	33,34 %	4,16 %	383 ms
ACPBP	-	-	13,74 %	86,33 %	278 ms

Lösungen unter Kaltstartbedingungen, welche der Verwendung im Produktiveinsatz entspricht, nur um weniger als 2 % zunimmt, sinkt die relative Verbesserung sogar unter Warmstartbedingungen auf maximal 3,01 %. Somit wird klar aufgezeigt, dass die Erweiterungen von CPBP zu ACPBP eine deutliche Leistungssteigerung bewirken und dass sich ACPBP als Optimierer für die modellprädiktive Regelung eignet.

2.5.1.2. Realzeitanalyse

Ein weiterer wichtiger Aspekt der Regelung bzw. Optimierung ist die Einhaltung der Systemzeitschranke, innerhalb welcher eine neue Stellgröße bestimmt werden muss. Wie in Abbildung 2.26 illustriert, erhöht sich die Berechnungszeit durch die Erweiterungen von ACPBP nur geringfügig. Dies ist an dem 75 %-Perzentil erkennbar. Allerdings wird die Systemzeitschranke von 300 ms (siehe Kapitel 6) nie überschritten. Dies wird von keinem anderen Vergleichsalgorithmus, der unter Kaltstartbedingungen eine Verbesserung erzielt, erreicht. CS ist zwar schneller in der Berechnung, generiert jedoch keine nutzbaren Trajektorien. Theoretisch konvergiert CS gegen ein lokales Optimum bei einer unbegrenzten Anzahl an Kostenfunktionsauswertungen, die allerdings zu Vergleichszwecken auf 1200 beschränkt sind. Jedoch gilt die Konvergenz nur für differenzierbare Funktionen, deren Gradient Lipschitz-stetig ist. Dies ist nicht bei allen Missionen der Fall.
Die Laufzeitkomplexität dieser Erweiterung beträgt $O(N)$ aufgrund der Auswahl der drei Zustände $\tilde{\mathbf{x}}_k, \tilde{\mathbf{x}}_k^1$ und $\tilde{\mathbf{x}}_k^2$. Das Lösen linearer Gleichungen erfordert eine kubische Laufzeit; jedoch ist die Anzahl der Gleichungen mit drei vorgegeben, was in einer kon-

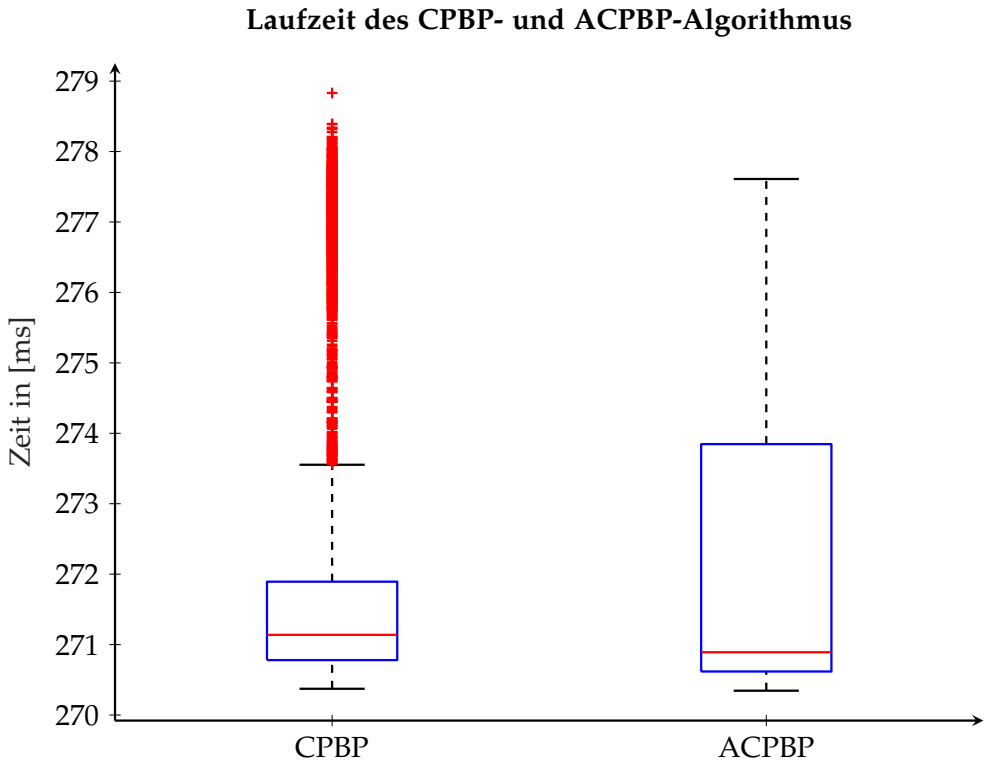
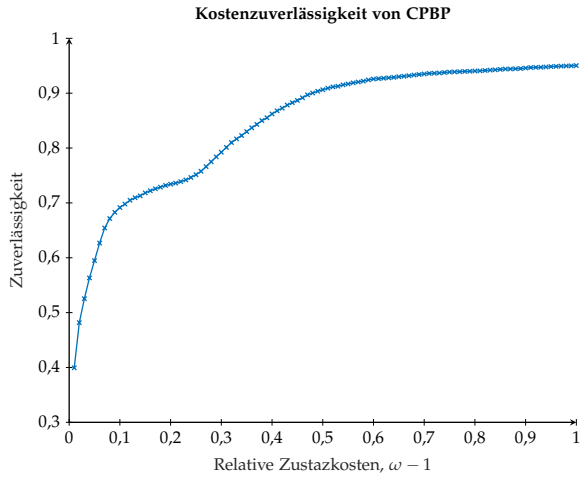


Abbildung 2.26.: Laufzeitvergleich zwischen CPBP und ACPBP.

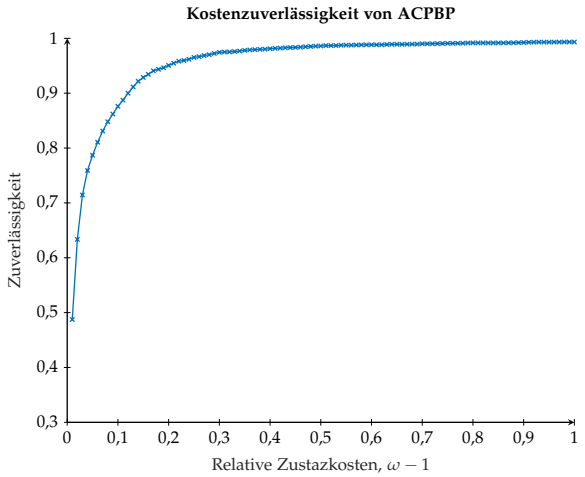
stanten Laufzeit resultiert. Die reale Laufzeit des ACPBP wird nur durch einen für Maschinensprache optimierten Quelltext im Vergleich zur CPBP-Standardimplementierung ermöglicht (siehe Abschnitt 4.1).

2.5.1.3. Zuverlässigkeitsanalyse

Als weiterer Baustein für die Anwendbarkeitsbewertung wird die Zuverlässigkeit der erzeugten Trajektorien betrachtet. Diese wird sowohl für die Kosten als auch für den örtlichen Versatz zur optimalen Lösung untersucht. Auf Grundlage dieser Zuverlässigkeitsanalyse können empirische Garantien für die Optimalität gegeben werden. Aus Abbildung 2.27 geht für CPBP klar hervor, dass mit einer Wahrscheinlichkeit von 76 % eine maximal 27 % schlechtere Lösung und mit einer Wahrscheinlichkeit von 90 % eine maximal 48 % schlechtere Lösung als die optimale gefunden wird. Wohingegen für ACPBP gilt, dass mit einer Wahrscheinlichkeit von 90 % eine maximal 12 % schlechtere Lösung und mit einer Wahrscheinlichkeit von 97 % eine maximal 25 % schlechtere Lösung als die optimale gefunden wird. Dies ist ein weiterer Beleg für die Leistungssteigerung durch die modellabhängige Erweiterung.
Um zu verstehen, ob sich die optimalen Trajektorien grundlegend von den mittels CPBP oder ACPBP berechneten Trajektorien unterscheiden, wird die Interzustandsdistanz, auch als Distanzversatz bezeichnet, ermittelt. Als Distanzmetrik wird die euklidische Norm unter Vernachlässigung der Ausrichtung $\overset{\dagger}{\gamma}$ verwendet. Die Distanz



(a) Für CPBP gilt, dass mit einer Wahrscheinlichkeit von 76 % eine maximal 27 % schlechtere Lösung als die optimale gefunden wird.



(b) Für ACPBP gilt, dass mit einer Wahrscheinlichkeit von 90 % eine maximal 12 % schlechtere Lösung als die optimale gefunden wird.

Abbildung 2.27.: Zuverlässigkeit der Kosten für den CPBP und ACPBP-Algorithmus. Die Zuverlässigkeit gibt eine Garantie für relative Zusatzkosten, die beschreiben, wie hoch die relativen Kosten im Vergleich zur optimalen Lösung sind.

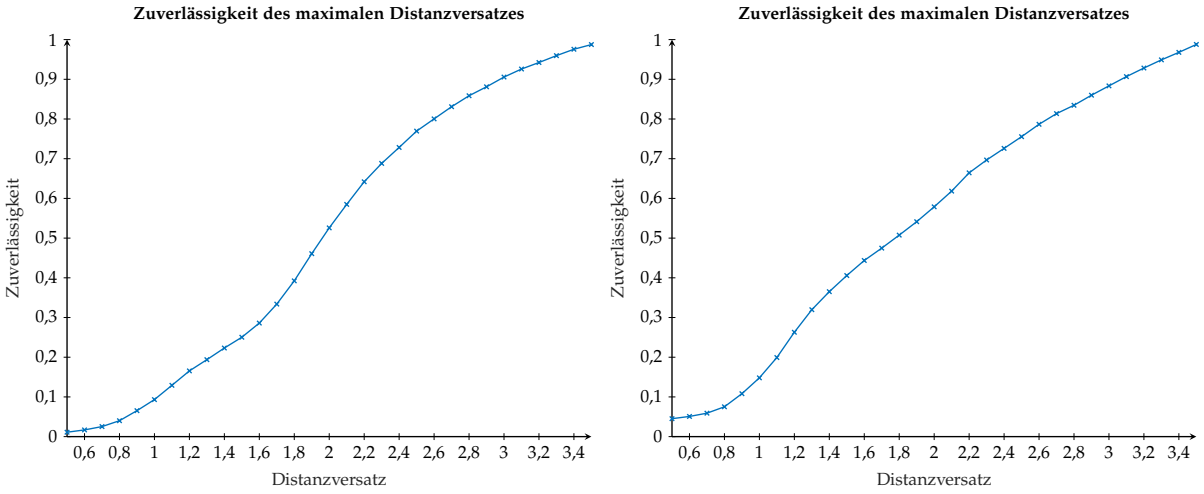
wird für jeden Planungsschritt $k = 1, 2, \ldots, K$ bestimmt:

$$d^{\text{Zuverlässigkeit}} = \max_{k=1,2,\ldots,K} \left\| \tilde{\mathbf{x}}_k - \mathbf{x}_k \right\|_{\dashv \overset{\dagger}{\gamma}} \tag{2.5.5}$$

Aufgrund der maximalen Geschwindigkeit von $\overset{\dagger}{v} = 3\,\mathrm{m\,s^{-1}}$ und einem Planungshorizont $t_K - t_0 \approx 30\,\mathrm{s}$ (siehe Kapitel 6) ergibt sich eine Trajektorienlänge von maximal 90 m. Wird dies mit dem maximalen Distanzversatz verglichen, so wird deutlich, dass sich die Trajektorien von CPBP und ACPBP um maximal 3,5 m von der optimalen Lösung unterscheiden. Der maximal mögliche Versatz beträgt 180 m. Folglich sind die generierten Trajektorien stets ähnlich zur optimalen Lösung und somit auch in der Regel brauchbar. Weiterhin zeigen die Abbildungen 2.28a und 2.28b, dass die Erweiterungen keine gänzlich neuen Trajektorien hinzufügen, sondern die Trajektorien in ihren Kosten optimieren. Es wird erneut hervorgehoben, dass die getroffenen Aussagen mit einer Genauigkeit von 98 % bei einer Glaubwürdigkeit von 99 % gelten.

2.5.2. Gradientenbasierte Optimierer

Um den ACPBP-Algorithmus umfänglich zu testen und zu bewerten, wird dieser ebenfalls mit einem hoch effizienten gradientenbasierten Optimierer für modellprädiktive Regler verglichen (vgl. [155]), der auf Hypergraphen aufbaut. Dabei werden sowohl differenzierbare als auch unstetige Kostenfunktionen verwendet, um die Vor- und Nachteile der jeweiligen Optimierungsklasse zu berücksichtigen. Weiterhin werden die zu optimierenden Systeme $\mathfrak{s}$ nicht nur auf die drei Modelle aus Abschnitt 2.3 begrenzt, sondern um bekannte, nicht lineare Vergleichssysteme erweitert.



(a) Distanzversatz für CPBP

(b) Distanzversatz für ACPBP

Abbildung 2.28.: Die Zuverlässigkeit des Distanzversatzes verdeutlicht die maximale örtliche Divergenz der erzeugten Trajektorie zur optimalen Trajektorie. Diese Differenz wird auf eine maximale Trajektorienlänge von 90 m bezogen, sodass die maximale Differenz 180 m entspricht. Zu beachten ist, das die optimale Trajektorie empirisch ermittelt wird.

2.5.2.1. Vergleichssysteme

Es werden drei weitere Systeme zum Vergleich verwendet:

Van-der-Pol-Oszillator Dies ist ein schwingungsfähiges System mit nichtlinearer Dämpfung und wird durch folgende kontinuierliche Differentialgleichung definiert:

$$\ddot{x}(t) + (x^2(t) - 1)\dot{x}(t) + x(t) = u(t) \tag{2.5.6}$$

Dies wird mit Hilfe des Zustandsvektors $\overset{\circ}{\mathbf{x}}(t) = [x(t); \dot{x}(t)]$ und der Stellgröße $\overset{\circ}{\mathbf{u}}(t) = [u(t)]$ mit $|u(t)| \leq 1$ in die Zustandsraumdarstellung überführt:

$$\dot{\overset{\circ}{\mathbf{x}}}(t) = \begin{bmatrix} \dot{x}(t) \\ -(x^2(t) - 1)\dot{x}(t) - x(t) + u(t) \end{bmatrix} \tag{2.5.7}$$

Duffing-Oszillator Dieses System wird durch die kontinuierliche Differentialgleichung 2.5.8 definiert (vgl. [122]):

$$\ddot{x}(t) + \kappa_{D_1}\dot{x}(t) + \kappa_{D_2}x(t) + \kappa_{D_3}x^3(t) = u(t) \tag{2.5.8}$$

Die Zustandsraumdarstellung mit dem Zustandsvektor $\overset{\circ}{\mathbf{x}}(t) = [x(t); \dot{x}(t)]$ und der Stellgröße $\overset{\circ}{\mathbf{u}} = [u(t)]$ mit $|u(t)| \leq 1$ ist:

$$\dot{\overset{\circ}{\mathbf{x}}}(t) = \begin{bmatrix} \dot{x}(t) \\ -\kappa_{D_1}\dot{x}(t) - \kappa_{D_2}x(t) - \kappa_{D_3}x^3(t) + u(t) \end{bmatrix} \tag{2.5.9}$$

Dabei definiert κ_{D_1} den Dämpfungsfaktor aufgrund der Viskosität, κ_{D_2} die lineare Steifigkeit und κ_{D_3} die Stärke der kubischen Nichtlinearität (vgl. [122]).

Rakete im Weltall Ein Raketensystem in der Schwerelosigkeit lässt sich über die zurückgelegte Distanz $s^{\text{Rakete}}(t)$, seine Geschwindigkeit $v^{\text{Rakete}}(t)$ und die zeitlich ändernde Masse $m^{\text{Rakete}}(t)$ mit $t \geq 0$ definieren (vgl. [156]):

$$\dot{s}^{\text{Rakete}}(t) = v^{\text{Rakete}}(t) \tag{2.5.10}$$

$$\dot{v}^{\text{Rakete}}(t) = u(t) - \frac{\kappa_{R_1}(v^{\text{Rakete}}(t))^2}{m^{\text{Rakete}}(t)} \tag{2.5.11}$$

$$\dot{m}^{\text{Rakete}}(t) = \kappa_{R_2} u^2(t) \tag{2.5.12}$$

Im Zustandsraum entspricht dies:

$$\dot{\overset{\circ}{\mathbf{x}}}(t) = \overset{\circ}{\mathbf{f}}(\overset{\circ}{\mathbf{x}}(t), \overset{\circ}{\mathbf{u}}(t)) = \begin{bmatrix} \dot{s}^{\text{Rakete}}(t) \\ \dot{v}^{\text{Rakete}}(t) \\ \dot{m}^{\text{Rakete}}(t) \end{bmatrix} = \begin{bmatrix} v^{\text{Rakete}}(t) \\ u(t) - \frac{\kappa_{R_1}(v^{\text{Rakete}}(t))^2}{m^{\text{Rakete}}(t)} \\ \kappa_{R_2} u^2(t) \end{bmatrix} \tag{2.5.13}$$

Die Stellgröße und die Geschwindigkeit werden erneut beschränkt mit $|u(t)| \leq 1$ und $-0{,}5\,\mathrm{m\,s^{-1}} \leq v^{\text{Rakete}}(t) \leq 1{,}7\,\mathrm{m\,s^{-1}}$.

2.5.2.2. Kostenfunktionen

Für die Vergleiche werden zwei Kostenfunktionen verwendet. Die quadratische Kostenfunktion wird in der MPC am häufigsten angewendet, da mit dieser Funktion Konvergenzbeweise geführt werden können und sie die Basis für optimale lineare Riccati-Regler bildet (vgl. [3]). Als Zweites wird eine stückweise definierte Funktion verwendet, die lokale und globale Minima aufweist.

Quadratische Kostenfunktion Diese Kostenfunktion wird über drei quadratische positiv definite Matrizen $\mathbf{P}$, $\mathbf{Q}$ und $\mathbf{R}$ eingestellt. Die Matrizen werden häufig ausschließlich über die Hauptdiagonale definiert und der Rest als Null angenommen. Die Diagonalform wird mit $\text{diag}([\ldots]) = \mathbf{Q}$ für eine Matrix $\mathbf{Q}$ definiert. Weiterhin bezieht sich eine quadratische Kostenfunktion stets auf eine Referenztrajektorie $\overset{\diamond}{T}^{\text{ref}} = [\overset{\diamond}{\mathbf{x}}_k^{\text{ref}}, \overset{\diamond}{\mathbf{u}}_k^{\text{ref}}]_{k=1,2,\ldots,K}$. Die Funktion kann nun beschrieben werden:

$$\begin{aligned} \overset{\circ}{\mathfrak{L}}(\overset{\circ}{\mathbf{x}}(t), \overset{\circ}{\mathbf{x}}(t), t_0, t) =& \Delta\overset{\circ}{\mathbf{x}}(t)^T \mathbf{P} \Delta\overset{\circ}{\mathbf{x}}(t) + \\ & \int_{t_0+t}^{t_0} \Delta\overset{\circ}{\mathbf{x}}(\tau)^T \mathbf{Q} \Delta\overset{\circ}{\mathbf{x}}(\tau) + \Delta\overset{\circ}{\mathbf{u}}(\tau)^T \mathbf{R} \Delta\overset{\circ}{\mathbf{u}}(\tau) d\tau \\ & \text{mit } \Delta\overset{\circ}{\mathbf{x}}(t) = \overset{\circ}{\mathbf{x}}(t) - \overset{\circ}{\mathbf{x}}^{\text{ref}}(t), \Delta\overset{\circ}{\mathbf{u}}(t) = \overset{\circ}{\mathbf{u}}(t) - \overset{\circ}{\mathbf{u}}^{\text{ref}}(t) \end{aligned} \tag{2.5.14}$$

Diese Funktion ist zeitlich nicht veränderlich und erfordert die Vorgabe einer Referenztrajektorie, weshalb sie nicht für eine komplexe Missionsbeschreibung genutzt werden kann. Da keine Endkostenabschätzung bekannt ist und diese auch nicht für jedes System hergeleitet werden kann, gilt $\mathbf{P} = \mathbf{0}$.

Unstetige Kostenfunktion Als Gegenstück zur differenzierbaren quadratischen Kostenfunktion mit einem eindeutigen lokalen und gleichzeitig globalen Optimum wird nun eine Funktion definiert, die nicht stetig ist und mehrere lokale und globale Optima aufweist:

$$d_{\mathfrak{L}}(\overset{\diamond}{\mathbf{x}}_k) = \|\overset{\diamond}{\mathbf{x}}_k - \overset{\diamond^{\text{ref}}}{\mathbf{x}}_k\|_{\dashv\gamma^{\dagger}} \tag{2.5.15}$$

$$\mathfrak{L}(d_{\mathfrak{L}}) = \begin{cases} -d_{\mathfrak{L}}, \text{für } d_{\mathfrak{L}} < 20 \\ -20, \text{für } 40 > d_{\mathfrak{L}} \geq 20 \\ -25, \text{für } 55 > d_{\mathfrak{L}} \geq 40 \\ -15, \text{für } 65 > d_{\mathfrak{L}} \geq 55 \\ -40, \text{für } 72 > d_{\mathfrak{L}} \geq 65 \\ -72 + d_{\mathfrak{L}}, \text{sonst} \end{cases} \tag{2.5.16}$$

Um jedoch mit dem verwendeten Testprogramm und der MPC Definition in der Dissertation von Rösmann [155] übereinzustimmen, basiert auch diese Funktion auf einer Referenztrajektorie. Durch die Sprungstellen und die Definitionsbereiche mit identischem Funktionswert sollte die Optimierung dieser Funktion für gradientenbasierte Optimierer nicht lösbar sein.

2.5.2.3. Konfiguration

Das quelloffene Programm (vgl. [155]) wird verwendet, um die Tests durchzuführen. Die Spezifikation des Testrechners ist im Anhang B.1.1 gelistet. Für die Vergleiche wird stets zunächst das Regelungsziel für ein System $\mathfrak{s}$ festgelegt. Ergänzende Einstellungen, wie z. B. Zustands- und Stellgrößenbeschränkungen, Kostenfunktionen und Referenztrajektorien werden nach Möglichkeit identisch gewählt. Reglerspezifische Konfigurationen werden stets angegeben.

Einstellungen gradientenbasierter Optimierung mit Hypergraphen Dieser Ansatz besitzt fünf wesentliche Einstellungsmöglichkeiten. Mit der Anzahl der Iterationen für die Lösung des Optimalsteuerungsproblems (siehe Abschnitt 2.4.1) kann die Lösungsgüte verbessert werden. Bei jeder Iteration wird eine suboptimale Lösung berechnet, welche als Startwert für die nächste Iteration genutzt wird. Jedoch steigt die Laufzeit mit jeder Iteration, sodass die Iterationsanzahl auf 1 begrenzt wird; es sei denn es wird keine Lösung gefunden, so wird diese auf 3 erhöht. Als Diskretisierungsverfahren für das gradientenbasierte Verfahren wird das Mehrfachschießverfahren eingesetzt. Dabei wird die vollständige Diskretisierung (Englisch: „Full Collocation“) verwendet, sodass jeder Schlupfvariable genau eine Stellgröße zugeordnet wird, um die dünn besetzte Struktur des Problems auszunutzen. Die Horizontlänge $t_K - t_0 = 1{,}1\,\text{s}$ wird mittels der Schrittanzahl $K = 11$ und der Zeitschrittweite $\Delta t_k = 0{,}1\,\text{s}$ eingestellt. Mit wachsender Horizontlänge nimmt die Berechnungszeit signifikant zu und führt oft zu keiner besseren Lösungsgüte. Als Integrationsverfahren wird das explizite Euler-Verfahren bzw. Euler-Cauchy-Verfahren genutzt. Die Testumgebung bietet drei Optimierer für die Lösung des Anfangswertproblems des gradientenbasierten Ansatzes. „Interior Point

Optimizer“ (kurz IPOPT, vgl. [124]) ist der leistungsfähigste der drei Optimierer und bildet in Kombination mit den Hypergraphen den Maßstab für Gradientenverfahren (vgl. [155]). Als letzte Einstellung wird die Warmstartfunktionalität für das Integrationsverfahren und den Optimierer aktiviert, sodass dieser vorherige Lösungen als initiale Schätzung für den folgenden Aufruf verwendet.

Einstellungen für ACPBP ACPBP ist ein stochastischer Optimierer dessen Parameter in Abhängigkeit zu den Zustands- und Stellgrößenbeschränkungen festgelegt werden. Aus Abschnitt 2.4.2.11 geht hervor, dass ein Stellgrößenmittelwert $\boldsymbol{\mu}_{k,\wp}^{(n)} = \boldsymbol{\mu}$ mit Matrix $\mathbf{C}_e$, die Stellgrößenstandardabweichung $\mathrm{diag}(\boldsymbol{\sigma}_0) = \sigma_0^2 \mathbf{C}_u$, die Standardabweichung der ersten Ableitung der Stellgröße $\mathrm{diag}(\boldsymbol{\sigma}_1) = \sigma_1^2 \mathbf{C}_u$ sowie die Standardabweichung der zweiten Ableitung der Stellgröße $\mathrm{diag}(\boldsymbol{\sigma}_2) = \sigma_2^2 \mathbf{C}_u$ angegeben werden müssen. Weiterhin kann der Implementierung noch die Standardabweichung des Zustandes, als $\boldsymbol{\sigma}_x$ definiert, initial mit übergeben werden. Dabei werden die Matrizen stets als Diagonalmatrizen aufgefasst und über den Vektor der Hauptdiagonalen definiert. Für die Wahl der Parameter werden folgende Schätzungen verwendet, sofern nicht anders angegeben:

- Die Zustandsstandardabweichung wird mittels der Stellgrößenstandardabweichung approximiert:

$$\boldsymbol{\sigma}_x = \begin{bmatrix} \boldsymbol{\sigma}_{[1],x} \\ \boldsymbol{\sigma}_{[2],x} \\ \boldsymbol{\sigma}_{[3],x} \end{bmatrix} = 0{,}5 \cdot \begin{bmatrix} \boldsymbol{\sigma}_{[1],0} \\ \boldsymbol{\sigma}_{[1],0} \\ \boldsymbol{\sigma}_{[2],0} \end{bmatrix} \tag{2.5.17}$$

- Die Standardabweichung der Stellgröße vergrößert sich bei Differenzierung:

$$\boldsymbol{\sigma}_1 = 2 \cdot \boldsymbol{\sigma}_0 \tag{2.5.18}$$

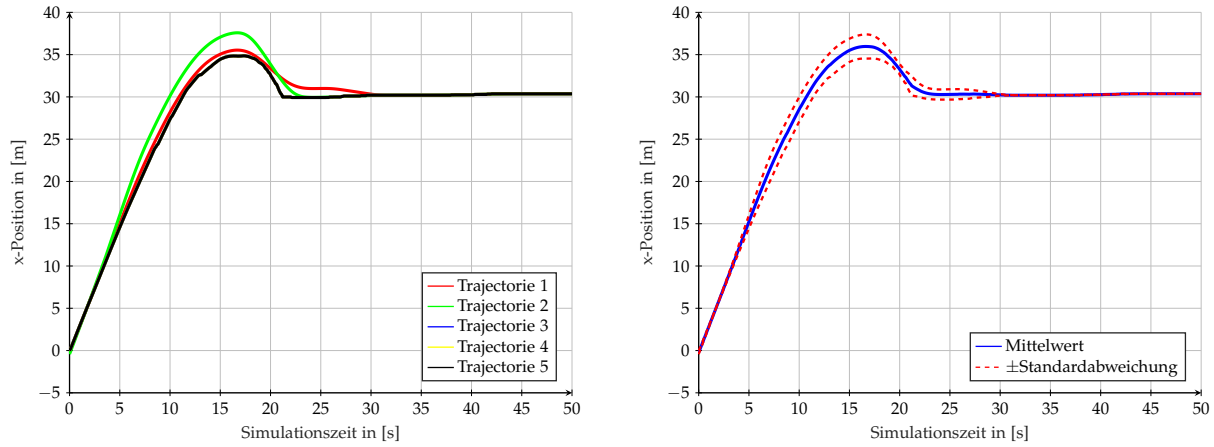
- Die erneute Differenzierung erhöht die Streuung ebenfalls:

$$\boldsymbol{\sigma}_2 = 10 \cdot \boldsymbol{\sigma}_1 \tag{2.5.19}$$

Diese Angaben bilden Richtwerte zur einheitlichen Parametrisierung und sind nicht optimal pro System und Regelungsziel ausgelegt. Falls signifikante Verbesserungen durch experimentelle Parameteranpassung erzielt werden, wird dies angegeben. Der Mutationsfaktor der Trajektorien wird auf $\sigma_m = 0{,}25$ festgelegt. Weiterhin nutzt der ACPBP-Algorithmus eine Horizontlänge $K = 60$ mit $\overset{\diamond}{\Delta} t_0 = 0{,}1\,\mathrm{s}$ und $t_{Schritt} = 1{,}045$. Dies ergibt einen Zeithorizont von $t_K - t_0 = 28{,}95\,\mathrm{s}$. Des Weiteren werden $N = 24$ Trajektorienstichproben verwendet.

2.5.2.4. Methodik der Testreihen

Für jedes der sechs Vergleichssysteme wird bei der Anwendung der quadratischen Kostenfunktion ein Referenzzustand $\mathbf{x}^{\mathrm{ref}}$ als Regelungsziel definiert und mit IPOPT, dem ACPBP- und Levenberg-Marquardt-Algorithmus gelöst. Als Bewertungsgrundlage wird die Ausregelzeit t^{Aus} für ein 5 %-Toleranzband verwendet. Die Anregelzeit und



(a) Stochastische Trajektorien

(b) Mittelwert und Standardabweichung

Abbildung 2.29.: $\overset{\dagger}{x}$-Koordinate eines Agenten mit dem Regelungsziel $\overset{\dagger}{x}^{\text{ref}} = 30\,\text{m}$.

Anstiegszeit werden kaum betrachtet, da sie für länger andauernde Missionen nicht relevant sind. Es sei denn, die Zeiten unterscheiden sich deutlich oder resultieren in einem physikalischen Effekt. Die Überschwingweite ist in der Regel so gering, dass sie zu vernachlässigen ist und dadurch Anregelzeit und Ausregelzeit identisch sind. Als Simulationszeit werden eine Zeitschrittweite von 0,1 s und eine Gesamtsimulationszeit von 50 s gewählt. Falls das System nicht ausgeregelt wird, wird die Simulationszeit verlängert. In diesen Experimenten soll die Regelungsgüte der Optimierer untersucht werden, daher wird stets nur ein System $\mathfrak{s}$ und keine Gruppen von Systemen und deren Wechselwirkung untersucht. Ein zusätzliches Kriterium für den Vergleich ist die benötigte Laufzeit der Optimierer zur Bestimmung der Stellgrößenfolge, um in der Echtzeitanwendung eingesetzt zu werden.

Behandlung stochastischer Ergebnisse Aufgrund der Stochastizität des ACPBP-Algorithmus wird die erzeugte Lösung stets von einem Rauschen überlagert. Dies führt zu stets anderen Trajektorien, sodass stets eine abweichende Lösung bei jeder Neuberechnung vorkommen kann, wie in Abbildung 2.29 dargestellt.
Die Abbildung 2.29a zeigt unterschiedliche Trajektorien für denselben Referenzzustand $\mathbf{x}^{\text{ref}}$ und für die identischen Optimierungsparameter. Für die Analyse wird der Mittelwert aus 20 Durchläufen ermittelt. Dieser wird, ebenso wie die Standardabweichung, in Abbildung 2.29b visualisiert. Aus Gründen der Übersichtlichkeit werden die Standardabweichungen und Extrema nur dann separat betrachtet, falls große Differenzen zum Mittelwert auftreten.

2.5.2.5. Regelung der Kettenfahrzeugsysteme

Die Optimierer werden anhand von vier Regelungszielen für die Kettenfahrzeuge untersucht. Die ersten beiden Experimente verwenden die quadratische Kostenfunktion aus Abschnitt 2.5.2.2. Anschließend wird ein Experiment mit IPOPT und ACPBP für die stückweise definierte Kostenfunktion durchgeführt. Die Stellgrößenbeschränkun-

gen sind stets $\mathbf{u}_- = [-0{,}5\,\mathrm{m\,s^{-1}} \quad -0{,}8976\,\mathrm{rad\,s^{-1}}]^T$ und $\mathbf{u}_+ = [3\,\mathrm{m\,s^{-1}} \quad 0{,}8976\,\mathrm{rad\,s^{-1}}]^T$. Die Reglerparameter für den ACPBP-Algorithmus werden entsprechend an die Stellgrößenbeschränkungen angepasst. Für den Mittelwert der Stellgrößen gilt $\boldsymbol{\mu} = [1{,}25\,\mathrm{m\,s^{-1}}$ $0\,\mathrm{rad\,s^{-1}}]^T$ und für die Standardabweichung $\boldsymbol{\sigma}_0 = [1{,}5\,\mathrm{m\,s^{-1}} \quad 0{,}8727\,\mathrm{rad\,s^{-1}}]^T$. Aus den Schätzregeln ergeben sich weiterhin $\boldsymbol{\sigma}_1 = [3\,\mathrm{m\,s^{-2}} \quad 1{,}7453\,\mathrm{rad\,s^{-2}}]^T$, $\boldsymbol{\sigma}_2 = [30\,\mathrm{m\,s^{-3}}$ $17{,}453\,\mathrm{rad\,s^{-3}}]^T$ und $\boldsymbol{\sigma}_x = [0{,}75\,\mathrm{m} \quad 0{,}75\,\mathrm{m} \quad 0{,}4363\,\mathrm{rad}]^T$. Diese Konfiguration wird für alle Experimente des Systems beibehalten.

Referenztrajektorie für die quadratische Kostenfunktion

Experiment 2.5.2: *Im ersten Experiment startet ein System im Zustand* $\mathbf{x}_0 = [0\,\mathrm{m} \quad 0\,\mathrm{m} \quad 0\,\mathrm{rad}]^T$ *und erhält* $\mathbf{x}^{ref} = [30\,\mathrm{m} \quad 30\,\mathrm{m} \quad 0\,\mathrm{rad}]^T$ *als Referenzzustand.*

Die quadratischen Kosten werden mit $\mathbf{Q} = \mathrm{diag}([20 \;\; 20 \;\; 0])$ und $\mathbf{R} = \mathrm{diag}([0 \;\; 0])$ definiert.

Experiment 2.5.3: *Bei einem zweiten Experiment soll ein System vom Zustand* $\mathbf{x}_0 = [0\,\mathrm{m} \quad 0\,\mathrm{m} \quad \pi\mathrm{rad}]^T$ *in den Zielzustand* $\mathbf{x}^{ref} = [30\,\mathrm{m} \quad 30\,\mathrm{m} \quad 0\,\mathrm{rad}]^T$ *überführt werden.*

Die initiale Ausrichtung des Agentensystems ist in die entgegengesetzte Richtung des Zielzustandes orientiert, sodass eine Drehung des Systems um 180° zur x-Achse durchgeführt werden muss. Dadurch werden die Optimierer auf die Funktionalität für nichtlineare Systeme getestet, da eine einfache Linearisierung im Arbeitspunkt nicht ausreichend ist, um das Ziel zu erreichen.

Lösung des ACPBP für die quadratische Kostenfunktion Abbildung 2.30 stellt die Zustandstrajektorien für das erste Experiment, das mit ACPBP optimiert wird, dar. Der Zustand des Agenten wird mit der blauen Trajektorie für die $\overset{\dagger}{x}$-Koordinate und der grünen Trajektorie für die $\overset{\dagger}{y}$-Koordinate visualisiert. Der Regler des Systems erhöht zunächst die Zustandskomponente $\overset{\dagger}{x}$ aufgrund der initialen Orientierung des Systems im Raum. Dabei beträgt die Anregelzeit etwa 11 s. Aufgrund der Minimierung der Stellgrößenänderung und der Stellgrößenbeschränkung findet ein Überschwingen statt, wodurch die $\overset{\dagger}{y}$-Komponente ihren Zielwert erreichen kann. Die Ausregelzeit beträgt 22 s, sodass das System innerhalb des Toleranzbandes von 5 % bleibt.
Abbildung 2.31 zeigt die benötigte Laufzeit des vollständigen Regelungsalgorithmus. Die initiale Berechnung dauert aufgrund des Kaltstarts mit etwa 0,0041 s am längsten. Anschließend profitieren die weiteren Berechnungen aufgrund des Warmstarts, sodass eine Rechenzeit zwischen 0,0017 s und 0,0031 s pro Zeitschritt erzielt wird. Der Mittelwert und die Standardabweichung betragen $\mu_t \approx 0{,}002\,329\,191$ s und $\sigma_t \approx 0{,}000\,438\,753$ s. Die gesamte Laufzeit über alle 500 Zeitschritte ist $t^{\mathrm{ges}} = 1{,}166\,925$ s.
Das Experiment 2.5.2 wird im Folgenden für einen Vergleich zwischen CPBP und ACPBP genutzt. Dazu zeigt Abbildung 2.32 die mittels CPBP berechnete Zustands- und Stellgrößentrajektorien. Im Vergleich zu ACPBP ist die Anregelzeit geringer jedoch entsteht kein Überschwingen. Das System ist etwa zur selben Zeit ausgeregelt,

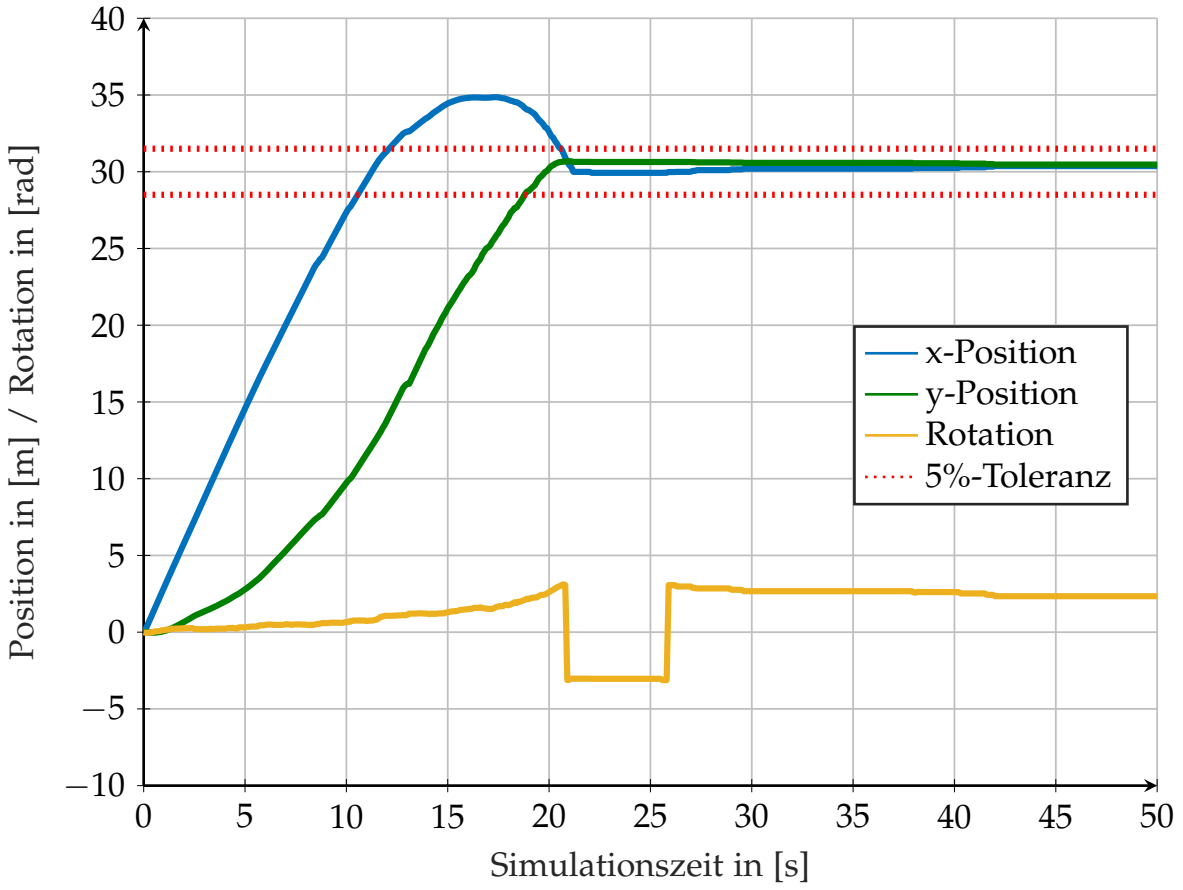


Abbildung 2.30.: Regelung eines Kettenfahrzeugsystems mittels ACPBP-Optimierers vom initialen Zustand $\mathbf{x}_0 = [0\,\text{m}\ \ 0\,\text{m}\ \ 0\,\text{rad}]^T$ zu $\mathbf{x}^{\text{ref}} = [30\,\text{m}\ \ 30\,\text{m}\ \ 0\,\text{rad}]^T$. Die einzelnen Zustandskomponenten werden jeweils getrennt dargestellt.

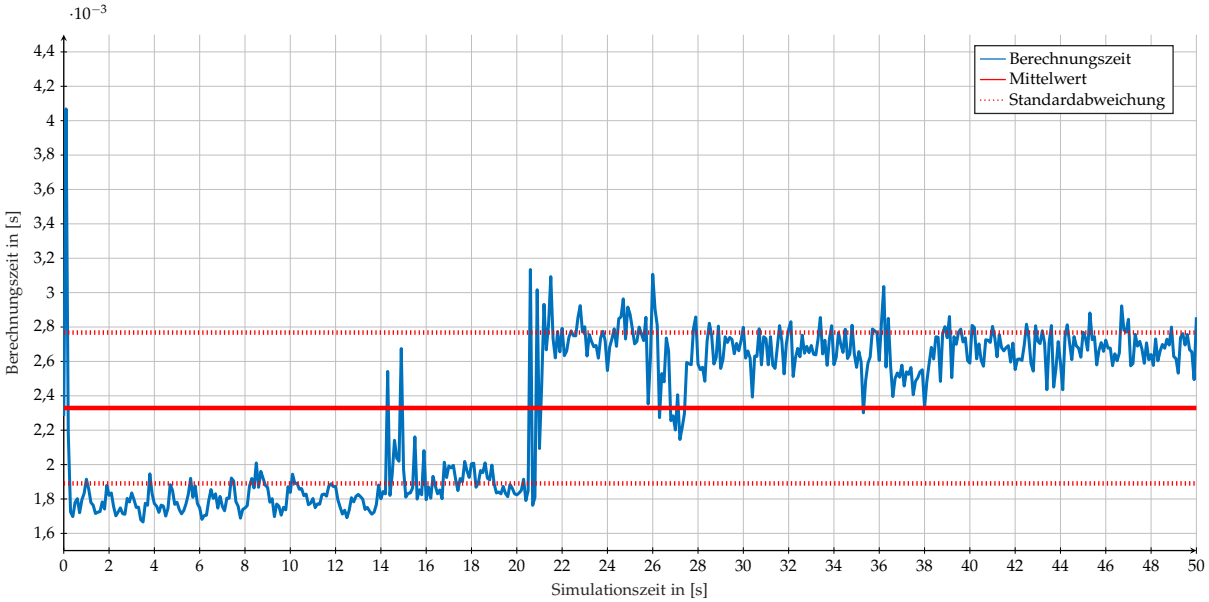
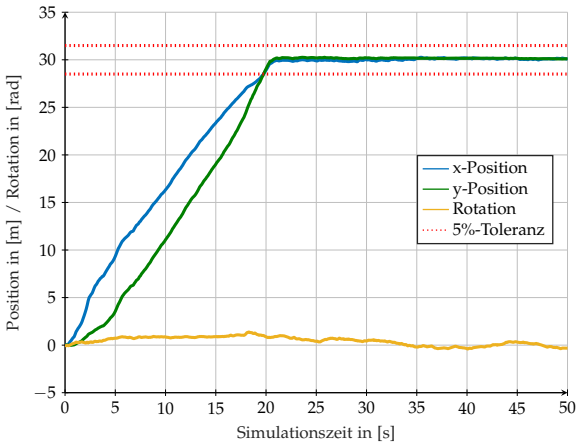
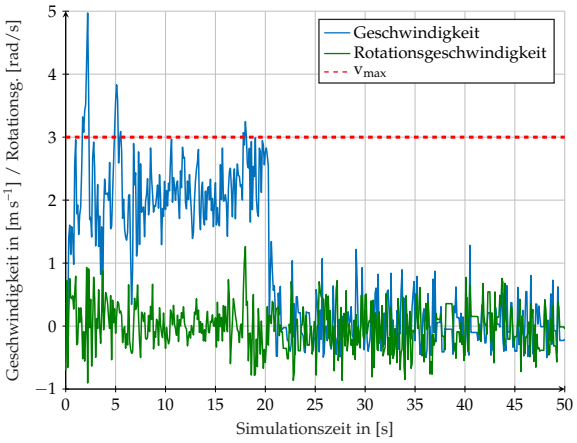


Abbildung 2.31.: Laufzeit der Kettenfahrzeugregelung im Experiment 2.5.2 mit ACPBP.

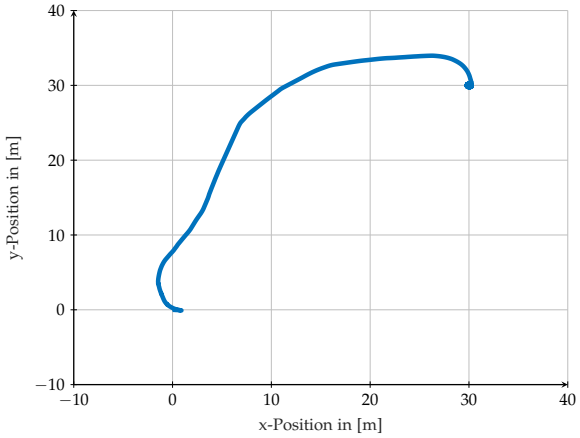


(a) Zustandstrajektorien berechnet mittels CPBP

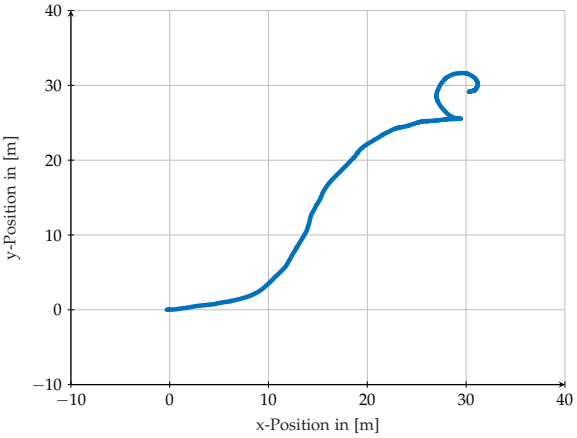


(b) Stellgrößentrajektorien berechnet mittels CPBP

Abbildung 2.32.: Lösung der ersten Regelungsaufgabe für Landfahrzeugroboter mit CPBP.



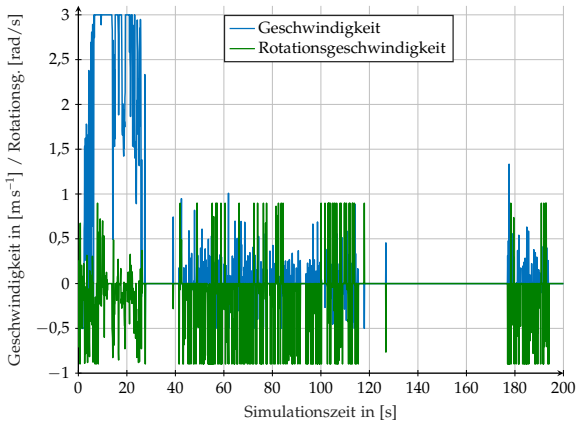
(a) Fahrtweg mit kürzerer Ausregelzeit



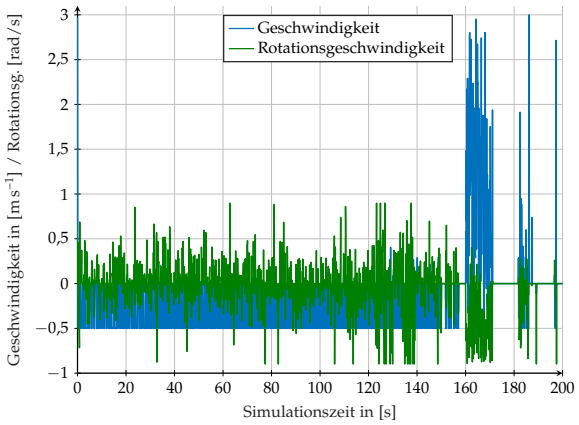
(b) Fahrtweg mit längerer Ausregelzeit

Abbildung 2.33.: Fahrtweg der Kettenfahrzeuge.

allerdings werden die Stellgrößenbeschränkungen verletzt. Es ist zu erkennen, dass die Stellgrößenbeschränkung für die Geschwindigkeit $\overset{\dagger,\dashv}{v}_{+} = 3\,\mathrm{m\,s^{-1}}$ nicht eingehalten wird. Sie wird mehrmals zu verschiedenen Zeiten überschritten, zum Teil sogar deutlich mit etwa $5\,\mathrm{m\,s^{-1}}$. Daher wird die Optimierung mit CPBP nicht weiter betrachtet.
Bei der Durchführung des Experimentes 2.5.3 werden grundlegend verschiedene Trajektorien generiert (siehe Abbildung 2.33). Daher wird die Simulationszeit auf 200 s erhöht, da der Regler ansonsten das System teilweise noch nicht zum Referenzzustand ausregelt.
Anhand der zugehörigen Stellgrößentrajektorien in Abbildungen 2.34 können die beiden Lösungen nachvollzogen werden. Die Stellgrößentrajektorie, die zur kürzeren Ausregelzeit führt, setzt eine hohe Geschwindigkeit $\overset{\dagger}{v}$ und Rotation $\overset{\dagger}{\iota}$ auf das System, sodass es, trotz der Ausrichtung des Systems in negative $\overset{\dagger}{x}$-Richtung, die Ausrichtung zügig korrigiert und sich dem Zielzustand nähert. Die andere Lösung nimmt keine Kostenerhöhung in Kauf, indem der Systemzustand $\mathbf{x}$ zunächst von der Referenz

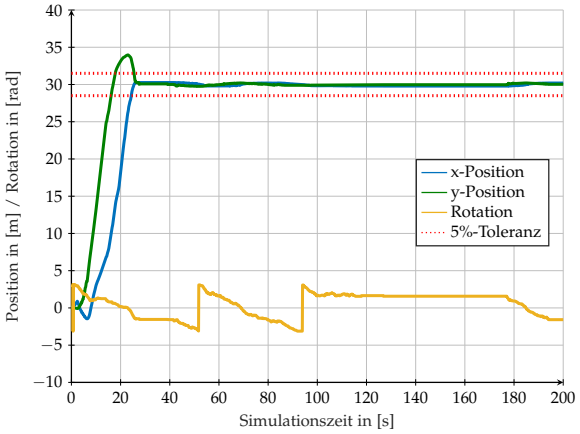


(a) Stellgrößen für kürzere Konvergenzzeit

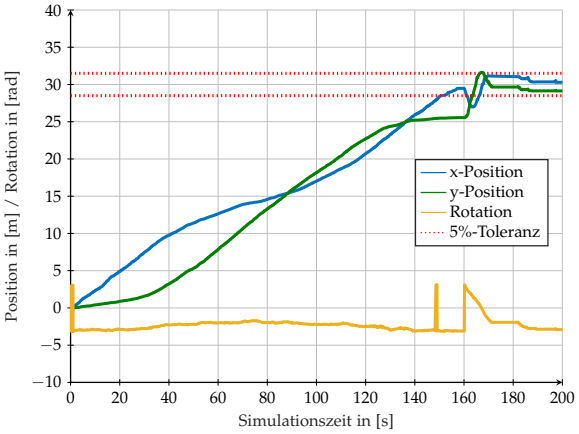


(b) Stellgrößen für längere Konvergenzzeit

Abbildung 2.34.: Stellgrößen der unterschiedlichen Zustandstrajektorien für Kettenfahrzeuge.



(a) Lösung mit kürzerer Ausregelzeit

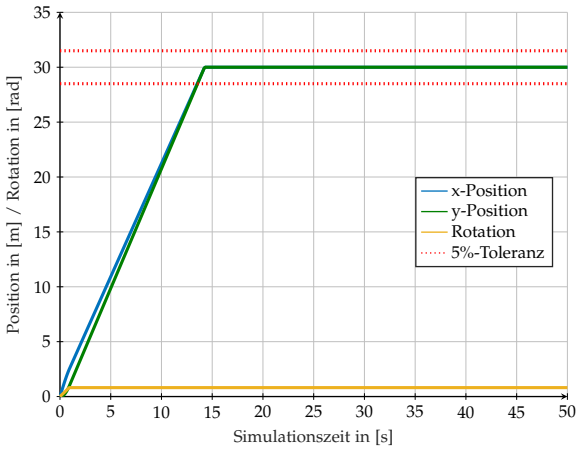


(b) Lösung mit längerer Ausregelzeit

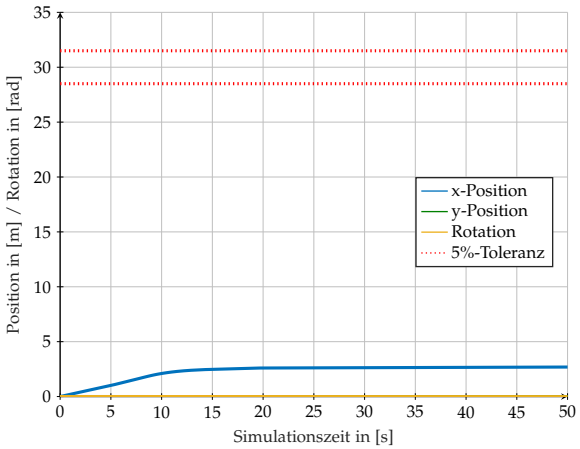
Abbildung 2.35.: Abweichung der Trajektorien bei Experiment 2.5.3.

entfernt wird, sondern korrigiert die Ausrichtung invers und verwendet die minimale Stellgröße $\overset{\dagger,\dashv}{v}_-$, um den Referenzzustand zu erreichen. Kurz vor dem Erreichen der Referenz wechselt der Regler das Verhalten und regelt das System ebenfalls im Kreisbogen mit positiver Stellgröße $\overset{\dagger}{v}$ zum Zielzustand. Die letztere Trajektorie kann anschaulich als Rückwärtsfahrt zum Ziel interpretiert werden, wohingegen die erstere Trajektorie stets vorwärts gerichtete Fahrt zum Ziel beschreibt. Aufgrund der unterschiedlichen Stellgrößenbeschränkungen $\overset{\dagger,\dashv}{v}_+$ und $\overset{\dagger,\dashv}{v}_-$ ergeben sich unterschiedliche Ausregelzeiten, die in Abbildung 2.35 deutlich erkennbar sind.

Gradientenbasierte Lösung der Regelungsziele mit quadratischer Kostenfunktion Die Einstellungen der gradientenbasierten Optimierer werden, wie in Abschnitt 2.5.2.2 beschrieben, gewählt. Das Ergebnis des Experimentes 2.5.2 mit den Optimierern IPOPT und Levenberg-Marquardt (kurz LM) ist in Abbildung 2.36 dargestellt. Dabei zeigt Abbildung 2.36a die mit IPOPT bestimmten Trajektorien. Diese werden innerhalb 14 s ohne Überschwingen ausgeregelt. Die Laufzeit dieser Lösung beträgt insgesamt $t^{\text{ges}} =$



(a) Ergebnis des Experiments 2.5.2 mittels IPOPT

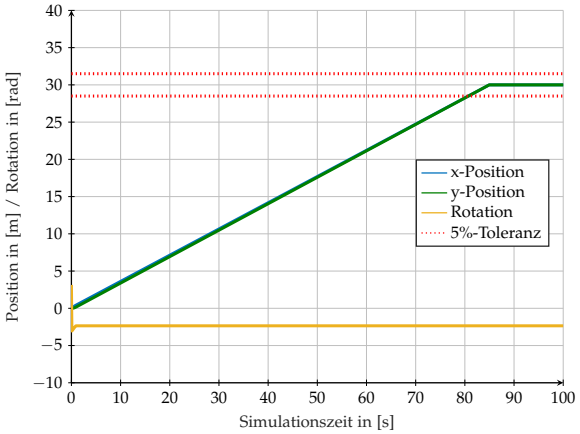


(b) Ergebnis des Experiments 2.5.2 mittels LM

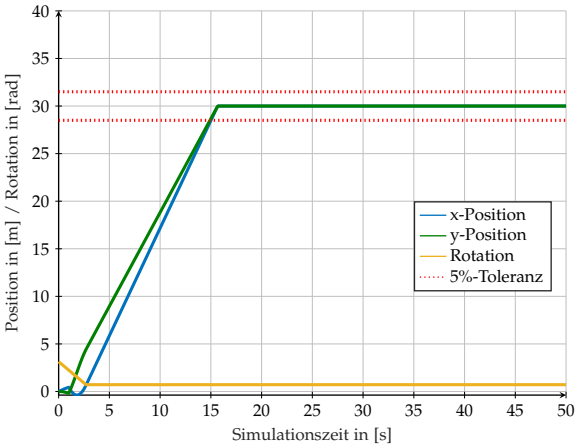
Abbildung 2.36.: Gradientenbasierte Lösung des Experimentes 2.5.2 für Kettenfahrzeugsysteme.

3,184 937 s mit dem Mittelwert $\mu_t = 0{,}006\,357\,16$ s und einer Standardabweichung von $\sigma_t = 0{,}001\,167\,6$ s. Im Gegensatz dazu ist der LM nicht in der Lage das System auszuregeln. In Abbildung 2.36b ist dargestellt, dass sich das System des Agenten 2,6 m in $\overset{\dagger}{x}$-Richtung bewegt und danach ruht. Eine Erhöhung der Iterationen für die Berechnung und Erweitern des Horizonts verbessern das Ergebnis nicht.
Abbildung 2.37 stellt die mit IPOPT berechneten Zustandstrajektorien und die auf das System angewendete Stellgrößenfolge des Experiments 2.5.3 mit unterschiedlichen Horizontlängen dar. Zum einen wird eine Horizontlänge $t_K - t_0 = 1{,}1$ s und zum anderen die Länge $t_K - t_0 = 1{,}6$ s gewählt. Aus den Abbildungen 2.37a und 2.37c geht hervor, dass die kürzere Horizontlänge ebenfalls, wie in Abbildung 2.35b, zur Stellgrößenwahl $\overset{\dagger}{v} = \overset{\dagger,\dashv}{v}_{-}$ führt. Diese Wahl induziert erneut eine längere Ausregelzeit. Es ist anhand der Abbildung 2.37b zu erkennen, dass der Horizont zu kurz gewählt ist, da eine Erweiterung des Horizontes mit ACPBP vergleichbare Ergebnisse erreicht. Mit der Horizontlänge von mindestens $t_K - t_0 = 1{,}6$ s erkennt der Optimierer das lokale Minimum bei der Wahl der negativen Stellgröße $\overset{\dagger}{v}$ und optimiert zum globalen Optimum hin mittels einer positiven Stellgröße $\overset{\dagger}{v}$. Diese Erkenntnis wird durch die reduzierte Ausregelzeit belegt. Die benötigte Zeit für die Berechnung mit einem Horizont von $t_K - t_0 = 1{,}1$ s beträgt $t^{\text{ges}} = 4{,}617$ s mit dem Mittelwert $\mu_t = 0{,}004\,61$ s und der Standardabweichung $\sigma_t = 0{,}001\,68$ s. Die Laufzeit mit einem Horizont der Länge $t_K - t_0 = 1{,}6$ s liegt bei $t^{\text{ges}} = 5{,}430\,89$ s mit dem Mittelwert $\mu_t = 0{,}010\,84$ s und der Standardabweichung $\sigma_t = 0{,}002\,98$ s.

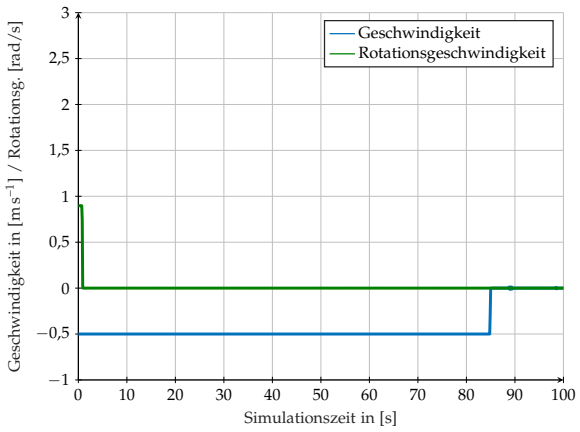
Regelung der Kettenfahrzeuge mit unstetiger Kostenfunktion Das Ziel der Regelung mit einer quadratischen Kostenfunktion ist das Erreichen bzw. Folgen einer Referenztrajektorie T^{ref}. Durch die Wahl $\mathbf{R} = \mathbf{0}$ wird die Stellgröße nicht in die Kosten mit einbezogen und kann frei gewählt werden. Verglichen damit entspricht die unstetige Kostenfunktion (siehe Abschnitt 2.5.2.2) einer Missionsbeschreibung, die das Optimie-



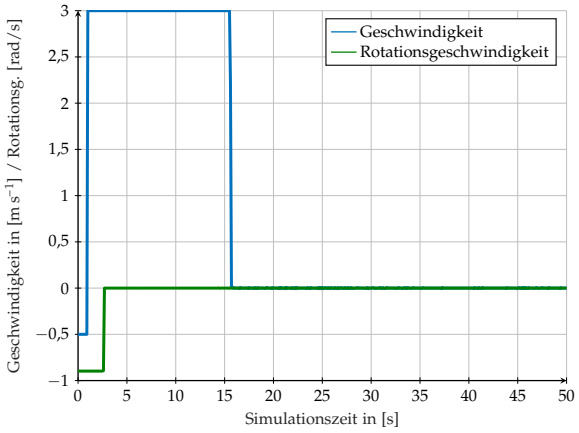
(a) Zustandstrajektorie für $T = 1{,}1\,\mathrm{s}$



(b) Zustandstrajektorie für $T = 1{,}6\,\mathrm{s}$

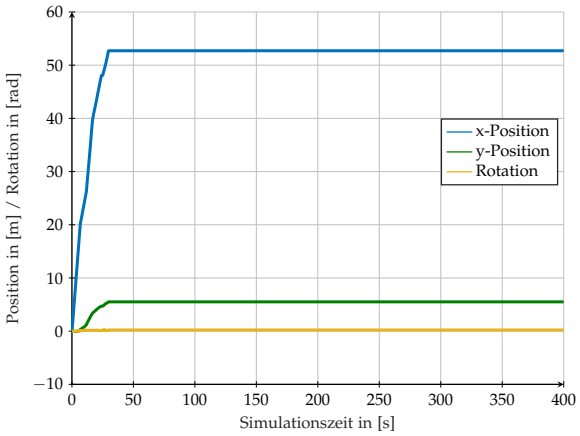


(c) Stellgrößentrajektorie für $T = 1{,}1\,\mathrm{s}$

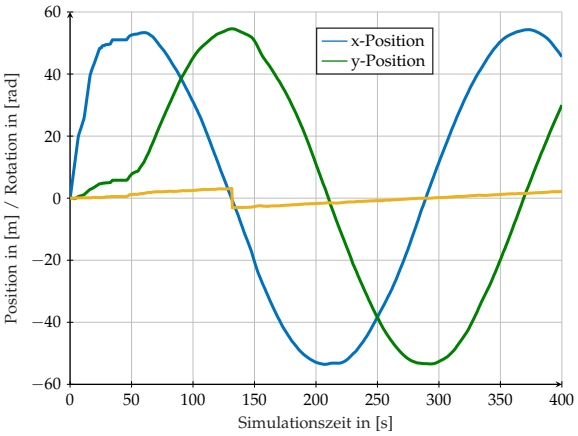


(d) Stellgrößentrajektorie für $T = 1{,}6\,\mathrm{s}$

Abbildung 2.37.: Lösung der zweiten Regelungsaufgabe für Landfahrzeugroboter mit IPOPT.



(a) 1. Lösung des ACPBP für Experiment 2.5.4



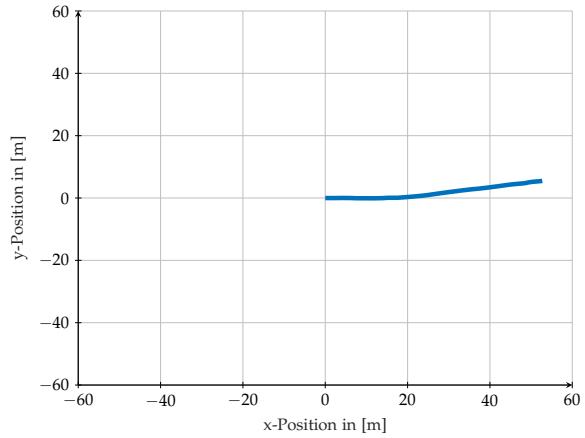
(b) 2. Lösung des ACPBP für Experiment 2.5.4

Abbildung 2.38.: Regelung mit ACPBP und unstetiger, stückweise konstanter Kostenfunktion.

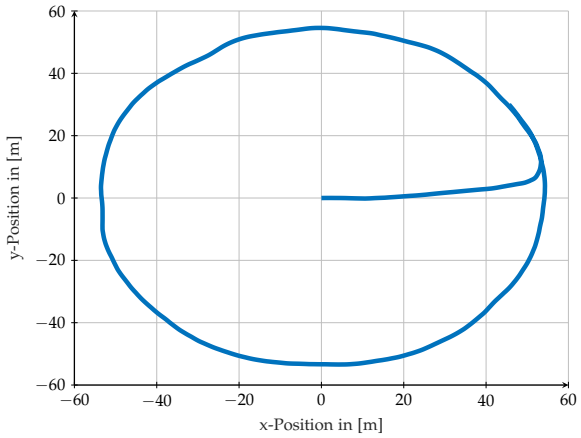
rungsziel implizit durch die Kosten beschreibt. Folglich wird nun $\mathbf{x}^{\text{ref}} = \mathbf{x}_0$ gewählt, sodass das System sich, basierend auf der Distanz vom Initialzustand, entfernen muss und die optimale Distanz aus der Kostenfunktion ermitteln muss. Diese Distanz soll dann vom Regler gehalten werden.

Experiment 2.5.4: *Das System in diesem Experiment soll die Funktion 2.5.16 minimieren.*

Die besondere Herausforderung für die Optimierer besteht in der Unstetigkeit und der stückweisen Konstanz der Funktion, weil diese keine Optimierungsrichtungsinformationen bieten. Die Regelung mit ACPBP generiert erneut zwei unterschiedliche Lösungen, die in Abbildung 2.38 dargestellt sind.
Die erste Lösung des ACPBP transferiert das System vom initialen Zustand in einen stabilen Endzustand, in dem das System ruht (siehe Abbildung 2.38a). Die zweite Lösung, visualisiert in Abbildung 2.38b, resultiert in einem oszillierenden Endzustand. Dies wird zur Verdeutlichung und Erhöhung des Verständnisses in Abbildung 2.39 als Positionstrajektorien abgebildet.
In Abbildung 2.39a ist deutlich zu erkennen, dass möglichst schnell eine Position mit möglichst großer $\overset{\dagger}{x}$- und $\overset{\dagger}{y}$-Koordinate aufgesucht und gehalten wird. Dem gegenüber wird in Abbildung 2.39b zunächst die gewünschte Distanz nur mittels der Erhöhung der $\overset{\dagger}{x}$-Koordinate angestrebt. Anschließend zirkuliert das System auf einem Einheitskreis mit der gewünschten Distanz, da die Stellgröße nicht mit in der Kostenfunktion inbegriffen ist. Als Konsequenz ist hervorzuheben, dass es somit zwei gleichwertige Lösungen gibt, die dieselben Kosten induzieren. Dies belegt Abbildung 2.40, die die Kosten beider Trajektorien über die Distanz skizziert.
Die resultierenden Kosten sind für beide Lösungen identisch und entsprechen dem Kostenverlauf der in Abschnitt 2.5.2.2 definierten Funktion 2.5.16. Jedoch wird das globale Minimum der Funktion bei einer Distanz zwischen $65 \leq d_{\mathfrak{L}} \leq 72$ nicht erreicht. Dennoch gilt das Regelungsziel als teilweise erfüllt, weil zumindest ein lokales Minimum erreicht wird.
Die verwendete Berechnungszeit beträgt $t^{\text{ges}} = 9{,}859\,07\,\text{s}$, $\mu_t = 0{,}002\,46\,\text{s}$ und $\sigma_t =$

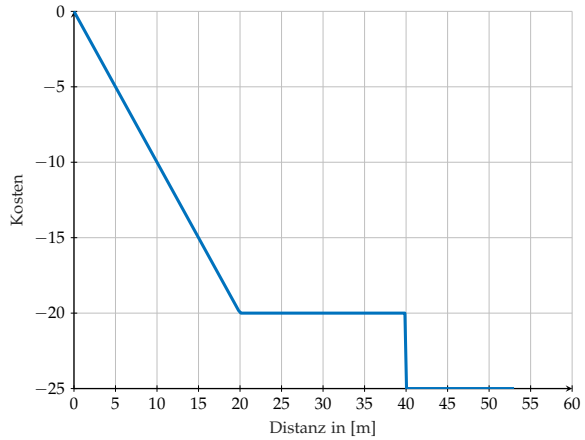


(a) Trajektorie der ersten Lösung des ACPBP für Experiment 2.5.4

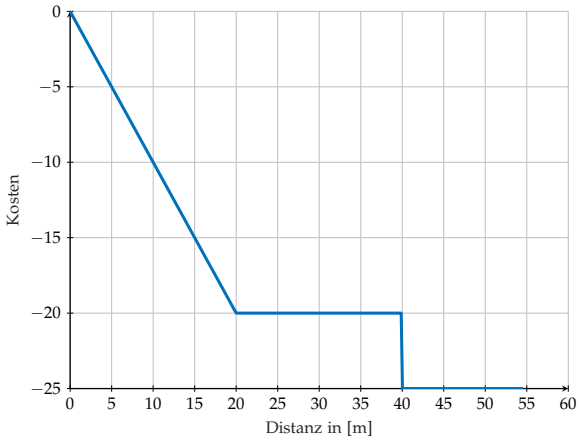


(b) Trajektorie der zweiten Lösung des ACPBP für Experiment 2.5.4

Abbildung 2.39.: Positionstrajektorien der Regelung mittels ACPBP für die unstetige Kostenfunktion.

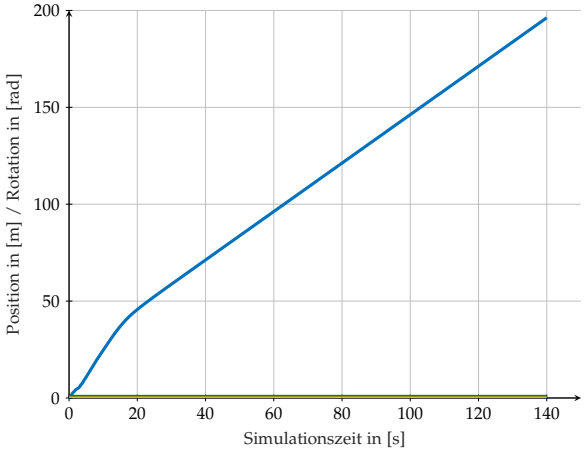


(a) Kosten der ersten Lösung des ACPBP für Experiment 2.5.4

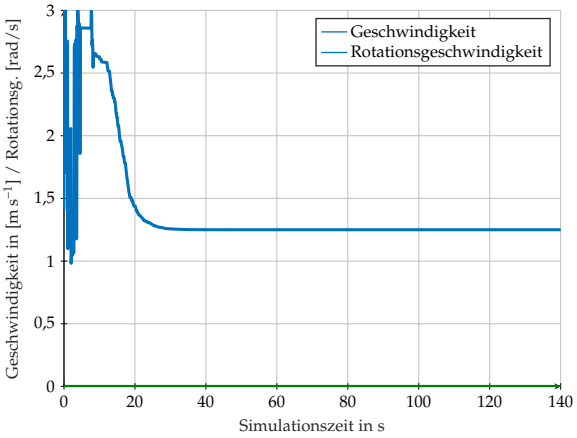


(b) Kosten der zweiten Lösung des ACPBP für Experiment 2.5.4

Abbildung 2.40.: Kosten in Abhängigkeit zur Distanz, der mit ACPBP ermittelten Trajektorienlösung.



(a) Zustandtrajektorie der mittels IPOPT ermittelten Lösung für die konstante Kostenfunktion



(b) Stellgrößentrajektorie der mittels IPOPT ermittelten Lösung für die konstante Kostenfunktion

Abbildung 2.41.: Regelung mittels IPOPT und unstetiger, stückweise konstanter Kostenfunktion.

0,000 24 s für die Lösung mit stabilem Endzustand und $t^{\text{ges}} = 11{,}955\,72\,\text{s}$, $\mu_t = 0{,}002\,99\,\text{s}$ und $\sigma_t = 0{,}000\,30\,\text{s}$ für die oszillierende Lösung.
Abbildung 2.41 veranschaulicht die mittels IPOPT berechneten Zustands- und Stellgrößentrajektorien. Dieser Optimierer ist aufgrund der fehlenden Referenztrajektorie und der Sprungstellen sowie der fehlenden Gradientenrichtungen bei konstanten Werten nicht in der Lage, das System weder in ein lokales noch in ein globales Minimum zu regeln.

2.5.2.6. Regelung für Landfahrzeugsysteme

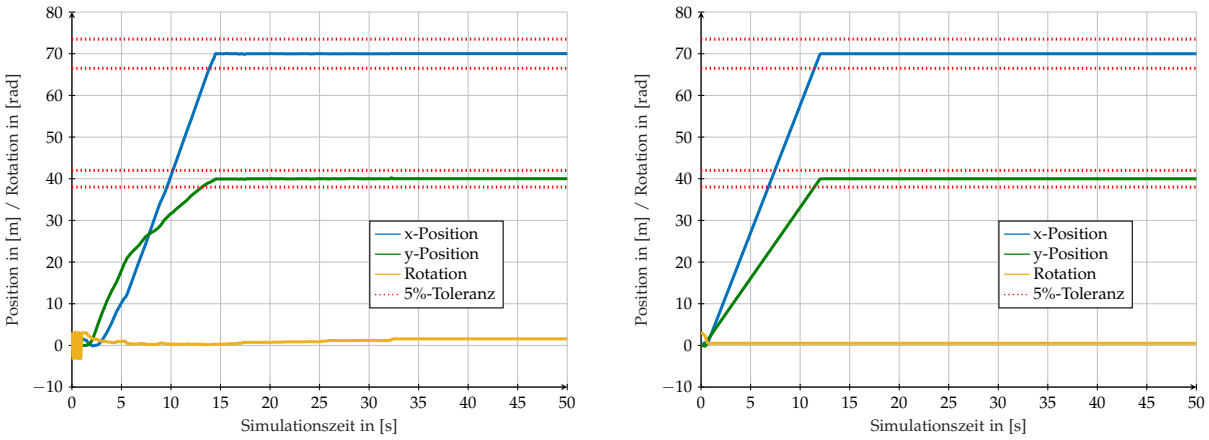
Zunächst werden die Experimente für die Landfahrzeugsysteme angegeben.

Experiment 2.5.5: *Im ersten Experiment soll ein System vom Startzustand* $\mathbf{x}_0 = [0\,\text{m} \quad 0\,\text{m} \quad \pi\text{rad}]^T$ *in den Zielzustand* $\mathbf{x}^{ref} = [70\,\text{m} \quad 40\,\text{m} \quad 0\,\text{rad}]^T$ *geregelt werden.*

Für die quadratische Kostenfunktion werden die Gewichtungsmatrizen erneut mit $\mathbf{Q} = \text{diag}([20 \;\; 20 \;\; 0])$ und $\mathbf{R} = \text{diag}([0 \;\; 0])$ definiert. Der Achsabstand des Systems beträgt $r_{\mapsto \odot_a} = 1{,}5\,\text{m}$. Die Stellgrößenbeschränkungen werden auf $\mathbf{u}_- = [-1{,}75\,\text{m}\,\text{s}^{-1} \;\; -1{,}0472\,\text{rad}]^T$ und $\mathbf{u}_+ = [7\,\text{m}\,\text{s}^{-1} \;\; 1{,}0472\,\text{rad}]^T$ festgesetzt. Dabei entspricht die maximale Geschwindigkeit von $\overset{\dagger,\dashv}{v}_+ = 7\,\text{m}\,\text{s}^{-1}$ der eines motorisierten Fahrrads. Das Regelungsziel des zweiten Experiments ist bis auf die Zielorientierung identisch zum Ersten.

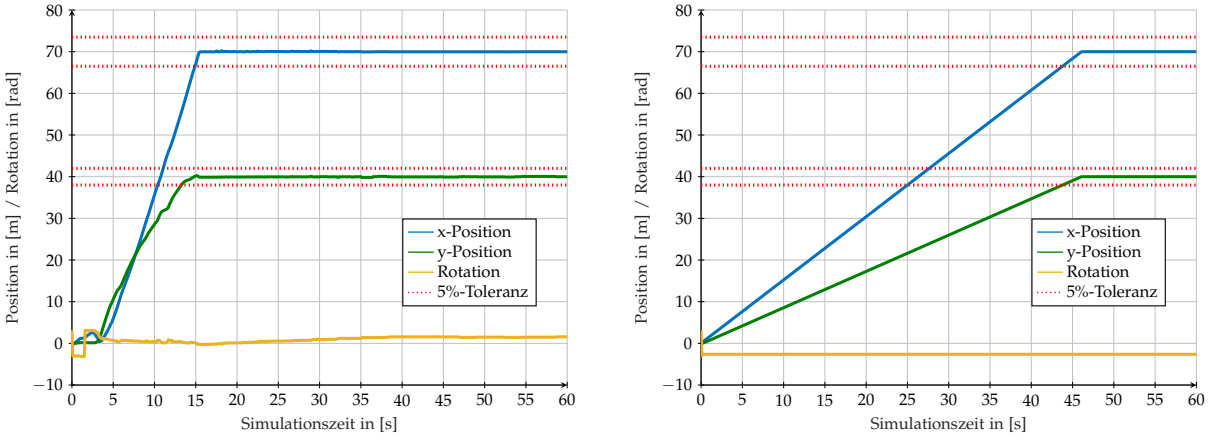
Experiment 2.5.6: *Es gilt* $\mathbf{x}^{ref} = [70\,\text{m} \;\; 40\,\text{m} \;\; \pi\text{rad}]^T$. *Das System soll sich entsprechend im Zielzustand entgegen der* $\overset{\dagger}{x}$*-Richtung positionieren.*

Die Parameter des ACPBP-Optimierers werden ebenfalls an die Stellgrößenbeschränkungen angepasst. Für den Mittelwert der Stellgrößen gilt $\boldsymbol{\mu} = [2{,}625\,\text{m}\,\text{s}^{-1} \;\; 0\,\text{rad}]^T$



(a) ACPBP-Lösung (b) IPOPT-Lösung

Abbildung 2.42.: Zustandstrajektorien des Experiments 2.5.5 für Landfahrzeuge.



(a) ACPBP-Lösung (b) IPOPT-Lösung

Abbildung 2.43.: Zustandstrajektorien der zweiten Regelungsaufgabe für Landfahrzeuge.

und für die Standardabweichung $\sigma_0 = [2\,\mathrm{m\,s^{-1}}\ \ 0{,}942\,48\,\mathrm{rad}]^T$. Die weiteren Werte werden mit Hilfe der Schätzregeln ermittelt. So sei $\sigma_1 = [4\,\mathrm{m\,s^{-2}}\ \ 1{,}884\,956\,\mathrm{rad\,s^{-1}}]^T$, $\sigma_2 = [40\,\mathrm{m\,s^{-3}}\ \ 18{,}8496\,\mathrm{rad\,s^{-2}}]^T$ und $\sigma_x = [1\,\mathrm{m}\ \ 1\,\mathrm{m}\ \ 0{,}471\,24\,\mathrm{rad}]^T$.

Ergebnisse der Regelungsexperimente für Landfahrzeuge Da der LM es nicht schafft die Systeme der Experimente 2.5.5 und 2.5.6 auszuregeln, werden nur noch IPOPT und ACPBP als Optimierer untersucht. Abbildung 2.42 zeigt die mit IPOPT und ACPBP berechneten Zustandstrajektorien für das erste Experiment dieser Systemklasse. Im Vergleich zur Systemklasse der Kettenfahrzeuge gibt es nur geringfügige Unterschiede zwischen den Ergebnissen der Optimierer. Die Ursache können die größeren Stellgrößenbeschränkungen sein. Aufgrund des großen Lenkwinkels $\overset{\dagger}{\iota}$ und der höheren Geschwindigkeit $\overset{\dagger,\dashv}{v}_+$ kann das System schneller seinen Zustand verändern. Beide Algorithmen liegen im Toleranzband und haben das System in den Zielzustand in weniger als 15 s ausgeregelt.
Die Ergebnisse für Experiment 2.5.6 sind in Abbildung 2.43 dargelegt. Die Lösung

Tabelle 2.4.: Rechenzeiten für das Autorobotersystem.

Optimierer	Experiment	Zeit	t^{ges}	μ_t	σ_t
ACPBP	2.5.5	50 s	1,408 951 s	0,002 812 s	0,000 601 7 s
ACPBP	2.5.6	60 s	1,679 385 s	0,002 794 s	0,000 522 0 s
IPOPT	2.5.5	50 s	5,627 550 s	0,011 233 s	0,004 094 s
IPOPT	2.5.6	60 s	10,4980 s	0,017 470 s	0,017 230 s

des IPOPT-Optimierers zeichnet sich dadurch aus, dass das System mit der minimalen Stellgröße $\overset{\dagger,\dashv}{v}_{-}$ und einer rückwärtigen Orientierung den Zielzustand ausregelt. Daraus resultiert eine höhere Ausregelzeit als mittels ACPBP. Dort wird das System innerhalb von etwa 15 s vollständig ausgeregelt. Dem gegenüber benötigt die Regelung mittels IPOPT 45 s bis zum Zielzustand. Die Erweiterung des Horizonts ändert das Verhalten des Optimierers nicht; ebenso wenig wie die Erhöhung der Anzahl der Iterationen. ACPBP ermittelt bei diesem Experiment keine grundlegend verschiedenen Lösungen. Die Berechnungszeiten mit Mittelwert und Standardabweichung werden für beide Experimente und Optimierer in Tabelle 2.4 aufgeführt.

2.5.2.7. Regelung von Wasserfahrzeugen

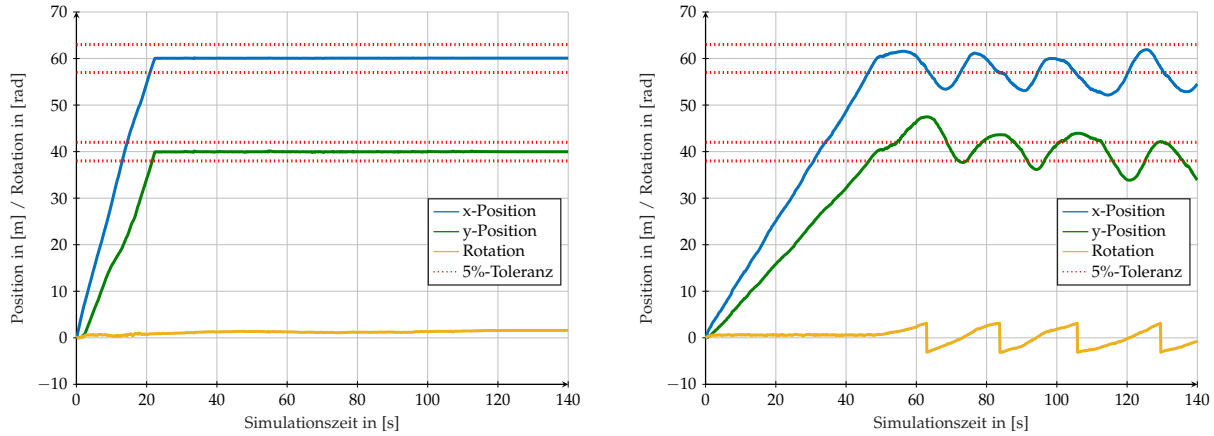
Systeme der Wasserfahrzeugkategorie werden anhand eines Experiments analysiert.

Experiment 2.5.7: *Das Ziel des Experiments ist es ein Wasserfahrzeugsystem vom Zustand* $\mathbf{x}_0 = [0\,\text{m}\ \ 0\,\text{m}\ \ 0\,\text{rad}]^T$ *in den Zielzustand* $\mathbf{x}^{ref} = [60\,\text{m}\ \ 40\,\text{m}\ \ 0\,\text{rad}]^T$ *zu regeln.*

Für die quadratische Kostenfunktion werden die Gewichtungsmatrizen mit $\mathbf{Q} = \text{diag}([20\ \ 20\ \ 0])$ und $\mathbf{R} = \text{diag}([0\ \ 0])$ definiert. Der Abstand zwischen Massenmittelpunkt und Ruder beträgt $r_{\mapsto\odot_m} = 2\,\text{m}$. Die Stellgrößenbeschränkungen werden mit $\mathbf{u}_- = [-0{,}875\,\text{m}\,\text{s}^{-1}\ \ -1{,}0472\,\text{rad}]^T$ und $\mathbf{u}_+ = [3{,}5\ \ 1{,}0472\,\text{rad}]^T$ angegeben. Die maximale Geschwindigkeit von $\overset{\dagger,\dashv}{v}_{+} = 3{,}5\,\text{m}\,\text{s}^{-1}$ entspricht in etwa der maximalen in Deutschland zulässigen Geschwindigkeit von Wasserfahrzeugen auf innerdeutschen Kanälen.

Parametereinstellungen für ACPBP bei Wasserfahrzeugsystemen Der Mittelwert der Stellgrößen sei $\boldsymbol{\mu} = [1{,}3125\,\text{m}\,\text{s}^{-1}\ \ 0\,\text{rad}]^T$ und die Standardabweichung sei $\boldsymbol{\sigma}_0 = [1{,}5\,\text{m}\,\text{s}^{-1}\ \ 0{,}942\,48\,\text{rad}]^T$. Die weiteren Werte werden mittels der Schätzregel bestimmt (siehe 2.5.2.3). Es sei $\boldsymbol{\sigma}_1 = [3\,\text{m}\,\text{s}^{-2}\ \ 1{,}884\,956\,\text{rad}\,\text{s}^{-1}]^T$, $\boldsymbol{\sigma}_2 = [30\,\text{m}\,\text{s}^{-3}\ \ 18{,}8496\,\text{rad}\,\text{s}^{-2}]^T$ und $\boldsymbol{\sigma}_x = [0{,}75\,\text{m}\,\text{s}^{-1}\ \ 0{,}75\,\text{m}\,\text{s}^{-1}\ \ 0{,}471\,24\,\text{rad}]^T$.

Ergebnisse der Regelung für das Bootsystem Die mittels ACPBP bestimmte Zustandstrajektorie, veranschaulicht in Abbildung 2.44a, regelt ds Zielsystem innerhalb von etwa 25 s in den Zielzustand und kann das System dort stabilisieren. Im Vergleich



(a) ACPBP-Lösung

(b) IPOPT-Lösung

Abbildung 2.44.: Regelung des Bootsystems.

Tabelle 2.5.: Rechenzeiten für das Bootsystem.

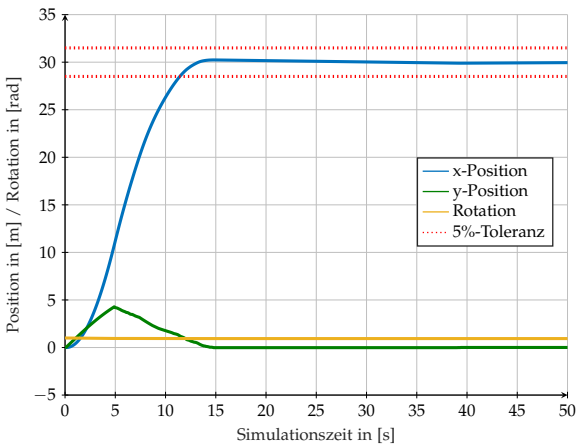
Optimierer	Zeit	t_{ges}	μ_t	σ_t
ACPBP	140 s	3,915 90 s	0,002 795 1 s	0,000 449 8 s
IPOPT	140 s	64,040 32 s	0,045 710 0 s	0,004 880 0 s

dazu erfordert die mittels IPOPT generierte Lösung eine signifikant höhere Anregelzeit als die Ausregelzeit mittels ACPBP. Weiterhin schafft es der Optimierer IPOPT nicht, eine stabilisierende Stellgrößenfolge zu bestimmen, sodass das System beginnt instabil aufzuschwingen (siehe Abbildung 2.44b). Aufgrund der komplexen Nichtlinearitäten des Systems ist es IPOPT nicht möglich, das Systemverhalten im aktuellen Arbeitspunkt adäquat zu linearisieren. Dies führt zu einer zu stark abweichenden Systemzustandsprädiktion, die eine instabile Stellgrößenfolge induziert. Aufgrund des Aufschwingens und der vergleichsweise sehr hohen Berechnungsdauer eignet sich IPOPT nicht für die Regelung dieser Systemklasse. Die Laufzeiten für die Regler des Experiments 2.5.7 sind in Tabelle 2.5 gelistet.

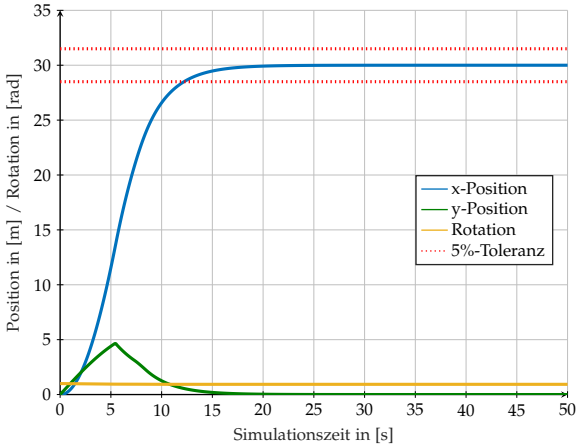
2.5.2.8. Regelung einer Rakete im Weltall

Um ACPBP als allgemeinen Optimierer für die modellprädiktive Regelung zu verwenden, wird dieser ebenfalls für System evaluiert, für die dieser Optimierer nicht ausgelegt ist. Somit wird seine Generalität überprüft. Als System dient das Modell einer Rakete im Weltall (siehe Abschnitt 2.5.2.1), welches wie folgt parametrisiert wird: Seien $\kappa_{R_1} = 0{,}02$ und $\kappa_{R_2} = 0{,}01$, sodass sie mit anderen Veröffentlichungen identisch gewählt sind (vgl. [155, 156]).
Es wird das gleiche Vergleichsexperiment wie in den Veröffentlichungen gewählt, um einen Vergleichsmaßstab zu setzen.

Experiment 2.5.8: *Das System soll vom Startzustand* $\mathbf{x}_0 = [0\,\mathrm{m}\ \ 0\,\mathrm{m\,s^{-1}}\ \ 1\,\mathrm{kg}]^T$ *in den Zielzustand* $\mathbf{x}^{ref} = [30\,\mathrm{m}\ \ 0\,\mathrm{m}\ \ \cdot]^T$ *geregelt werden.*



(a) ACPBP-Lösung für das Experiment 2.5.8



(b) IPOPT-Lösung für das Experiment 2.5.8

Abbildung 2.45.: Regelung eines Raketensystems im Weltall.

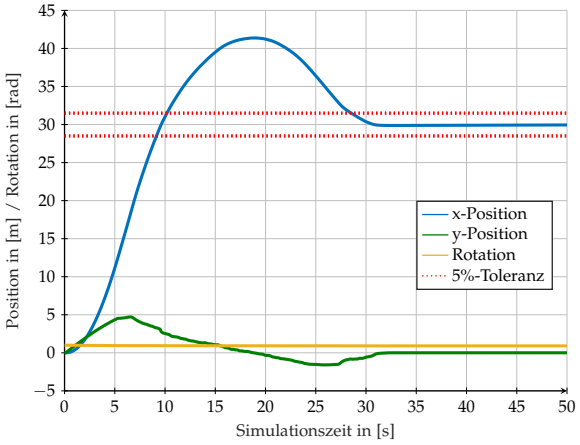
Demnach soll das Raketensystem eine Strecke von 30 m zurücklegen. Die finale Masse der Rakete ist dabei unbekannt, abhängig von der Regelung und soll nicht beschränkt werden. Die Masse definiert den vorhandenen Treibstoff der Rakete. Die quadratische Kostenfunktion wird mit $\mathbf{Q} = \text{diag}([20 \;\; 100 \;\; 0])$ und $\mathbf{R} = \text{diag}([0])$ definiert. Zudem gilt die Stellgrößenbeschränkung $|u| \leq 1$.

ACPBP-Konfiguration für die Rakete im Weltall Die Systemdynamik wird zwar mit drei Zuständen, allerdings nur mit einer Stellgröße beschrieben. Die passende Implementierung des ACPBP wird in der Reglertestumgebung eingesetzt. Jedoch arbeitet der simulierte Gradientenabstieg (siehe Abschnitt 2.4.3.3) nicht mehr vollständig korrekt, wodurch die Lösungsgüte reduziert werden kann. Für den Mittelwert der Stellgrößen gilt $\mu = 0$ und für die Standardabweichung wird $\sigma_0 = 0{,}2$ gewählt. Die restlichen Parameter werden mittels der Schätzregeln berechnet. Sei $\sigma_1 = 0{,}4$, $\sigma_2 = 4$ und $\boldsymbol{\sigma}_x = [0{,}1\,\text{m} \;\; 0{,}1\,\text{m}\,\text{s}^{-1} \;\; 0\,\text{kg}]^T$.

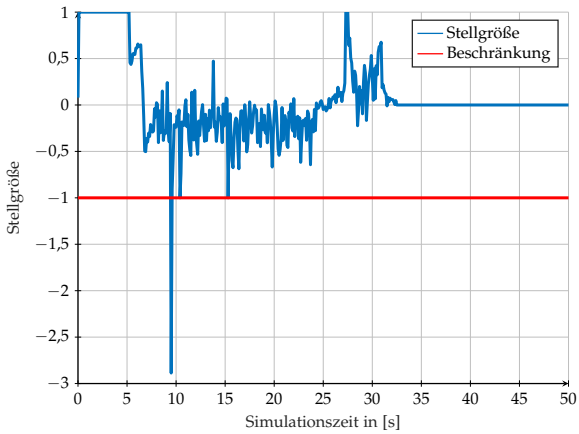
Ergebnisse der Regelung für das Raketensystem Die Ergebnisse der Regelung mittels ACPBP und IPOT sind in Abbildung 2.45 veranschaulicht. IPOPT verwendet eine Horizontlänge von $t_K - t_0 = 3\,\text{s}$ zur Berechnung der in Abbildung 2.45b dargestellten Lösung. Die Steigerung der Horizontlänge ermöglicht eine kürzere Ausregelzeit, die jedoch mit einer erhöhten Berechnungszeit einhergeht. Die erzeugte Zustandstrajektorie ist nahezu identisch zur mittels ACPBP erzeugten Lösung, die in Abbildung 2.45a abgebildet ist. Die Ausregelzeiten sind zwar nahezu identisch, aber bis zum Erreichen des Referenzwertes innerhalb des 5 %-Toleranzbandes vergehen bei der Lösung mittels IPOPT etwa 5 s mehr. Dennoch wird die Ausregelzeit als Gütemaß verwendet. Die Laufzeiten sind in Tabelle 2.6 aufgeführt und zeigen eine deutliche Laufzeitdiskrepanz zwischen den Optimierern. Es gilt, dass ACPBP für dieses Experiment etwa 10-fach schneller optimiert als IPOPT.
Mittels Variation der Gewichtungsmatrizen und Reglerparametern wird festgestellt, dass ACPBP die Stellgrößenbeschränkung nicht zu jedem Zeitpunkt einhält. Dies

Tabelle 2.6.: Rechenzeiten für die Rakete im Weltraum.

Optimierer	Zeit	t_{ges}	μ_t	σ_t
ACPBP	50 s	0,621 64 s	0,001 240 s	0,000 110 s
IPOPT	50 s	6,544 50 s	0,013 063 s	0,010 101 s



(a) Zustandstrajektorie mit Überschwingen der $\overset{\dagger}{x}$-Koordinate



(b) Stellgrößenverletzung für eine unpassende Gewichtung

Abbildung 2.46.: Stellgrößenüberschreitung mittels ACPBP bei unpassender Gewichtung.

tritt auf, falls die Gewichtung der Geschwindigkeit nicht deutlich höher ist als die Gewichtung der Strecke. Zudem resultiert aus der unpassenden Gewichtung ein Überschwingen des Systems. Dieses Verhalten wird in Abbildung 2.46 für die Gewichtungsmatrix $\mathbf{Q} = \text{diag}([20\ \ 20\ \ 0])$ und $R = 0$ dargestellt. IPOPT garantiert auch bei dieser Gewichtung die Einhaltung der Beschränkungen und weist kein Überschwingen auf.

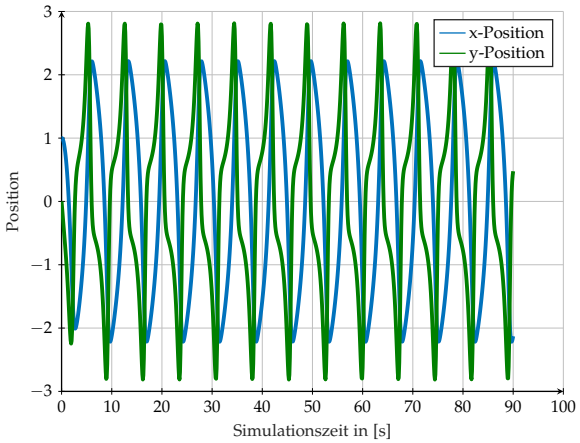
2.5.2.9. Regelung des Van-der-Pol-Oszillators

Die Güte der Optimierer wird für ein weiteres Standardvergleichssystem ausgewertet.

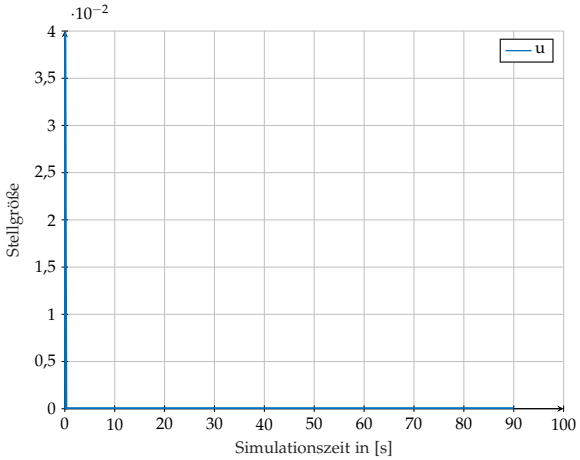
Experiment 2.5.9: *Das Regelungsziel des Van-der-Pol-Oszillators besteht darin vom Startzustand* $\mathbf{x}_0 = [1\ \ 0]$ *in den Zielzustand* $\mathbf{x}^{ref} = [0\ \ 0]^T$ *zu regeln.*

Die Gewichtungsmatrizen der quadratischen Kostenfunktion werden aus der Veröffentlichung Rösmann u. a. [159] mit $\mathbf{Q} = \text{diag}([1{,}5\ \ 0{,}5])$ und $R = 0.1$ übernommen. Ebenso wird die Stellgrößenbeschränkung $|u| \leq 1$ identisch gewählt, um eine Vergleichbarkeit zu gewährleisten. Die Parameter des ACPBP werden bis auf die Zustandsstandardabweichung aus dem Experiment 2.5.8 beibehalten. Der Van-der-Pol-Oszillator besitzt einen zweidimensionalen Zustandsvektor; daher sei $\sigma_x = [0{,}1\ \ 0{,}1]^T$.

Ergebnisse der Regelung des Van-der-Pol-Oszillators Die mit ACPBP berechneten Zustandstrajektorien und die dazugehörige Stellgrößenfolge sind in Abbildung 2.47 dargestellt. Aus Abbildung 2.47 ist ersichtlich, dass der Regler mit ACPBP-Optimierung

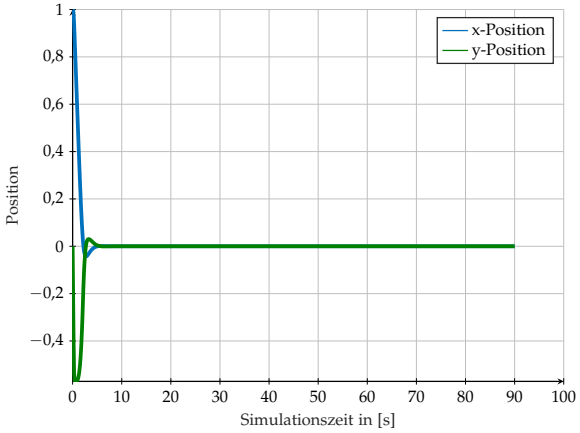


(a) Zustandstrajektorie der ACPBP-Lösung des Experimentes 2.5.9

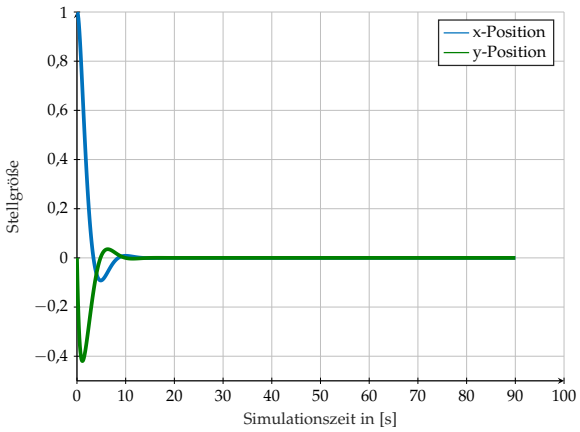


(b) Stellgröße der ACPBP-Lösung des Experimentes 2.5.9

Abbildung 2.47.: Regelung des Van-der-Pol-Oszillators mittels ACPBP.



(a) IPOPT-Lösung mit Horizontlänge $t_K - t_0 = 2{,}5\,\mathrm{s}$

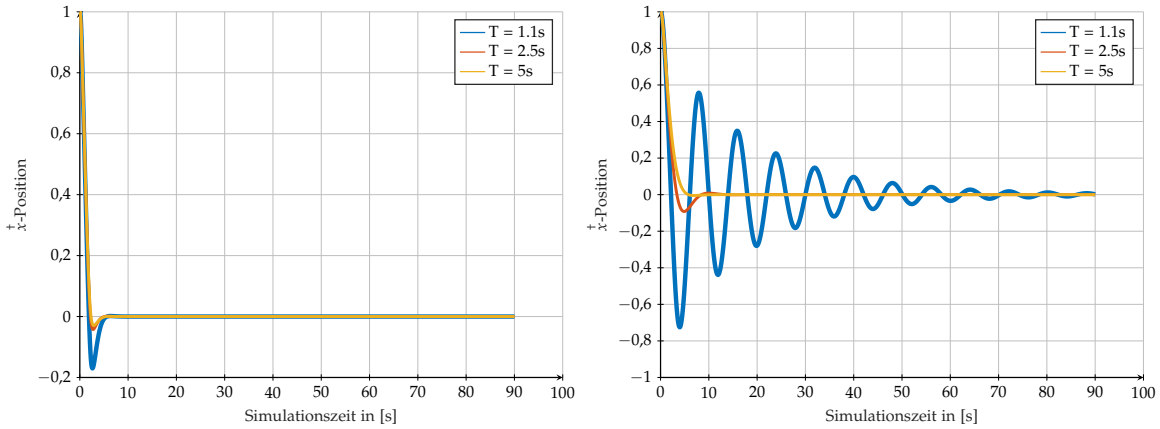


(b) LM-Lösung mit Horizontlänge $t_K - t_0 = 2{,}5\,\mathrm{s}$

Abbildung 2.48.: Regelung des Van-der-Pol-Oszillators mit gradientenbasierten Verfahren.

nicht in der Lage ist den Referenzzustand zu erreichen und das System dort zu stabilisieren. Stattdessen erzeugt der Optimierer eine Impulsfunktion, die das System in Schwingung mit Amplituden $\mathbf{x}_{[1]} = 2{,}2$ und $\mathbf{x}_{[2]} = 2{,}8$ versetzt. Durch Variation der Parameter kann ein stabilisierendes Verhalten erzeugt werden, jedoch verletzt dieses die Stellgrößenbeschränkungen. Somit ist ACPBP nicht für diese Systemklasse geeignet.

Durch Anpassung der Horizontlänge von $t_K - t_0 = 1{,}1\,\mathrm{s}$ auf $t_K - t_0 = 2{,}5\,\mathrm{s}$ für LM kann die Ausregelzeit deutlich reduziert werden (siehe Abbildung 2.49). Eine weitere Steigerung der Horizontlänge erhöht die Lösungsgüte nur minimal, steigert die Berechnungsdauer jedoch deutlich. Diese Abhängigkeit wird in der Laufzeitentabelle 2.7 hervorgehoben und belegt. Im Vergleich dazu ist ein Ausregeln sowohl mit IPOPT als auch dem LM-Algorithmus möglich. Abbildung 2.48 zeigt die berechneten Zustandsfolgen für den Van-der-Pol-Oszillator mittels IPOPT und LM-Algorithmus. Die Durchführung des Experiments zeigt, dass mit der Steigerung der Horizontlänge auf



(a) IPOPT mit variierenden Horizontlängen

(b) LM mit variierenden Horizontlängen

Abbildung 2.49.: Regelung des Van-der-Pol-Oszillators mit unterschiedlicher Horizontlänge.

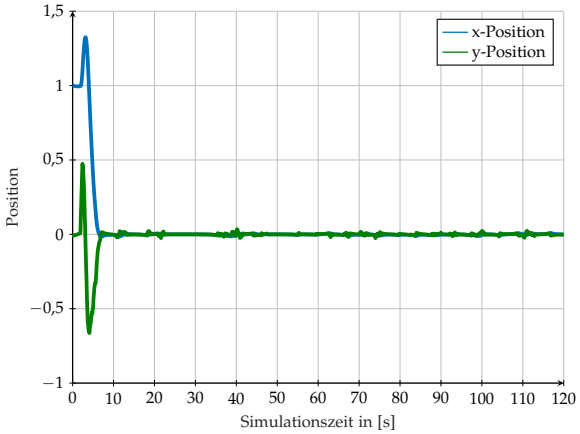
Tabelle 2.7.: Rechenzeiten der Regelung des Van-der-Pol-Oszillators.

Optimierer	Horizontlänge	Zeit	t_{ges}	μ_t	σ_t
IPOPT	1,1 s	90 s	1,568 49 s	0,001 74 s	0,000 42 s
IPOPT	2,5 s	90 s	2,505 90 s	0,002 78 s	0,001 09 s
IPOPT	5 s	90 s	4,382 62 s	0,004 86 s	0,001 98 s
LM	1,1 s	90 s	0,279 82 s	0,000 31 s	0,000 04 s
LM	2,5 s	90 s	1,370 43 s	0,001 52 s	0,000 13 s
LM	5 s	90 s	7,669 64 s	0,008 51 s	0,000 71 s

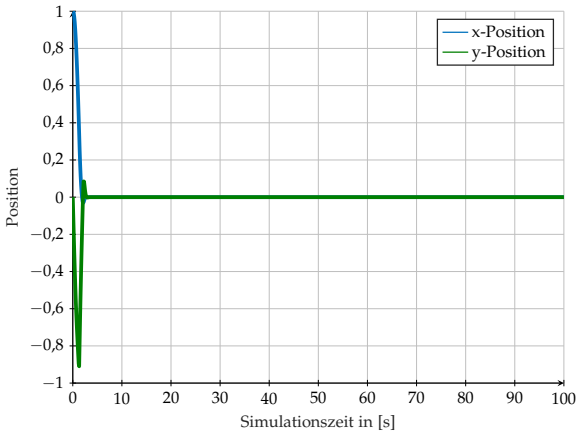
$t_K - t_0 = 2{,}5\,\text{s}$ das Überschwingen und die Ausregelzeit reduziert werden können. Zur Erläuterung wird die $\mathbf{x}_{[1]}$-Komponente für die Optimierer bei unterschiedlicher Horizontlänge in Abbildung 2.49 visualisiert. Das Systemverhalten der zweiten Komponenten ist analog und wird daher nicht näher betrachtet.

Die Lösungsgüte mittels IPOPT bleibt ab $t_K - t_0 = 2{,}5\,\text{s}$ konstant, wohingegen die berechnete Lösung des LMS in Abbildung 2.49a stetig verbessert wird. Ab $t_K - t_0 = 5\,\text{s}$ tritt kein Überschwingen des Systems mehr auf. Für kurze Horizontlängen schwingt das System stärker auf, konvergiert jedoch trotzdem in den Zielzustand. Die Laufzeiten des Reglers für die unterschiedlichen Horizontlängen sind in Tabelle 2.7 aufgeführt.

Die Berechnungsdauer des LMs steigt nichtlinear und ist für kurze Horizontlängen gering. Die Dauer der IPOPT-Lösung steigt nicht mit derselben Rate wie die des LMs. Sie weist bereits für kurze Horizontlängen eine deutlich längere Berechnungszeit auf; allerdings steigt diese für längere Horizonte nur gering an. Werden beide Algorithmen mit einem ähnlichen Zeitkontingent verglichen, so ist die Regelungsgüte für IPOPT bei $t_K - t_0 = 1{,}1\,\text{s}$ dennoch höher als die des LMs bei $t_K - t_0 = 2{,}5\,\text{s}$. Denn die Überschwingweite und die Ausregelzeit sind geringer. Für eine hohe Regelungsgüte mit unbegrenzter Berechnungsdauer sollte der LM mit einer großen Horizontlänge gewählt werden.



(a) Lösung mittels ACPBP für Experiment 2.5.10



(b) Lösung mit IPOPT für Experiment 2.5.10

Abbildung 2.50.: Regelung des Duffing-Oszillators.

2.5.2.10. Regelung des Duffing-Oszillators

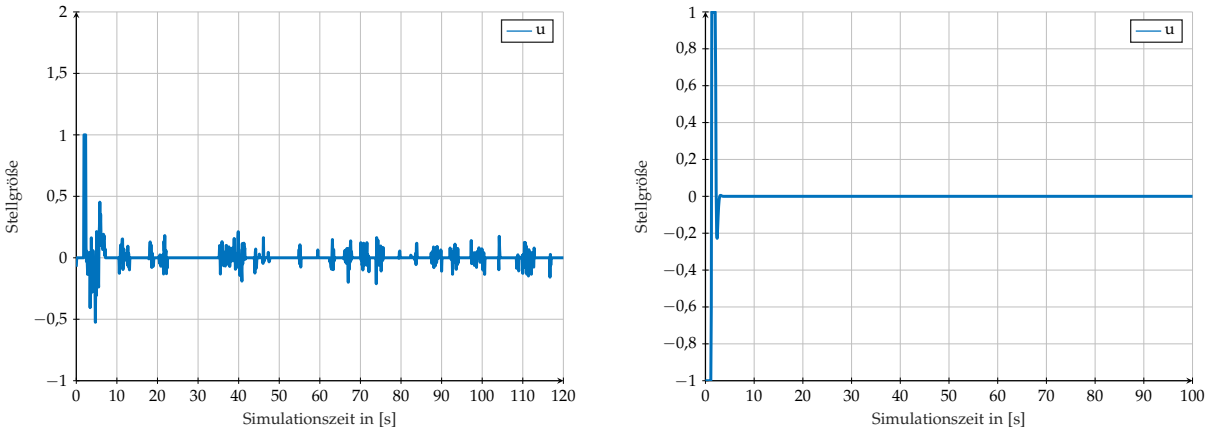
Das dritte Vergleichssystem ist der Duffing-Oszillator, welcher ebenfalls mittels eines Experiments evaluiert wird.

Experiment 2.5.10: *Ziel des Experimentes ist es, das System vom Startzustand* $\mathbf{x}_0 = [1 \;\; 0]$ *in den Referenzzustand* $\mathbf{x}^{ref} = [0 \;\; 0]^T$ *zu regeln.*

Für die quadratische Kostenfunktion werden die Gewichtungsmatrizen mit $\mathbf{Q} = \text{diag}([1000 \;\; 5])$ und $R = 0.1$ definiert. Die Stellgrößenbeschränkung ist $|u| \leq 1$ und es seien $\kappa_{D_1} = 0{,}125$, $\kappa_{D_2} = -1$ und $\kappa_{D_3} = 1$. Die Parameter für ACPBP werden identisch zur Regelung des Van-der-Pol-Oszillators gewählt.

Ergebnisse der Regelung des Duffing-Oszillators Die ausgeregelten Zustandstrajektorien des Reglers mittels ACPBP und IPOPT sind in Abbildung 2.50 dargestellt. Beide Optimierer regeln das System aus. Abbildung 2.50a zeigt ein kurzzeitiges Aufschwingen der Trajektorie, bis das System nach 10 s Simulationszeit im Referenzzustand stabilisiert wird. Aufgrund der stochastischen Lösungen mittels ACPBP werden Stellgrößen erzeugt, wodurch das System kurzzeitig den Referenzzustand verlässt, aber dann wieder ausgeregelt wird. Das 5 %-Toleranzband wird dabei nicht verletzt, da die maximale Abweichung 0,05 beträgt. Der Optimierer IPOPT regelt das System ohne aufschwingen und innerhalb von 4 s aus. Die Stellgrößentrajektorien beider Lösungen sind in Abbildung 2.51 abgebildet. Abbildung 2.51 zeigt dabei deutlich die stochastische Lösung des ACPBP. Bei diesem Experiment wird dazu bewusst nur ein Durchlauf mittels ACPBP durchgeführt, sodass nicht der Mittelwert abgebildet ist.

Die Laufzeiten der Regelung des Duffing-Oszillators mittels der beiden Optimierer sind in Tabelle 2.8 gelistet. IPOPT weist ausschließlich bei diesem Experiment mit durchschnittlich 0,001 96 s pro Zeitschritt eine geringere Laufzeit als ACPBP auf.



(a) Stellgröße die für ACPBP-Lösung

(b) Stellgröße die für IPOPT-Lösung

Abbildung 2.51.: Stellgrößen für die Regelung des Duffing-Oszillators.

Tabelle 2.8.: Berechnungsdauer für das System des Duffing-Oszillators.

Optimierer	Zeit	t_{ges}	μ_t	σ_t
ACPBP	120 s	2,576 09 s	0,002 14 s	0,000 20 s
IPOPT	120 s	1,963 99 s	0,001 96 s	0,000 76 s

2.6. Zwischenfazit und Bewertung der Anwendbarkeit

Dieses Kapitel beschäftigt sich grundlegend mit der Regelung einzelner Agentensysteme in der Ebene. Diese Systeme können über Kostenfunktionen zu einem dezentralen Schwarm mit emergentem Verhalten verknüpft werden ohne ein globales Optimalsteuerungsproblem (siehe Abschnitt 2.4.1) zu erzeugen. Zur Regelung der einzelnen Systeme werden sowohl lernende Ansätze (siehe Abschnitt 2.1) als auch planende Ansätze (siehe Abschnitt 2.2) hinsichtlich ihre Anwendbarkeit evaluiert. Dabei ist festzustellen, dass sich lernenden Ansätze nicht für die in Kapitel 1 geforderten Anforderungen eignen, weil sie zwar aus Zeitreihen der Vergangenheit lernen können; jedoch bieten sie keine Zukunftsprädiktionen. Ferner sollen weitere Missionen für den Schwarm als Programmerweiterung hinzugefügt werden können, ohne dass alle Agenten neu angelernt werden müssen. Weiterhin ist das größte Problem die Abbildung eines Roboterschwarms mit dynamischer Größe im Zustandsraum eines einzelnen Agenten, um emergentes Verhalten zu erlernen. Folglich fokussiert sich diese Arbeit auf planende Ansätze, die mit Hilfe eines Modells den Suchraum für optimale Trajektorien reduzieren und Zukunftsprädiktionen anhand des Modells ermöglichen. Dabei wird herausgestellt, dass die Verwendung eines Modells nicht nur zur Trajektorienvalidierung sondern auch zur Generierung verwendet werden sollte. Dazu werden verschiedene Systeme beschrieben (siehe Abschnitt 2.3), die einen Großteil der Oberflächenfahrzeuge abbilden. Basierend auf diesen Modellen werden verschiedene Optimierer (siehe Abschnitt 2.4) innerhalb der modellprädiktiven Regelung ausgewertet, die die Struktur des Modells zur Laufzeitreduktion mit verschiedenen Diskretisierungsverfahren explorieren. CPBP wird als Verfahren ausgewählt und um weitere

Funktionen und Lösungsansätze ergänzt, die dessen Lösungsgüte deutlich steigern (siehe Abschnitt 2.5). Dabei gilt, dass der entworfene ACPBP-Algorithmus sich unter Betrachtung verschiedener Aspekte für die Regelung der angestrebten Systeme eignet. Er erzielt die besten Lösungen unter Einhaltung der Systemzeitschranken und ist ebenfalls in der Lage stark nichtlineare Systeme und unstetige Kostenfunktionen brauchbar zu optimieren. Die Ergebnisse des Algorithmus sind nicht theoretisch bewiesen optimal, zeigen jedoch in der empirischen Auswertung eine hohe Güte. Die Güte des Reglers wird ebenfalls für regelungstechnische Vergleichsmaßstäbe untersucht. Damit wird die Generalität des Ansatzes dargelegt. Aufgrund des definierten Planungshorizontes K und des fehlenden globalen Konvergenzbeweises lässt sich ACPBP in die Klasse der lokalen Pfadplanungsalgorithmen einordnen. Allerdings kann durch die Missionsbeschreibung (siehe Kapitel 5) bzw. die Modellierung der Kostenfunktion eine globale Konvergenz unterstützt werden.

2.6.1. Unterschiede und Gemeinsamkeiten von planenden und lernenden Verfahren

Wie durch die Notation und die Querverweise betont, sind planende und lernende Verfahren nicht vollständig disjunkt. Gerade die dynamisch trainierten Verfahren, wie z. B. PPO, die Rückspielpuffer verwenden und Episoden auswerten, besitzen zwar keinen Planungshorizont, sondern einen Erinnerungshorizont. Allerdings wird über diesen ebenfalls ein Gütefunktional bestimmt und zur Anpassung und Auswahl der Stellgrößen verwendet. Im Vergleich zu modellprädiktiven Reglern existieren für lernende Verfahren Methoden, die nur mit gelegentlichen Belohnungen arbeiten können (Englisch: „Sparse Reward“). Dies bedeutet, dass die Belohnungsfunktion nicht auf dem Gesamtzustandsraum bzw. dessen zeitlichen Produkts definiert ist oder nur für bestimmte Werte nicht das neutrale Element der Kostenaggregatsfunktion annimmt. Beispielsweise werden nur Belohnungen bei erfolgreichem Abschluss einer Mission erzeugt. Diese Belohnung wird dann rückwirkend auf die Aktionssequenz aufgeteilt, um das Verhalten zu erlernen. MPC mit gradientenfreien Optimierern, die auch unstetige Funktionen optimieren können, bieten eine analoge Funktionsweise (siehe Abschnitt 2.5). Ebenfalls weist die interne Struktur der KNN Parallelen zu kaskadierten Reglern und die Neuronen Parallelen zu erweiterten PID-Reglern auf (siehe Abschnitt 2.1.7). Die Anwendbarkeit beider Verfahren beruht jedoch maßgeblich auf der Formulierung der Kosten- bzw. Belohnungsfunktion durch Expertenwissen (Englisch: „Reward Engineering“). Dieses dient anschließend dem Training der KNN und muss alle notwendigen Informationen und Zusammenhänge enthalten, um Rückschlüsse und logisches Handeln zu erlernen. Allerdings bieten KNN keine theoretische Validierung oder Nachvollziehbarkeit der Aktionswahl, sodass Verhaltens- und Sicherheitszertifizierungen erschwert werden. Dem gegenüber bieten MPC zum Teil vollständig beweisbare Optimalität und eine Nachvollziehbarkeit der Trajektorienauswahl zur Laufzeit (vgl. [155]). Dies geht allerdings mit deutlich längeren Laufzeiten einher. Um die Vorzüge beider Verfahren auszunutzen, sind Kombinationen zu untersuchen, bei denen das Modell durch lernende Verfahren mittels der Interaktion mit dem System und der Prädiktion anhand des Modells zur Laufzeit verbessert und an die Parameter

Tabelle 2.9.: Übersicht über die lernenden Verfahren.

Optimierer	Verfahren	Laufzeit	Optimalität	Vorteil	Nachteil
DQL [179]	RL	++	Heuristik	Modellfrei	Training, zeitlos
PPO [164]	RL	++	Heuristik	„On-Policy“	Keine Prädiktion
LSTM [49]	RL	++	Heuristik	Gedächtnis	Keine Prädiktion

angepasst wird. Diese erhöhte Modellgüte bei einer dennoch geringen Laufzeit resultiert in akkurateren Prädiktionen und somit in einer für das reale System optimierten Trajektorie (vgl. [7]).

2.6.2. Ausblick auf künftige Regelungsverfahren für autonome Agentenschwärme

Künftige Erweiterungen der vorgestellten Verfahren werden sich vor allem mit der Kombination der Verfahren und dem Ausgleich der Nachteile beschäftigen (vgl. [104]). Somit lassen sich lernende Ansätze vor allem im Bereich Adaption und Anpassung an die Realität zur Reduktion des Fehlers zwischen Prädiktion und Realität einsetzen. Ebenfalls lassen sich durch die Steigerung der Rechenleistung, der Reduktion der Funktionsauswertungszeiten und einer erhöhten Modellgüte mehr parallele, weitreichendere und detailliert Trajektorien bestimmen. Diese ermöglichen die Regelung größerer Agentenschwärme und die Lösung komplexerer kooperativer Missionen. Zur Bewertung und Analyse der bestehenden und neu entwickelten Ansätze können noch weitere vergleichende Experimente im Pfadplanungsumfeld durchgeführt werden. Dazu eignet sich neben der PAGMO-Umgebung (vgl. [15]) auch die Programmbibliothek „Open Motion Planning Library“ (kurz OMPL, vgl. [178]). Diese Bibliothek enthält verschiedene Planungsalgorithmen und Referenzvergleiche aus dem Bereich der Robotik und ist damit spezifischer als die allgemeinen regelungstechnischen Vergleiche (siehe Abschnitt 2.5). Jedoch liegt der Fokus auf der Bewegungsplanung und nicht so universell auf der Minimierung beliebiger Kostenfunktionen, die eine große Bandbreite möglicher Missionen abdeckt (siehe Abschnitt 5). Zur Zertifizierung des ACPBP sollte versucht werden die Konvergenz und theoretische Optimalität für verschiedene Szenarien auszuweiten und zu zeigen (siehe Anhang C.5). Als Abschluss werden die wichtigsten angesprochenen Algorithmen kurz mit ihren zentralen Eigenschaften tabellarisch dargestellt (siehe Tabelle 2.9, 2.10, 2.11 und 2.12):

Tabelle 2.10.: Übersicht über die globalen Pfadplaner.

Optimierer	Verfahren	Laufzeit	Optimalität	Vorteil	Nachteil
Dijkstra [5]	Graphen	+	Global	Modellfrei	Graphen
A* [102]	Graphen	-	Global	Gitterwelt	Gitterwelt
RRT [96]	Graphen	- -	Global	Kontinuierlich	keine Prädiktion
RRT* [71]	Graphen	- -	Global	Pfadoptimierung	Komplex
RRT-A* [99]	Graphen	-	Lokal	Schnell	Lokale Minima
RRT+ [196]	Graphen	-	Lokal	Systembeachtung	Lokale Minima
informed RRT [71]	Graphen	-	Global	Schnell	keine Realzeit
RT-RRT* [120]	Graphen	-	Global	Schnell	Keine Systembeachtung
PRM [86]	Graphen	- -	Global	Multiple Pfade	Langsam

Tabelle 2.11.: Übersicht über die lokalen Pfadplaner.

Optimierer	Verfahren	Laufzeit	Optimalität	Vorteil	Nachteil
DWA [41]	Stochastisch	++	Lokal	Systembeachtung	Kurze Prädiktion
TEB [158]	Gradienten-basiert	+	Lokal	Zeitoptimal	Keine Lösungsgarantie
TEB-MPC [160]	Gradienten-basiert	+	Lokal	Harte Beschränkungen	Keine Lösungsgarantie
CPBP [57]	Stochastisch	+	Lokal	Hohe Lösungsdimension	Rauschen
ACPBP	Stochastisch	+	Lokal	Sehr effizient	Rauschen

Tabelle 2.12.: Übersicht über die beschriebenen Optimierungsverfahren.

Optimierer	Verfahren	Laufzeit	Optimalität	Vorteil	Nachteil
CS [63]	Direkt	-	Global	Konvergenzbeweis	Keine Prädiktion
Cobyla [138]	Direkt	0	Lokal	Gradientenfrei	Lineare Approximation
PSO [73]	Stochastisch	0	Global	Konvergenzbeweis	Rauschen
DE1220 [15]	Evolutionär	0	Global	Modellfrei	Langsam
Hypergraph [159, 155]	Gradienten-basiert	+	Lokal	Harte Beschränkungen	Gradienten-basiert

3

Drahtlose Knotenpunktnetzwerktechnik

Das Ziel dieser Arbeit ist die Entwicklung eines intelligenten, dezentralen und emergenten Schwarmverhaltens für autonome Roboterschwärme. Dazu müssen die mobilen Einheiten untereinander sowie ggf. mit Kommandanten, die Missionen erteilen, kommunizieren. Aufgrund der Mobilität und der dynamischen Topologie des Schwarmes können keine statischen Netzwerke verwendet werden. Diese Einschränkung bezieht sich sowohl auf die physikalische Bitübertragungsschicht als auch auf die logische Netzwerkschicht und betrifft folglich die transportorientierten Schichten des ISO/OSI-Referenzmodells (Englisch: „Open Systems Interconnection Model“, kurz OSI-Modell) 1 bis 4 (vgl. [42]). Die unbekannten Gegebenheiten eines in Kapitel 1 definierten Katastrophenfalls als Einsatzszenario garantieren keine Sichtverbindung zwischen und zu den Agenten und weisen ggf. eine erhöhte Geräuschkulisse auf. Daher entfallen licht- oder laserbasierte sowie akustische Medien der Signalübertragung, sodass sich diese Arbeit ausschließlich mit Knotenpunktnetzwerken auf Grundlage von Radiowellen befasst.

Definition 3.0.1: *Knotenpunktnetzwerke (Englisch: „Mesh Networks“) unterscheiden sich grundlegend von hierarchischen oder zentralen Netzwerkarchitekturen, da sie aus homogenen Knoten bestehen, die die gleichen funktionalen Eigenschaften besitzen und keiner Abhängigkeit zueinander unterliegen (vgl. [42]).*

Allgemeine Grundlagen der Netzwerk- und Übertragungstechnik sind in dem Buch Freyer [42] erklärt.

3.1. Paketbasierte Übertragung und Routing

Die Datenübertragung im Netzwerk erfolgt paketbasiert, denn die dynamische Topologie erschwert verbindungsorientierte Übertragungen. Letztere eigenen sich vor allem bei statischen Leitungsnetzen, jedoch nicht bei der Verwendung eines geteilten Mediums wie dem freien Raum. Dort interferieren die Wellen und erschweren die Dekodierung der Signale. Weiterhin begünstigt die Verwendung zustandsloser Pakete die Informationsvermittlung, weil diese beliebige Pfade durch das dynamische Netzwerk nehmen können und dabei dupliziert sowie verworfen werden können. Die Verschiebung der Komplexität von der Verbindung hinzu den Paketen wirkt sich aller-

dings nachteilig auf die Paketgröße aus, da diese alle nötigen Informationen enthalten müssen, um eine logische Ordnung und den Zustand zu rekonstruieren.
Routingverfahren für mobile ad hoc Netzwerke (kurz MANET) berechnen Pfade und Tabellen, um Pakete zielgerichtet von einem Knoten zu einem oder mehreren anderen Knoten zu transferieren (Englisch: „Singlecast“ oder „Multicast“). In dem Papier G. Kaur und P. Thakur [44] wird ein Überblick über die verschiedenen Verfahren gegeben. Allerdings unterscheidet sich die Aufgabe der Steuerungsinformationsverteilung innerhalb eines Schwarms von dem Datentransfer für universelle Daten aus dem Internet an Endnutzer, da bidirektional allen Teilnehmern die jeweils eigenen Informationen mitgeteilt werden müssen. Dieser Rundfunkansatz (Englisch: „Broadcast“) muss zur Reduktion der Netzwerklast beachtet werden. Für die Regelung autonomer Roboterschwärme werden zwei Nachrichtentypen definiert:

Definition 3.1.1 (Allgemeine Rundfunknachricht)**:** *Die allgemeine Rundfunknachricht ist für alle Knoten des Netzwerks bestimmt und wird bei Empfang von jedem Knoten einmalig wieder auf das Medium gegeben, um auf Basis des Echo-Algorithmus das gesamte Netzwerk zu fluten (vgl. [186]).*

Dieser Nachrichtentyp ist vor allem für Missionsinformationen und den globalen Wissenstransfer bestimmt, bei denen die Roboter denselben Kenntnisstand besitzen müssen. Weiterhin können so ebenfalls Informationen über unbekannte statische Hindernisse, die mit geringer Frequenz auftreten und zum Aufbau einer Karte nötig sind, übertragen werden.

Definition 3.1.2 (Nachbarschaftsrundfunknachricht)**:** *Die Nachbarschaftsrundfunknachricht ist für alle benachbarten Knoten eines Knotens bestimmt und wird weder wiederholt noch weitergeleitet, weil ausschließlich lokale, temporäre Informationen transferiert werden.*

Dieser Nachrichtentyp eignet sich vor allem für hochfrequente Daten, die nur eine temporäre und lokale Gültigkeit haben. Als Beispiel für diesen Typ sind zyklische Trajektorieninformationen und dynamische Hindernisse zu nennen. Diese ändern sich stets und beeinflussen ausschließlich die unmittelbare Umgebung. Universelles Knoten-zu-Knoten-Routing kann über eine Verkettung der Nachbarschaftsrundfunknachrichten repliziert werden. Durch diese Vereinfachung reduziert sich das Routing auf einen Paketfilter basierend auf einer eindeutigen Nachrichtenkennung, die Identifizierungsnummer (Englisch: „Identification Digits“, kurz ID), und einer logischen Paketordnung.

3.2. Synchronisation und logische Ordnung

Der Nachteil der zustandslosen Nachrichtenvermittlung besteht darin, dass nicht bekannt ist, ob die Nachricht bereits empfangen worden ist und ob ihr Inhalt zeitlich vor oder nach den bereits empfangenen Paketen entstanden ist. Dies ist für die Kausalitätskette der Informationen entscheidend. Somit ist es beispielsweise wichtig, ob die Rücknahme einer Mission für die aktuelle Mission oder eine vorherige desselben Typs

gilt.

Beispiel 3.2.1: *Ein Sender sendet die Befehlsfolge „fahre vorwärts“, „stop“ und „90° Drehung“ an ein autonomes Auto. Dieses empfängt z. B. die Befehlsfolge „90° Drehung“, „fahre vorwärts“ und „stop“.*

Um eine logische Ordnung auf den Nachrichten zu induzieren und somit implizit die Knoten logisch zu synchronisieren, wird das erweiterte Konzept der Lamport-Uhren verwendet (vgl. [92, 141]). Diese logischen Uhren bilden einen mathematischen Verband mit Null als Minimum. Dazu sei der Netzwerkgraph $G^{\hookrightarrow} = (V^{\hookrightarrow}, E^{\hookrightarrow})$ über seine Knoten $v^{\hookrightarrow} \in V^{\hookrightarrow}$ und seine Kanten $E^{\hookrightarrow} \ni e^{\hookrightarrow} = (v_1^{\hookrightarrow}, v_2^{\hookrightarrow})$ mit $v_1^{\hookrightarrow}, v_2^{\hookrightarrow} \in V^{\hookrightarrow}$ definiert. Sei eine Nachricht dann eine geordnete Menge $\mathbf{m}^{\hookrightarrow} =< i^{\hookrightarrow}, u^{\hookrightarrow}, t^{\hookrightarrow}, s^{\hookrightarrow}, \mathbf{d}^{\hookrightarrow} >$. Hier definiert $i^{\hookrightarrow}$ die Identifikationsnummer (Englisch: „Identification Digit“, kurz ID), $u^{\hookrightarrow}$ den Zeitstempel einer Lamport-Uhr $\mathfrak{u}(v^{\hookrightarrow})$ eines Knotens $v^{\hookrightarrow} \in V^{\hookrightarrow}$, $t^{\hookrightarrow}$ den Typen der Nachricht, $s^{\hookrightarrow}$ die Größe der Nachricht und $\mathbf{d}^{\hookrightarrow}$ die Nutzdaten der Nachricht. Zu Beginn des Netzwerks gilt für jeden Knoten:

$$\forall v^{\hookrightarrow} \in V^{\hookrightarrow}.\ \mathfrak{u}(v^{\hookrightarrow}) = 0 \tag{3.2.1}$$

Anschließend wird für jede eigene erzeugte Nachricht $\mathbf{m}^{\hookrightarrow}$ eines Knoten $v^{\hookrightarrow}$ ein Zeitstempel mittels Algorithmus 3.1 erzeugt und die Nachricht anschließend versendet. Für Nachrichten, die nur empfangen, verarbeitet und ggf. weitergeleitet werden, gilt, dass ausschließlich die Aktualisierungsfunktion aufgerufen wird. Auch wenn eine Nachricht $\mathbf{m}^{\hookrightarrow}_{\text{weiter}} =< i^{\hookrightarrow}, u^{\hookrightarrow}_{\text{weiter}}, t^{\hookrightarrow}, s^{\hookrightarrow}, \mathbf{d}^{\hookrightarrow}_{\text{weiter}} >$ von einem Knoten $v^{\hookrightarrow}_{+}$ mit einer höheren Uhr $\mathfrak{u}(v^{\hookrightarrow}_{+}) > u_{\text{weiter}}$ weitergeleitet wird, so gilt, dass der Inhalt der Nachricht $\mathbf{d}^{\hookrightarrow}_{\text{weiter}}$ nur auf Informationen mit kleinerem Zeitstempel $u^{\hookrightarrow}_{-} < u^{\hookrightarrow}_{\text{weiter}}$ basiert, weil die Daten $\mathbf{d}^{\hookrightarrow}_{\text{weiter}}$ durch das Weiterleiten nicht modifiziert werden. Somit bleibt die Kausalitätskette konsistent (vgl. [92, 141]).

Algorithmus 3.1.: Funktionen einer Lamport-Uhr mit Weiterleitung

Voraussetzung: $\mathfrak{u}$: Uhr, $v^{\hookrightarrow}$: Knoten, $\mathbf{m}^{\hookrightarrow}$: Nachricht

1: **function** GETTIMESTAMPFORMESSAGE($\mathfrak{u}, v^{\hookrightarrow}, \mathbf{m}^{\hookrightarrow}$)
2: $\quad\mathfrak{u}(v^{\hookrightarrow}) \leftarrow \mathfrak{u}(v^{\hookrightarrow}) + 1$ ▷ Uhr vorstellen
3: $\quad i \leftarrow < i^{\hookrightarrow}, \cdot, \cdot, \cdot, \cdot >= \mathbf{m}^{\hookrightarrow}$
4: $\quad t \leftarrow < \cdot, \cdot, t^{\hookrightarrow}, \cdot, \cdot >= \mathbf{m}^{\hookrightarrow}$
5: $\quad s \leftarrow < \cdot, \cdot, \cdot, s^{\hookrightarrow}, \cdot >= \mathbf{m}^{\hookrightarrow}$
6: $\quad \mathbf{d} \leftarrow < \cdot, \cdot, \cdot, \cdot, \mathbf{d}^{\hookrightarrow} >= \mathbf{m}^{\hookrightarrow}$
7: $\quad \mathbf{m}' \leftarrow < i, \mathfrak{u}(v^{\hookrightarrow}), t, s, \mathbf{d} >$
8: $\quad$ **return** $\mathbf{m}'$

1: **function** UPDATELAMPORTCLOCKBASEDONMESSAGE($\mathfrak{u}, v^{\hookrightarrow}, \mathbf{m}^{\hookrightarrow}$)
2: $\quad u \leftarrow < \cdot, u^{\hookrightarrow}, \cdot, \cdot, \cdot >= \mathbf{m}^{\hookrightarrow}$ ▷ Daten empfangen
3: $\quad \mathfrak{u}(v^{\hookrightarrow}) \leftarrow \max(\mathfrak{u}(v^{\hookrightarrow}), u) + 1$ ▷ Uhr aktualisieren
4: $\quad$ **return** $\mathfrak{u}$

3.3. Physikalische Übertragung

Aufgrund der Lizenz- und Genehmigungsfreiheit der Frequenzen, der Einfachheit der Beschaffung und der Kosten der Geräte sowie der Reproduzierbarkeit der Daten werden nur Frequenzen aus den ISM- und SDR-Bändern betrachtet. Dabei steht ISM für „Industrial, Scientific and Medical“ und SDR für „Short Range Device“. Genauer betrachtet werden das 2,4 GHz und das 868 MHz Band. Das erste ermöglicht eine höhere Übertragungsrate aufgrund der Trägerfrequenz. Dafür können im zweiten Band deutlich höhere Reichweiten erzielt werden. Als Multiplexverfahren werden sowohl zeitbasierte (Englisch: „Time Division Multiple Access“, kurz TDMA) als auch frequenzbasierte (Englisch: „Frequency Division Multiple Access“, kurz FDMA) Varianten angewendet. Weiterhin wird das geteilte Medium abgehört und es wird versucht, Signalinterferenzen bzw. Paketkollisionen aktiv zu verhindern (Englisch: „Carrier Sense Multiple Access / Collision Avoidance“, kurz CSMA/CA).

3.4. 2,4 GHz-Netzwerk

Im 2,4 GHz-Netzwerk wird der IEEE 802.11-Standard angewendet. Anhand dessen wird eine prädiktive Übertragungsratenschätzung entwickelt, sodass die Aspekte der Verbindungsqualität und der Verbundenheit mit dem Schwarm bei der Missionsplanung und Trajektoriengenerierung beachtet werden können. Dazu wird eine abstandsbasierte Qualitätsfunktion benötigt, die zu einer Kostenfunktion $\mathfrak{L}$ erweitert werden kann (siehe Abschnitt 5.5).

3.4.1. Übertragungsratenschätzung

Dieser Ansatz basiert auf der Idee aus der Masterarbeit Puzicha [141]. Bei der Übertragungsratenschätzung wird versucht das Übertragungs- und Störverhalten möglichst realitätsnah im Simulator abzubilden. Letzteres wird dabei aktiv durch Störsender und natürliches Umgebungsrauschen als auch passiv durch Abschattung aufgrund von Aufbauten, Hindernissen und Gebäuden berücksichtigt. Mittels des orthogonalen Frequenzmultiplexverfahrens (Englisch: „Orthogonal Frequency Division Multiplexing“, kurz OFDM) übertragene Signale werden auf Unterträger $u^{\leftrightsquigarrow}$ aufgeteilt (vgl. [103]). Bei einer Kanalbreite $\Delta f_c^{\leftrightsquigarrow}$ mit dem Trägerabstand $\Delta f_t^{\leftrightsquigarrow}$ kann die Unterträgerzahl $u^{\leftrightsquigarrow} \approx \frac{\Delta f_c^{\leftrightsquigarrow}}{\Delta f_t^{\leftrightsquigarrow}}$ unter Beachtung der Schutzabstände ermittelt werden. Der Trägerabstand bestimmt die Symbolzeit $t_s^{\leftrightsquigarrow}$:

$$t_s^{\leftrightsquigarrow} = \frac{1}{\Delta f_t^{\leftrightsquigarrow}} \tag{3.4.1}$$

Je Symbolzeit $t_s^{\leftrightsquigarrow}$ wird jeder Unterträger zeitgleich mit genau einem Symbol moduliert. Mögliche Modulationsverfahren und Techniken sind im Buch Freyer [42] gelistet; wobei sich diese Arbeit auf die vier in Tabelle 3.1 geführten Verfahren beschränkt. Auf Grundlage dieser Informationen lässt sich nun die Übertragungsrate $(\rho^{\leftrightsquigarrow})'$ in Bit pro

Tabelle 3.1.: Modulationsverfahren und deren Symbolbreiten.

Abkürzung	Bezeichnung	Symbolbreite $w^{\leftrightsquigarrow}$
QPSK	Quadrature Phase-Shift Keying	2 bit
16 QAM	Quadrature Amplitude Modulation	4 bit
64 QAM	Quadrature Amplitude Modulation	6 bit
256 QAM	Quadrature Amplitude Modulation	8 bit

Sekunde bestimmen:

$$\frac{\Delta f_c^{\leftrightsquigarrow}}{\Delta f_t^{\leftrightsquigarrow}} \approx u^{\leftrightsquigarrow} \tag{3.4.2}$$

$$(\rho^{\leftrightsquigarrow})' = \frac{u^{\leftrightsquigarrow} \cdot w^{\leftrightsquigarrow}}{t_s^{\leftrightsquigarrow}} = u^{\leftrightsquigarrow} \cdot w^{\leftrightsquigarrow} \Delta f_t^{\leftrightsquigarrow} \tag{3.4.3}$$

Beispiel 3.4.1: *Seien* $\Delta f_c^{\leftrightsquigarrow} = 10\,\text{MHz}$, $\Delta f_t^{\leftrightsquigarrow} = 15\,\text{kHz}$, $w^{\leftrightsquigarrow} = 8\,\text{bit}$. *So ergibt sich eine Basisdatenrate* $(\rho^{\leftrightsquigarrow})'$*:*

$$\frac{\Delta f_c^{\leftrightsquigarrow}}{\Delta f_t^{\leftrightsquigarrow}} = \frac{10\,\text{MHz}}{15\,\text{kHz}} \approx 600 = u^{\leftrightsquigarrow} \tag{3.4.4}$$

$$(\rho^{\leftrightsquigarrow})' = 72\,\text{Mbit} = 600 \cdot 8\,\text{bit} \cdot 15\,\text{kHz} = u^{\leftrightsquigarrow} \cdot w^{\leftrightsquigarrow} \cdot \Delta f_t^{\leftrightsquigarrow} \tag{3.4.5}$$

„Multiple Input Multiple Output"-Verfahren (kurz MIMO) ermöglichen die Verarbeitung paralleler Datenströme, die jeweils eine andere Kodierung und Polarisation nutzen, aber im selben Frequenzband liegen. Das 2,4 GHz bietet bis zu $v^{\leftrightsquigarrow} = 3$ parallele Ströme und das 5 GHz Band bis zu $v^{\leftrightsquigarrow} = 4$ Ströme (vgl. [9]). Folglich lässt sich für das Beispiel 3.4.1 die Bruttoübertragungsrate $\rho^{\leftrightsquigarrow}$ berechnen:

$$\rho^{\leftrightsquigarrow} = v^{\leftrightsquigarrow} \cdot (\rho^{\leftrightsquigarrow})' \tag{3.4.6}$$

Somit sind bei der Konfiguration im Beispiel 3.4.1 bis zu 288 Mbit brutto möglich. Allerdings ist die Nettoübertragungsrate $\rho^{\tilde{\leftrightsquigarrow}} = \eta\rho^{\leftrightsquigarrow}$, auch Nutzdatenrate genannt, aufgrund von Kontrollstrukturen der Übertragungsprotokolle geringer (vgl. [103]). Des Weiteren ist die Erzielung dieser Raten nur unter optimalen Bedingungen des Transportmediums möglich. Daher wird versucht anhand der Empfangsfeldstärke des Empfängers die tatsächliche Bruttoübertragungsrate $\rho^{\leftrightsquigarrow}$ anzunähern. Die Friis-Gleichung (vgl. [55]) beschreibt den Leistungszusammenhang vom Sender und Empfänger im freien Raum:

$$P_{r^{\leftrightsquigarrow}} = \underbrace{P_{s^{\leftrightsquigarrow}} + G_{s^{\leftrightsquigarrow}} + G_{r^{\leftrightsquigarrow}}}_{\text{Übertragungsleistung}} + \underbrace{20 \cdot \log_{10}\left(\frac{c^{\leftrightsquigarrow}}{f^{\leftrightsquigarrow} \cdot 4\pi \cdot d^{\leftrightsquigarrow}}\right)}_{\text{Freiraumdämpfung}} + P_{\mathcal{O}^{\leftrightsquigarrow}} \tag{3.4.7}$$

Die Empfangsleistung wird in Dezibel bezogen auf ein Milliwatt (dB mW) angegeben. So ergibt sich die Empfangsleistung $P_{r^{\leftrightsquigarrow}}$ (Englisch: „Receiver Signal Strength Index", kurz RSSI) eines Empfängers $r^{\leftrightsquigarrow}$ aus der Sendeleistung $P_{s^{\leftrightsquigarrow}}$ des Senders $s^{\leftrightsquigarrow}$ sowie der jeweiligen Antennenverstärkung $G_{r^{\leftrightsquigarrow}}$ und $G_{s^{\leftrightsquigarrow}}$ und der Freiraumdämpfung. $P_{\mathcal{O}^{\leftrightsquigarrow}}$ sei

Tabelle 3.2.: Abstrahierte Signalqualitäten $\gamma^{\leftrightsquigarrow}$ basierend auf RSSI $P_{r^{\leftrightsquigarrow}}$ und SNR $\varphi^{\leftrightsquigarrow}$.

		$P_{r^{\leftrightsquigarrow}}$ in [dB mW]								
	$\gamma^{\leftrightsquigarrow}$	−64	−65	−69	−73	−76	−78	−80	−81	$P_{-}^{\leftrightsquigarrow}$
$\varphi^{\leftrightsquigarrow}$ in [dB mW]	21	1	0,889	0,667	0,444	0,333	0,222	0,167	0,111	0
	20	0,889	0,889	0,667	0,444	0,333	0,222	0,167	0,111	0
	16	0,667	0,667	0,667	0,444	0,333	0,222	0,167	0,111	0
	12	0,444	0,444	0,444	0,444	0,333	0,222	0,167	0,111	0
	9	0,333	0,333	0,333	0,333	0,333	0,222	0,167	0,111	0
	7	0,222	0,222	0,222	0,222	0,222	0,222	0,167	0,111	0
	5	0,167	0,167	0,167	0,167	0,167	0,167	0,167	0,111	0
	4	0,111	0,111	0,111	0,111	0,111	0,111	0,111	0,111	0
	1	0	0	0	0	0	0	0	0	0

die Dämpfung der Hindernisse $o^{\leftrightsquigarrow} \in \mathcal{O}^{\leftrightsquigarrow}$ in der Übertragungsstrecke. Ferner sei $c^{\leftrightsquigarrow}$ die Lichtgeschwindigkeit, $f^{\leftrightsquigarrow}$ die Trägerfrequenz des Signals und $d^{\leftrightsquigarrow}$ der Abstand zwischen Sendeknoten $v_{s^{\leftrightsquigarrow}}^{\hookrightarrow}$ und Empfängerknoten $v_{r^{\leftrightsquigarrow}}^{\hookrightarrow}$ des Netzwerkgraphen $G^{\hookrightarrow}$:

$$d^{\leftrightsquigarrow} = \|e_{v_{s^{\leftrightsquigarrow}}^{\hookrightarrow} \to v_{r^{\leftrightsquigarrow}}^{\hookrightarrow}}\| = \|v_{r^{\leftrightsquigarrow}}^{\hookrightarrow} - v_{s^{\leftrightsquigarrow}}^{\hookrightarrow}\| \tag{3.4.8}$$

Das applizierte Ausbreitungsmodell der elektromagnetischen Wellen sei ein einfaches Strahlenmodell (Englisch: „Line Of Sight“, kurz LOS) ohne Reflexionsbetrachtung. Hindernisse, die die Strahlen schneiden, werden unter Subtraktion ihrer Dämpfung im Modell berücksichtigt.

3.4.2. Störeinflüsse

Als weiterer Störeinfluss auf das Signal wird das natürliche Wärmerauschen bzw. Umgebungsrauschen ermittelt (vgl. [16]):

$$P_e' = 10\log_{10}(k_B T^{\leftrightsquigarrow} \Delta f_c^{\leftrightsquigarrow} \cdot 1000) = 10\log_{10}(k_B T^{\leftrightsquigarrow} \cdot 1000) + 10\log_{10}(\Delta f_c^{\leftrightsquigarrow}) \tag{3.4.9}$$

Dabei sei k_B die Boltzmann-Konstante. Bei Raumtemperatur von $T^{\leftrightsquigarrow} = 300\,\mathrm{K}$ ergibt sich eine Vereinfachung:

$$P_e^{\leftrightsquigarrow} = -174\,\mathrm{dB\,mW} + 10\log_{10}(\Delta f_c^{\leftrightsquigarrow}) \tag{3.4.10}$$

Für das Beispiel 3.4.1 ergibt sich somit ein Grundrauschen von $P_e^{\leftrightsquigarrow} = -104\,\mathrm{dB\,mW}$. Weitere aktive Sender oder Störsender $a^{\leftrightsquigarrow} \in \mathcal{A}^{\leftrightsquigarrow}$ in der Umgebung erhöhen das Rauschniveau ebenfalls; so ergibt sich ein Signalrauschabstand (Englisch: „Signal to Noise Ratio“, kurz SNR) von:

$$\varphi^{\leftrightsquigarrow} = P_{r^{\leftrightsquigarrow}} - (P_e^{\leftrightsquigarrow} + P_{\mathcal{A}^{\leftrightsquigarrow}}) \tag{3.4.11}$$

Basierend auf den Veröffentlichungen Sekar [167] und Puzicha [141] lassen sich Signalqualitätsabhängigkeiten $\gamma^{\leftrightsquigarrow}$ für RSSI $P_{r^{\leftrightsquigarrow}}$ und SNR $\varphi^{\leftrightsquigarrow}$ abstrahieren (siehe Tabelle 3.2), sodass sie sich nicht mehr auf einen Protokollstandard beziehen.

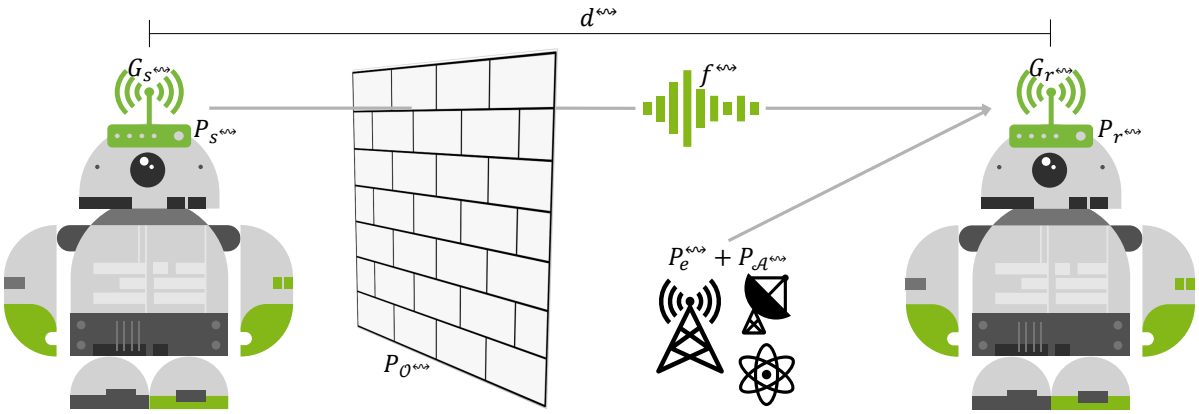


Abbildung 3.1.: Darstellung der Übertragungsgrößen.

Ein vollständiger Verbindungsabbruch entsteht, falls entweder die Empfangsleistung (RSSI) $P_r^{\leftrightsquigarrow}$ unter einen gerätespezifisches Sensitivitätsminimum $P_{\_}^{\leftrightsquigarrow}$ fällt oder der SNR $\varphi^{\leftrightsquigarrow}$ kleiner oder gleich eins ist, sodass die Daten im Signal nicht mehr vom Rauschen extrahiert werden können. Der Gesamtzusammenhang ist in Abbildung 3.1 visualisiert.

3.5. „Long Range“ Netzwerk

Die hohe Trägerfrequenz von 2,4 GHz bzw. 5 GHz für den IEEE 802.11-Standard und die geringe erlaubte Sendeleistung im SDR- bzw. ISM-Band ermöglichen nur vergleichsweise geringe Reichweiten, eine erhöhte Rauschanfälligkeit sowie eine große Leistungsabsorption bei Hindernissen. Daher wird ebenfalls eine komplementäre Technologie für große Reichweiten und eine robuste, störungsunempfindliche Signalübertragung mit geringer Übertragungsrate $\rho^{\leftrightsquigarrow}$ auf ihre Anwendbarkeit für Roboterschwärme hin untersucht. Als Technologie wird „Long Range“ (kurz LoRa) ausgewählt, da es bereits einige Ansätze zur Nutzung im Katastrophenschutz und als Knotenpunktnetzwerk gibt. Diese Einsatzbereiche entsprechen den in der Einleitung (siehe Kapitel 1) formulierten Zielen. LoRa ist eine Modulationstechnologie, die sich durch eine große Reichweite und eine hohe Resistenz gegenüber Störungen durch andere Funksignale auszeichnet. Sie hat jedoch eine niedrige Symbolrate. LoRa ist eine patentierte Modulation, die sich derzeit im Besitz der Firma *Semtech Cooperation* befindet. Nach den Grundlagen der Modulation und der Analyse des Standes der Technik wird gezeigt, wie sich LoRa für mobile Roboterschwärme nutzen lässt und welche Übertragungsrate erreicht werden kann.

3.5.1. Signal- und Zirpfrequenzspreizung

Während bei den vorherigen Modulationsverfahren (siehe Abschnitt 3.3) die Information Symbol für Symbol auf das Trägersignal moduliert wird, wird bei der Signalspreizung das Signal zunächst erweitert. Jedes Symbol wird mit einem Spreizcode multipliziert. Dieser Code ist definiert als eine Folge von Symbolen, die als Chips bezeichnet

werden und dem Sender und dem Empfänger bekannt sind. Die Multiplikation ist als bitweise *XOR*-Operation definiert. Das Verhältnis zwischen der Chiprate $\rho_c^{\leftrightsquigarrow}$ und der ursprünglichen Symbolrate $\rho_s^{\leftrightsquigarrow}$ wird als Spreizfaktor (Englisch: „Spreading Factor“, kurz *SF*) bezeichnet:

$$SF = \frac{\rho_c^{\leftrightsquigarrow}}{\rho_s^{\leftrightsquigarrow}} \tag{3.5.1}$$

Da jedes Symbol durch mindestens einen Chip kodiert wird, muss $SF \geq 1$ gelten. Da ein Symbol durch mehrere Chips dargestellt wird, kommt es natürlich zu einer erheblichen Reduzierung der Symbol-Bitrate. Diese Chips werden mit einem klassischen Modulationsverfahren, z. B. der Phasenumtastung (Englisch: „Phase Shift Key“, kurz PSK), übertragen. Durch die Erweiterung des Signals ergibt sich ein Gewinn an Genauigkeit in dB (siehe [40]):

$$G_p = 10 \cdot log_{10}(SF)\text{dB} \tag{3.5.2}$$

Diese Art der Signalspreizung wird „Direct Sequence Spread Spectrum“ (kurz DSSS) genannt. Ein Nachteil dieses Verfahrens ist die genaue zeitliche Synchronisierung von Sender und Empfänger. Ein Versatz von einem Symbol führt zu fälschlicher Datenrekonstruktion aus dem Signal.
Dem gegenüber ist Zirpfrequenzspreizung (Englisch: „Chirp Frequency Spreading“, kurz CFS) eine Technologie zur Streuung des Trägersignals über die gesamte Bandbreite. Dazu wird die Trägerfrequenz variiert, um die gesamte Bandbreite abzudecken. Ein Durchlauf von einem Ende des Frequenzspektrums der Bandbreite bis zum Anderen wird als Zirp (Englisch: „Chirp“) bezeichnet. Die Modulation der Symbole erfolgt durch die Zirprichtung. Somit kann ein Durchlauf durch die Frequenzen von der höheren zur niedrigeren Frequenz als Null kodiert werden und entspricht einer fallenden Flanke. Die logische Eins kann entsprechend invers gebildet werden. Aufgrund der kontinuierlichen Frequenzvariation ist diese Modulationstechnik sehr resistent gegenüber Signalinterferenzen auf einzelnen Funkkanälen. Weiterhin kann die Zirpfrequenzspreizung das Synchronisationsproblem zwischen Sender und Empfänger lösen. Ein Zirp dient als Markierung eines exakten Zeitpunktes zur Synchronisierung des Spreizcodes.

3.5.2. LoRa-Modulation

Die LoRa-Modulation verwendet eine Kombination aus Signalspreizung und Zirpfrequenzspreizung, die eine genaue Synchronisierung von Chipsequenzen auch auf weniger komplexen und damit kostengünstigen Schaltungen ermöglicht. Im Gegensatz zur Signalspreizung gibt es keinen definierten Spreizcode. Der Spreizfaktor gibt hier die Länge der Teilsequenzen an, in die die Bitfolge des Modulationssignals unterteilt wird. Jede dieser Teilsequenzen ist ein LoRa-Symbol, das auf dem Trägersignal als linearer Durchlauf durch das Frequenzband sichtbar ist und als Zirp bezeichnet wird. Ein Aufwärtslauf wird als Aufzirp (Englisch: „‚Up-Chirp‘ “) und ein Abwärtslauf als Abzirp (Englisch: „‚Down-Chirp‘ “) bezeichnet. Ist der Zirp über die gesamte Symboldauer $t_s^{\leftrightsquigarrow}$ kontinuierlich, wird er als Basiszirp (Englisch: „‚Raw-Chirp‘ “) bezeichnet. Die Anzahl aller darstellbaren Symbole $|\mathcal{W}^{\leftrightsquigarrow}|$ ist durch den Spreizfaktor *SF*

mit $|\mathcal{W}^{\leftrightsquigarrow}| = 2^{SF}$ definiert. Die Frequenz eines einzelnen Basiszirps in Bezug auf die Zeit t und die Trägerfrequenz $f^{\leftrightsquigarrow}$ wird wie folgt berechnet (vgl. [40]):

$$f_r^{\leftrightsquigarrow}(t) = f^{\leftrightsquigarrow} \pm \frac{\Delta f_c^{\leftrightsquigarrow}}{t_s^{\leftrightsquigarrow}} \cdot t \tag{3.5.3}$$

Sei $t \in \left[-\frac{t_s^{\leftrightsquigarrow}}{2}, \frac{t_s^{\leftrightsquigarrow}}{2}\right]$ und $\Delta f_c^{\leftrightsquigarrow}$ bezeichne die mögliche Bandbreite bzw. Kanalbreite. Das Zeichen entspricht einem Aufzirp oder Abzirp. Jede Übertragung beginnt mit 12 Basiszirps, die als Präambel bezeichnet werden. Davon sind die ersten 8 Aufzirps, gefolgt von 2 Abzirps. Diese werden hauptsächlich verwendet, um das Referenzzeitfenster für die kommende Modulation zu definieren. Die Anzahl der möglichen Symbole $|\mathcal{W}^{\leftrightsquigarrow}|$ definiert auch die Anzahl der unterschiedlichen Phasensprünge, die innerhalb eines Zirp kodiert werden können. Alle LoRa-Symbole sind mit $w^{\leftrightsquigarrow} \in \{0, ..., |\mathcal{W}^{\leftrightsquigarrow}| - 1\}$ nummeriert. Beginnend mit dem ersten zu übertragenden Symbol wird ein Zirp gesendet, das kein Basiszirp ist, sondern eine um die Zeit $\tau_{w^{\leftrightsquigarrow}} = \frac{w^{\leftrightsquigarrow}}{\Delta f_c^{\leftrightsquigarrow}}$ verzögerte Version. Für das LoRa-Symbol mit dem Index $w^{\leftrightsquigarrow}$, bei dem $\tau_{w^{\leftrightsquigarrow}} = 0$ gilt, wird also ein kontinuierlicher Basiszirp übertragen. Andere Symbole enthalten einen Frequenzsprung des Basiszirps bei:

$$t + \tau_{w^{\leftrightsquigarrow}} \tag{3.5.4}$$

Damit ist der gesamte Frequenzgang innerhalb der Periode eines LoRa-Symbols $f_r^{w^{\leftrightsquigarrow}}(t)$ definiert als:

$$f_r^{w^{\leftrightsquigarrow}}(t) = \begin{cases} \frac{\Delta f_c^{\leftrightsquigarrow}}{t_s^{\leftrightsquigarrow}}(t - \tau_{w^{\leftrightsquigarrow}}) + \Delta f_c^{\leftrightsquigarrow} & \text{if } t \in \left[-\frac{t_s^{\leftrightsquigarrow}}{2}, -\frac{t_s^{\leftrightsquigarrow}}{2} + \frac{w^{\leftrightsquigarrow}}{\Delta f_c^{\leftrightsquigarrow}}\right] \\ \frac{\Delta f_c^{\leftrightsquigarrow}}{t_s^{\leftrightsquigarrow}}(t - \tau_{w^{\leftrightsquigarrow}}) & \text{if } t \in \left[-\frac{t_s^{\leftrightsquigarrow}}{2} + \frac{w^{\leftrightsquigarrow}}{\Delta f_c^{\leftrightsquigarrow}}, \frac{\Delta f_c^{\leftrightsquigarrow}}{2}\right] \end{cases} \tag{3.5.5}$$

Der Empfänger muss dann nur noch die Verschiebung $\tau_{w^{\leftrightsquigarrow}}$ der Phase zum Basiszirp berechnen, mit der er über $w^{\leftrightsquigarrow} = \tau_{w^{\leftrightsquigarrow}} \cdot \Delta f_c^{\leftrightsquigarrow}$ das Symbol und damit die Bits des Modulationssignals wiederherstellen kann.

3.5.3. Übertragungsrate der LoRa-Modulation

Ein wichtiger Aspekt der LoRa-Modulation für mobile ad hoc Netze ist die Übertragungsrate von Kontrollnachrichten. Die Beziehung zwischen der Kanalbandbreite $\Delta f_c^{\leftrightsquigarrow}$, dem Spreizfaktor SF und der Chiprate $\rho_c^{\leftrightsquigarrow}$ ist wie folgt: Pro Hertz der Bandbreite $\Delta f_c^{\leftrightsquigarrow}$ wird ein Chip pro Sekunde gesendet. Dies führt zu einer Chiprate $\rho_c^{\leftrightsquigarrow}$ von:

$$\rho_c^{\leftrightsquigarrow} = \frac{\Delta f_c^{\leftrightsquigarrow}}{1\,\mathrm{s}} = [\mathrm{Hz\,s^{-1}}] \tag{3.5.6}$$

Zur Kodierung eines LoRa-Symbols sind $|\mathcal{W}^{\leftrightsquigarrow}|$ Chips erforderlich, da es $|\mathcal{W}^{\leftrightsquigarrow}| = 2^{SF}$ verschiedene Phasensprünge gibt. Die Symboldauer $t_s^{\leftrightsquigarrow}$ wird also wie folgt festgelegt:

$$t_s^{\leftrightsquigarrow} = \frac{2^{SF}}{\Delta f_c^{\leftrightsquigarrow}} \tag{3.5.7}$$

Tabelle 3.3.: Übertragungsraten für verschiedene mögliche Konfigurationen.

SF	$\Delta f_c^{\leftrightsquigarrow}$	CR	$\rho^{\leftrightsquigarrow}$
7	500Hz	4/5	21875bit/s
7	250Hz	4/5	10938bit/s
8	250Hz	4/5	6250bit/s
9	250Hz	4/5	3516bit/s
10	250Hz	4/5	1953bit/s
11	250Hz	4/5	1074bit/s
12	250Hz	4/5	586bit/s

Jedes Symbol kodiert eine Anzahl von SFbit des Modulationssignals. Folglich ist die Basisbitrate $(\rho^{\leftrightsquigarrow})'$ definiert als:

$$(\rho^{\leftrightsquigarrow})' = SF \cdot \frac{\Delta f_c^{\leftrightsquigarrow}}{2^{SF}} \tag{3.5.8}$$

Die LoRa-Spezifikation enthält ein zusätzliches Fehlerkorrekturverfahren, das die Basisübertragungsrate $(\rho^{\leftrightsquigarrow})'$ reduziert. Bei der Kodierrate wird eine Paritätsinformation mitgeschickt, um das Signal robuster zu machen. Diese Information wird mit $CR \in \left\{\frac{4}{5}, \frac{4}{6}, \frac{4}{7}, \frac{4}{8}\right\}$ beschrieben und reduziert die Übertragungsrate um diesen Faktor (vgl. [40]):

$$\rho^{\leftrightsquigarrow} = CR \cdot (\rho^{\leftrightsquigarrow})' \tag{3.5.9}$$

Die Funktion wird von 2^{SF} dominiert. Mit steigendem Spreizfaktor nimmt $\rho^{\leftrightsquigarrow}$ bei konstanter Bandbreite $\Delta f_c^{\leftrightsquigarrow}$ und Kodierrate CR fast logarithmisch zur Basis 2 ab. Die theoretische Bitrate $\rho^{\leftrightsquigarrow}$ wird pro Paket durch das Präambel von 12 Zirps, welches 12,25 $t_s^{\leftrightsquigarrow}$ entspricht, einem Header von 20 bit, der immer mit $CR = \frac{4}{8}$ gesendet wird, und einem möglichen zyklischen Redundanzprüfungscode (Englisch: „Cyclic Redundancy Check“, kurz CRC) von 2 B zur tatsächlichen Nutzdatenrate $\rho^{\leftrightsquigarrow}$ reduziert. Mögliche Übertragungsraten sind in der Tabelle 3.3 aufgeführt.

3.5.4. Stand der Technik

Es wird zunächst ein Überblick über die relevante Forschung und die hier angestrebten Erweiterungen gegeben. Eine Studie von Berto, Napoletano und Savi [14] befasst sich hauptsächlich mit der Entwicklung einer flexiblen LoRa-basierten Netzwerklösung, die eine interne Knoten-zu-Knoten-Kommunikation zwischen LoRa-Endgeräten ohne Zugangsknoten (Englisch: „Gateway“) ermöglicht, insbesondere für Katastrophen. Die Lösung beinhaltet Multi-Hop-Knotenpunktnetzwerkfunktionen für die Kommunikation über größere Entfernungen. Um eine günstige und einfache Kommunikation ohne Internet zu gewährleisten, wird dabei ein reduzierterer Protokollstapel (Englisch: „Networking Stack“) als LoRaWAN vorgesehen. Dieser wird auf der konventionellen physikalischen LoRa-Schicht aufgesetzt. Dadurch können die teilnehmenden Knoten nicht nur als Endpunkte für die Datenerfassung dienen, sondern auch komplexe Aufgaben lokal berechnen. Allerdings werden die Versuche mit einer Nutzlast von nur 240 B

durchgeführt. Der Ansatz erweist sich als nützlich für Anwendungen, die nur wenige gleichzeitige Übertragungen benötigen. Weil die Versuche in einem Labor durchgeführt werden, wird die Multi-Hop-Kommunikation nicht in realen Umgebungen unter Verwendung echter Anwendungen bewertet.

In einem weiteren Beitrag von C. Ebi u. a. [17] werden die Hauptprobleme der Reichweite, der Paketübertragungsrate (Englisch: „Packet Delivery Rate“, kurz PDR) und des Energieverbrauchs bei der Übertragung von Prozessüberwachungsdaten von entfernten oder unterirdischen Standorten behandelt. Ein Feldexperiment unterstützt die Autoren, die bestehenden Einschränkungen mit der Entwicklung einer LoRa-basierten Knotenpunktnetzwerkarchitektur anzugehen, die es unterirdischen Sensorknoten (kurz SN) ermöglicht, sich in bestehende LoRaWAN-Netzwerke unter Verwendung von zwischengeschalteten Weiterleitungsknoten (Englisch: „Repeater Node“, kurz RN) zu integrieren. Ein entscheidender Aspekt dieses Ansatzes ist die zeitschlitzbasierte Übertragung zwischen exakt synchronisierten SN und RN. Trotz der Verbesserung der Übertragungsqualität und der Verringerung der Paketfehlerraten weist diese Studie einige Einschränkungen auf. Aufgrund der inhärenten Nutzlastbeschränkung ist die maximale Anzahl von Sensorknoten auf fünf begrenzt. Das Problem des Zentralenausfallknotens und sicherheitsrelevanter Probleme werden durch die vorgestellte Strategie nicht vollständig gelöst. Wenn außerdem ein SN ständig daran scheitert, einem Teilnetz beizutreten, weil kein RN verfügbar ist, wird viel Energie in Form von Sendeleistung verbraucht. Zudem wird leider nur die Single-Hop-Implementierung einem umfassenden Feldtest unterzogen.

Die Veröffentlichung V. D. Pham u. a. [187] befasst sich mit ähnlichen Problemen. Die Übertragungslatenz und die Paketverlustrate einer auf der LoRa-Technologie basierenden Knotenpunktnetzwerkarchitektur werden mit Hilfe eines Simulationsmodells im Simulator OMNET++ (vgl. [84]) unter Berücksichtigung verschiedener Konfigurationsfaktoren wie dem Spreizfaktor und der Kodierungsrate untersucht. Aus den modellbasierten Erkenntnissen werden Schlussfolgerungen hinsichtlich des Potenzials für den Einsatz von LoRa-Knotenpunktnetzwerken zur Erhöhung der Netzabdeckung und zur Integration anderer Kurzstreckennetzen gezogen. Allerdings weist das Netzwerk eine hohe Verzögerung bei der Datenübertragung auf. Dabei ist hervorzuheben, dass der beschriebene Algorithmus zur Berechnung einer Route auf den Signalstärkeinformationen zwischen allen Knoten beruht (vgl. [187, S. 1282]). Es wird nicht erläutert wie ein spezifischer Knoten diese Informationen erhalten soll. Weiterhin wird nicht beachtet, dass die am Markt verfügbaren LoRa-Module nur im Sende- oder Empfangszustand, aber niemals in beiden Zuständen gleichzeitig, sein können. Bei der Betrachtung der Tabelle der verwendeten Parameter in V. D. Pham u. a. [187, S. 1283] wird deutlich, dass ein ad hoc Netzwerk mit sehr geringer Last simuliert wird. Es werden Pakete mit einer zufälligen Größe zwischen 20 B und 150 B versendet. Das Sendeintervall ist im Mittel 120 s. Dies steht offensichtlich im starken Kontrast zu dem aus den Anforderungen abgeleiteten Netzwerk, das im Grenzbereich betrieben werden muss. Aus Abbildung 7 der Veröffentlichung V. D. Pham u. a. [187, S. 1284] geht eine Empfangswahrscheinlichkeit von 99,94 % hervor, falls ein solches Paket über 10 Knoten geleitet wird. In einem simplen ad hoc Netzwerk, welches andere Netzwerkknoten nicht berücksichtigt, wird ein Knoten genau dann senden, falls Daten anliegen.

Aufgrund der in dieser Arbeit entstehenden hohen Datenlast würde stetig gesendet werden, wodurch auch kein Paket empfangen werden könnte. Dies würde in einer Paketverlustrate von 100 % resultieren. Folglich müssen vor allem hohe Auslastungszustände berücksichtigt werden. Ein verbessertes LoRa-Protokoll wird in der Arbeit H. Huh und J. Y. Kim [53] vorgeschlagen, das in privaten Netzwerken für „Internet of Things"-Anwendungen (kurz IoT) eingesetzt werden kann.
Dieses Protokoll nutzt Knotenpunktnetzwerke, um die Netzwerkabdeckung zu verbessern, und bildet eine Kombination aus zeitschlitzbasiertem und ereignisdiskretem Verfahren (Englisch: „Time-slotted Event-Driven System", kurz TEDS), um die Kollisionsrate zu reduzieren. In der Veröffentlichung H. Lee und K. Ke [54] wird eine empirische Bewertung der Leistung eines LoRa-Knotenpunktnetzwerks in einem großen Stadtgebiet durchgeführt, indem die Übertragung und Sammlung von Sensordaten überwacht wird. Der Nachteil der vorgestellten Methode ist, dass sie aufgrund der hohen Latenz zwischen der Datengenerierung in einem Knoten und dem Hochladen zum Zugangsknotenpunkt nicht für harte Echtzeitsysteme mit kurzer Antwortzeit geeignet ist. Außerdem bestehen Bedenken hinsichtlich der Sicherheit eines solchen Netzes und ein erheblicher Stromverbrauch ist in solchen Systemen unvermeidlich.
Eine ähnliche Studie von J. G. Panicker, M. Azman und R. Kashyap [72] schlägt den Entwurf eines LoRa-basierten Knotenpunktnetzwerks zur Verfolgung von Tieren und zur Überwachung ihrer Stati durch in ihren Halsbändern eingebettete Sensoren über weite Gebiete wie Felder oder Wälder vor. Es werden verschiedene Modelle erarbeitet, die auf der Größe des abzudeckenden Gebietes und seinen geografischen Merkmalen basieren. Als erstes wird ein Knotenpunktnetzwerk mit mehreren LoRa-Zugangsknotenpunkten, zweitens ein Netzwerk mit LoRa-Knoten für dichtere bewachsene Regionen und drittens eine Kombination aus beiden für Gebiete, in denen die Verteilung der Fauna ungleichmäßig ist, postuliert. Dabei handelt es sich jedoch nur um ein theoretisches Konzept und es ist eine empirische Untersuchung erforderlich, um die PDR für jedes Modell zu bestimmen, da sie weitgehend von der Geomorphologie der Region und vielen anderen Aspekten abhängt.
Eine Modifikation der bestehenden LoRa-Knotenpunktnetzwerkprotokolle wird vorgestellt, um den Energieverbrauch zu minimieren (vgl. [58]). Dabei wird LoRa in einer Viehüberwachungsanwendung eingesetzt, indem ein mobiles ad hoc Knotenpunktnetzwerk aufgebaut wird, um Positions- und Beschleunigungsdaten von Rindern zu sammeln. Die gesammelten Daten werden dann an eine Basisstation übertragen. Um den Energieverbrauch zu senken, sorgen die Satellitenpositionssensoren für eine globale Zeitsynchronisierung, sodass spezifische kollisionsfreie Sendezeiten vergeben werden können. Um Manipulationen (Englisch: „Spoofing") zu vermeiden und die Vertraulichkeit des Nachrichtenaustauschs zwischen den Sensoren und der Basisstation zu gewährleisten, werden Authentifizierungs- und Verschlüsselungstechniken eingesetzt. Die Sicherheit des Systems wird jedoch nicht getestet und muss bewertet werden, bevor diese Architektur in der Datenerfassung eingesetzt wird.
Neben der Literatur existieren praktische Ansätze für ad hoc Netzwerke basierend auf der LoRa Modulation. Ein Beispiel ist das *Meshtastic* Projekt (vgl. [60]), das Firmware für verschiedene LoRa-Module sowie Schnittstellen für Smartphones und Rechnersysteme entwickelt. Es ist zudem eine „Android" Anwendung verfügbar, welche das ad

hoc Netzwerk zum Übermitteln von Textnachrichten verwendet (vgl. [61]). Diese Anwendung unterstützt ebenfalls verlässliche Übertragungen, bei denen ein Datenpaket erneut gesendet wird, falls keine Bestätigung vom Empfänger empfangen worden ist. Allgemeine Rundfunknachrichten werden ebenfalls unterstützt. Außerdem ist eine Anbindung des „Global Positioning System“ (kurz GPS) der Module möglich.
Zusammenfassend lässt sich sagen, dass LoRa-Netzwerke in zahlreichen Studien für eine Vielzahl von IoT-Anwendungen genutzt und getestet werden. Frühere Entwicklungen sind in der Lage kurze Datenpakete von bis zu 80 Byte zu übertragen und arbeiten typischerweise mit statischen Knotenpunktnetzwerken. Ebenfalls existieren Ansätze zur Überwachung natürlicher Herden bzw. Schwärme. Im Gegensatz dazu konzentriert sich diese Arbeit auf die Verwendung eines kostengünstigen LoRa-basierten Netzwerks in einer wesentlich größeren Umgebung von bis zu 25 km^2, das in der Lage ist, relativ umfangreiche Daten mit einer hohen Frequenz in einer sich dynamisch verändernden Netzwerktopologie zu übertragen. Dazu wird versucht die Technologie im Grenzbereich zu betreiben.

3.6. Nachrichten und Pakete

Zunächst wird kurz der Unterschied zwischen Nachrichten und Paketen definiert:

Definition 3.6.1: *Eine Nachricht beschreibt eine über das Netzwerk zu versendende Informationseinheit, die von anderen Prozessen weiter verarbeitet wird. Nachrichten können je nach Datengröße als ein Paket oder fragmentiert mittels mehrerer Pakete über das Netzwerk verschickt werden. Anhand der Paketköpfe eines Pakets können die Nachrichten von einem Paketempfänger rekonstruiert werden.*

Die Agenten des Schwarms werden durch kurze Nachrichten gesteuert und übertragen ihre Regelungsdaten per Nachbarschaftsrundfunk (siehe Definition 3.1.2). Der Simulator (siehe Abschnitt 4.2) emuliert ebenfalls ein ad hoc Knotenpunktnetzwerk auf Basis der in Abschnitt 3.3 vorgestellten Zusammenhänge. Diese Netzwerke lassen sich auch durch Netzwerkbrücken (siehe Abschnitt 4.2.17) verbinden. Dadurch lassen sich reale und virtuelle Agenten kombiniert betreiben. Die in Abschnitt 4.2.13 beschriebenen Nachrichten können in fünf Kategorien zusammengefasst werden und weisen die in Tabelle 3.4 gelisteten Größen auf. Die erste Kategorie (siehe Tabelle 3.4) umfasst zwei Nachrichten, um das Netzwerk, die Modelle und die Zeitsynchronisation aller Agenten zu erhalten. Sie benötigen nur wenige Bytes pro Nachricht und haben eine Rate von 0,1 Hz. Die „Sync“-Nachricht wird ereignisbasiert gesendet, sobald die Zeitschritte nicht mehr synchron sind. Dies geschieht etwa alle 10 min. Manchmal treten ereignisgesteuerte lokale Hindernismeldungen auf.
Die dritte Kategorie, die die größte Datenrate benötigt, beinhaltet Trajektorienaktualisierungspakete der Agenten, die einen Bedarf von 1468 B pro Nachricht erzeugen. Diese Pakete werden periodisch alle 300 ms generiert und benötigen daher eine Nutzdatenrate von mehr als 4894 $\mathrm{B\,s^{-1}}$, um kontinuierlich übertragen zu werden. Daher wird in Abschnitt 4.2.13 eine modellbasierte Kompression präsentiert, die die Paketgröße auf 144 B komprimiert.

Tabelle 3.4.: Nachrichtentypen des Autonomiekerns für Agenten und ihre Größen (siehe Abschnitt 4.2.13).

Kategorie	Nachrichtentyp	Nachrichtengröße
1	Alive	28 B
	Sync	32 B
2	Static Obstacle	40 B
3	Robot	1468 B
	Robot Short	144 B
4	Chain Transport Ack	20 B
	Chain Transport	28 B
	Convoy	40 B
	Mission Follow Path	? + 96 B
	Mission Multiple Targets	? + 80 B
	Mission	28 B
	Transportation	32 B
	Triangle Line Update	16 B
	Triangle Position Update	20 B
5	Data	? + 56 B

Die nächste Kategorie hat die höchste Priorität unter allen Nachrichten, da die Nachrichten von allen Agenten empfangen werden müssen, um ihr Verhalten durch Starten und Beenden der Missionen zu steuern. Sie werden als allgemeiner Rundfunk (siehe Definition 3.1.1) übertragen. Allerdings sind diese Nachrichten im Vergleich zur vorherigen Kategorie klein. Für einige Missionen gibt es spezialisierte Nachrichten, z. B. definiert die Nachricht „Mission Follow Path“ den Pfad, dem ein oder mehrere Agenten folgen sollen. Tabelle 3.4 gibt die Größe einer solchen Nachricht mit einem „?“ an, da sie mit der Anzahl der Wegpunkte in der Nachricht variiert. Jeder Wegpunkt erfordert zusätzliche 8 B. Ähnlich verhält es sich mit der Anzahl der Ziele für eine Konvoibegleitmission, die die Größe der Nachricht „Mission Multiple Targets“ erhöht. Die letzte Kategorie besteht aus allen anderen Nachrichtentypen und enthält derzeit eine Datennachricht, die beliebige anwendungsbezogene Daten enthalten kann und nur auf 60 kB per Definition begrenzt ist. Obwohl die Häufigkeit der Nachrichten dieser Kategorie völlig unbekannt ist, kann davon ausgegangen werden, dass sie selten vorkommen oder via anderer Kommunikationskanäle übertragen werden, anstatt über die Steuerungsschnittstelle. Die Nachrichten werden in den Kapiteln 4 und 5 detailliert betrachtet.

3.6.1. Fragmentierung und Kompression

Um Netzwerke mit niedrigen Nutzdatenraten wie LoRa nutzen zu können, muss die von der Anwendung benötigte Nutzdatenmenge auf wenige Bytes reduziert werden. Für die Anwendung innerhalb des Roboterschwarms sind insbesondere die „Robot“-Nachrichten (siehe Tabelle 3.4) das Hauptproblem aufgrund der großen Größe und der

kurzen Senderate von $\frac{1}{0,3}$Hz. Diese Nachrichten erfordern eine Übertragungsrate von $39\,227\,\mathrm{bit\,s^{-1}}$ pro Knoten, wobei alle anderen Nachrichtentypen vernachlässigt werden. Folglich ist die benötigte Datenrate höher als die vom LoRa-Standard maximal bereitgestellte Übertragungsrate (siehe Tabelle 3.3). Daher werden zwei Verfahren eingeführt, um große Datenpakete zu verschicken. Zum Einen werden Datenpakete von bis zu 60 kB fragmentiert verschickt. Damit das Netzwerk nicht verstopft wird, dürfen diese aber nur eine geringe Ankunftsrate aufweisen. Die Daten mit einer ebenfalls hohen Ankunftsrate müssen zuvor komprimiert werden. Die modellbasierte Kompression als Teil des Programmkerns wird in Abschnitt 4.2.13.1 beschrieben. Jedoch erfolgt der Betrieb des Netzwerks auch mittels Kompression im Grenzbereich der Übertragungsrate, sodass stets darauf geachtet werden muss keine veralteten Daten und möglichst wenige zusätzliche Kontrollnachrichten zu versenden.

3.7. Netzwerkprotokoll mobiler ad hoc Netzwerke mit großer Reichweite

Der im Folgenden beschriebene Netzwerkstapel ist für eine Architektur mit zwei echtparallelen Prozessen, zum Beispiel eine Zweikernarchitektur, ausgelegt. Ein Prozesskern wird für die serielle Kommunikation mit der Anwendungsschicht verwendet, um Nachrichten als serielle Pakete zu senden und zu empfangen (siehe Abbildung 3.2). Ein zweiter Prozess wickelt die gesamten Sende- und Empfangsfunktionen für das LoRa-Modul über eine serielle Peripherieschnittstelle (Englisch: „Serial Peripheral Interface“, kurz SPI) ab (siehe Abbildung 3.3 und 3.4). Des Weiteren wird aufgrund der Technologie und der am Markt verfügbaren Module davon ausgegangen, dass sich das Modul entweder im Empfangs- oder Sendezustand befinden kann.

3.7.1. Serielle Kommunikation mit der Anwendungsschicht

Der Prozess zur Kommunikation mit der Anwendung, hier als „Prozess 1“ definiert, beginnt mit einer Prüfung, ob neue Datenpakete verfügbar sind. Jede Übertragung beginnt mit „Magic Bytes“, um die Übertragung zu synchronisieren und den Beginn der Nutzlast anzuzeigen. Das serielle Nutzlastpaket beginnt immer mit einem Paketkopf (Englisch: „Header“), der die Größe der Nutzlast angibt, um den erforderlichen Speicher auf der Hardware zu reservieren. Falls die Daten korrekt empfangen werden, wird die Rücksprungfunktion (Englisch: „Callback Function“) „OnSerialPacketReceive“ aufgerufen. Diese Funktion extrahiert den Paket- bzw. Nachrichtentyp. Es existieren Datenpakete, die über das Netzwerk versendet werden sollen, und Steuerungspakete zur Abfrage und Konfiguration der jeweiligen Netzwerkknoten. Falls das zu verarbeitende Paket ein Datenpaket beinhaltet, das via LoRa übertragen werden soll, so werden diese Daten aus dem Paket in den statischen Direktzugriffspeicher (Englisch: „Static Random Access Memory“, kurz SRAM) kopiert und das Informationsbit (Englisch: „Flag“) „PackageReady“ wird auf wahr gesetzt. Dieses Bit zeigt dem anderen Prozess, als „Prozess 2“ definiert, die Verfügbarkeit neuer Daten im SRAM

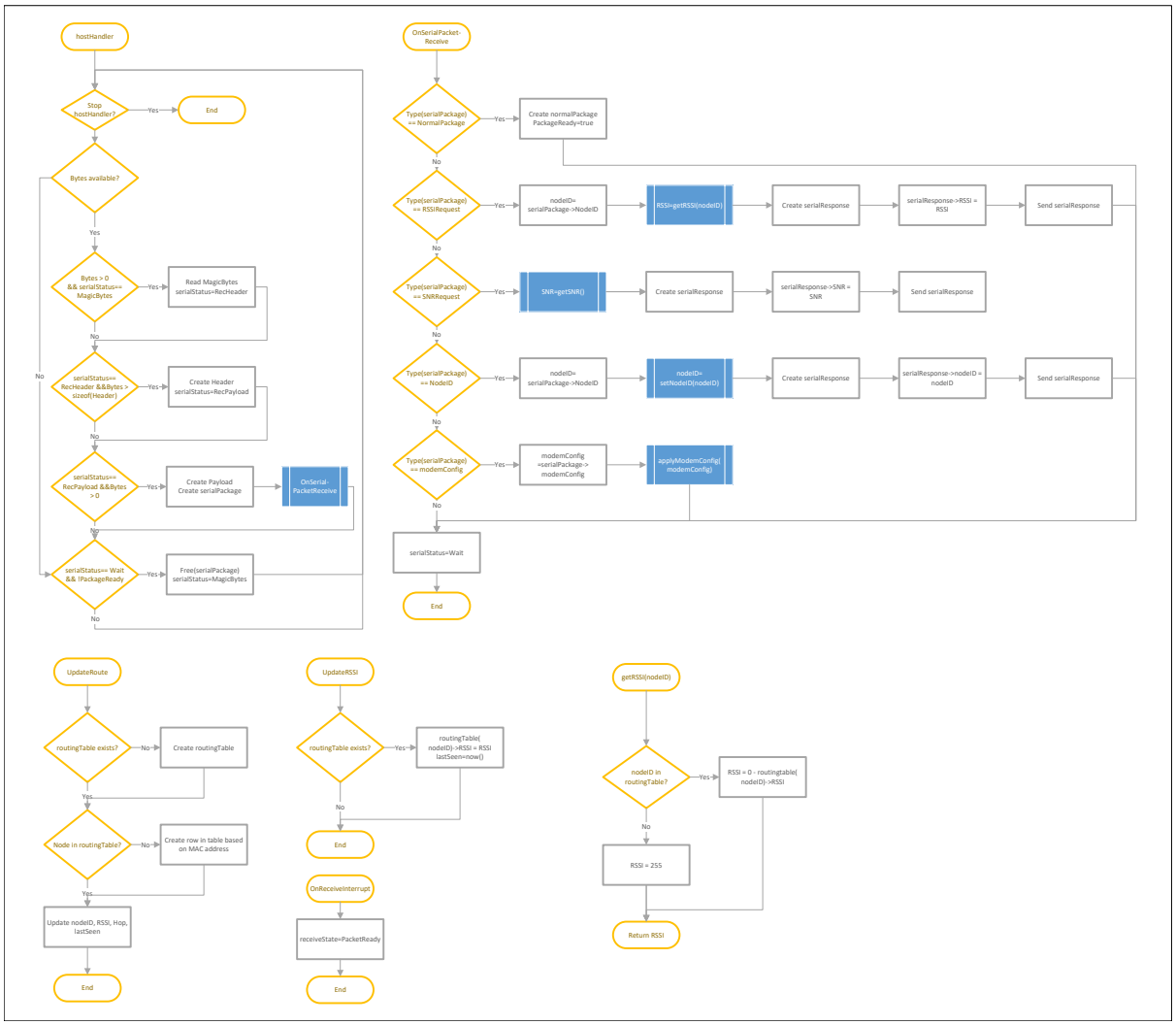


Abbildung 3.2.: Programmablaufplan des „Prozesses 1“ zur Kommunikation mit der Anwendungsschicht via serieller Pakete.

zum Senden an. Sobald der „Prozess 2“ die Daten verarbeitet hat, wird das Bit zurück auf falsch gesetzt und ermöglicht somit dem „Prozess 1“ neue Daten zu empfangen.

Steuerungspakete hingegen werden direkt von dem „Prozess 1“ verarbeitet. Um den aktuellen Wert des SNR zu ermitteln, wird das entsprechende Register des LoRa-Moduls gelesen und durch ein kurzes serielles Paket an die Anwendung zurückgegeben. Das SNR und der RSSI werden als positive Werte übertragen. Der RSSI wird aus der Routing-Tabelle gelesen und ist pro Knoten in der Tabelle vorhanden. Eine neue Konfiguration des LoRa-Moduls, das auch als Modem bezeichnet wird, wird mittels eines Reinitialisierungsaufrufs übernommen. Dabei können die Bandbreite, die Frequenz, die Präambelänge, CR und SF eingestellt werden. Es ist ebenfalls möglich, eine neue Knotenidentifikationsnummer (Englisch: „Node-Identification-Digit“, kurz Knoten-ID) anzufordern. Diese Nummer wird zunächst initial belegt; beispielsweise mittels des letzten Bytes der „Wiresless Local Area Network Media Access Control“-Adresse (kurz WLAN MAC-Adresse) des jeweiligen Modems.

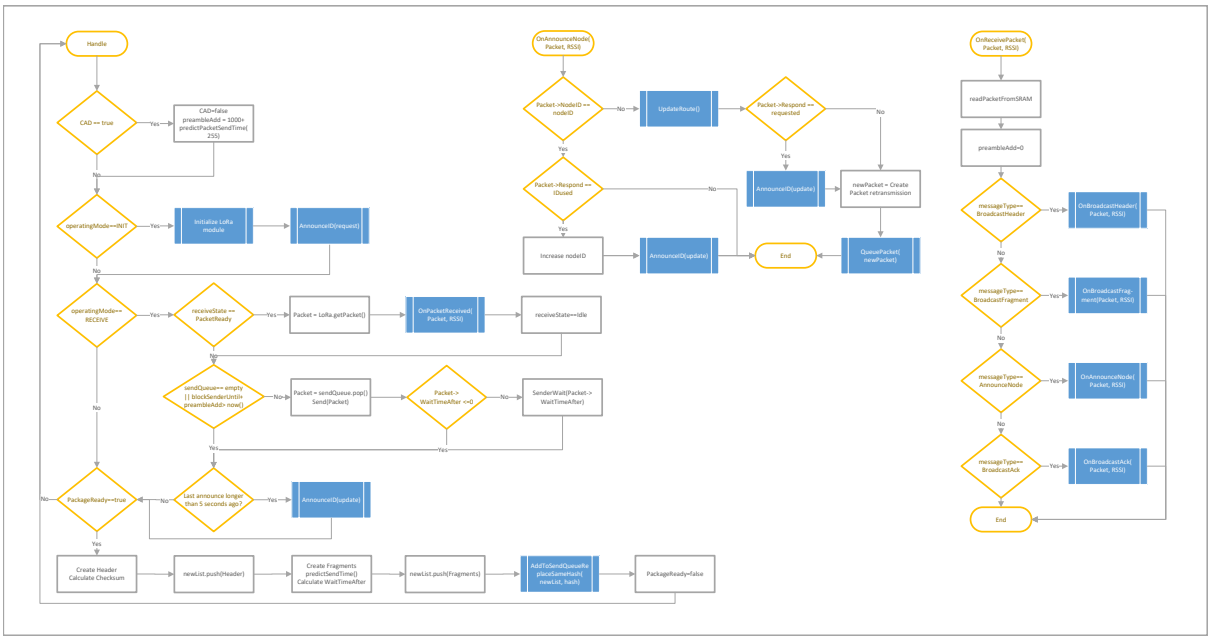


Abbildung 3.3.: Programmablaufplan des „Prozesses 2“ zur Kommunikation innerhalb des ad hoc Netzwerks mittels Long Range .

3.7.2. Kommunikation mit dem LoRa-Modul

Durch das Setzen der Knoten-ID auf einen neuen Wert wird die Funktion „AnnounceID“ mit dem Parameter „request“ aufgerufen, um ein neues Paket in die Sendewarteschlange einzureihen. „request“ bedeutet, dass alle Netzwerkteilnehmer gefragt werden, ob diese Knoten-ID noch nicht vergeben ist. Falls dies der Fall ist, empfängt der Knoten nur Ankündigungsnachrichten des Typs „update“ von den anderen Knoten, um die anfängliche Routing-Tabelle aufzubauen. Wird die Knoten-ID hingegen bereits verwendet, empfängt der anfragende Knoten eine Nachricht des Typs „usedID“, woraufhin dieser seine Knoten-ID inkrementiert und den Ankündigungsprozess neu startet, bis eine gültige Knoten-ID gefunden wird. Ein Überlauf ist kein Problem, da nur eine eindeutige Zahl gefunden werden muss.
Die Hauptfunktion des „Prozesses 2“ ist die Funktion „Handle“, die eine Endlosschleife durchläuft. Empfängt das Modem eine Präambel, wird das Informationsbit „CAD“ auf wahr gesetzt. Dies zeigt eine Übertragung an und veranlasst andere Sender, die die Präambel empfangen haben, mit ihrer Übertragung zu warten. Die Blockierungszeit der Sender wird um die „preambleAdd“-Zeit erhöht. Die „preambleAdd“-Zeit innerhalb der Blockierungszeit in Millisekunden ist allgemein hoch im Vergleich zur Übertragungszeit eines einzelnen Pakets, wird allerdings wieder aufgehoben, falls die Nachricht empfangen wird. Nach der Überprüfung auf eine mögliche Übertragung wird der aktuelle Zustand des Modems ausgewertet. Sofern es nicht initialisiert ist, wird das Modem mit Standardwerten konfiguriert und die initiale Knoten-ID angekündigt. Andernfalls befindet sich das System im Zustand „RECEIVE“.
Ein Unterbrechungsindikatorbit (Englisch: „Interrupt Flag“), mit „receiveState“ bezeichnet, entscheidet, ob ein empfangenes Paket durch Aufruf der Funktion „OnPacketReceive“ und Kopieren der Daten aus dem SRAM behandelt werden muss.

Anschließend veranlasst der Algorithmus das Senden eines Pakets aus der Paketwarteschlange „sendQueue“, sofern diese nicht leer ist und die Blockierungszeit des Modems abgelaufen ist. Dies gilt, falls die Blockierungszeit kleiner als die aktuelle Systemzeit ist. Die Zeiten werden stets als Zeitstempel in Millisekunden interpretiert. Diese Blockierungszeit besteht aus der Zeit, definiert als „blockSenderUntil“, und der zuvor erwähnten „preambleAdd“-Zeit, die nach dem Empfang einer Nachricht auf Null zurückgesetzt wird. Die letzten beiden möglichen Aktionen pro „Handle“-Zyklus sind die Bekanntgabe der Knoten-ID nach einem bestimmten Zeitintervall und die Verarbeitung eines empfangenen seriellen Datenpakets aus dem SRAM. Ersteres dient als „Lebenszeichen“-Nachricht an alle Netzwerkteilnehmer, sodass diese wissen, dass der Knoten noch aktiv ist, auch wenn er selber keine Daten zu senden hat. Die Verarbeitung der seriellen Datenpakete aus dem SRAM wird durch das Informationsbit „PackageReady“ gestartet, das vom „Prozess 1“ gesetzt wird und als nicht blockierende Interprozesskommunikation agiert. Aufgrund der klaren Lese- und Schreibreihenfolge treten keine Zugriffskonflikte (Englisch: „Race Conditions“) oder Artefakte auf.

Die Verarbeitung beginnt mit der Erstellung des Paketkopfes und der Berechnung der Prüfsumme. Der Kopf wird zu einer Liste hinzugefügt, anschließend werden die Datenfragmentpakete berechnet und mit Daten gefüllt. Nach dem Hinzufügen der Fragmente zu derselben Liste, die nun die gesamte Nachricht vollständig beschreibt, wird die Liste der Sendewarteschlange hinzugefügt. Der Einfügungsvorgang durchsucht die Paketköpfe (Englisch: „Header“) in der Warteschlange und vergleicht ihre von der Anwendung angegebenen Streuwerte (Englisch: „Hash“), die einer anwendungsspezifischen Typenklassifizierung entsprechen (siehe Abschnitt 4.2.13). Falls sie übereinstimmen, wird die mit dem Paketkopf verbundene Nachricht durch die neue Nachricht in Form der Paketliste in der Warteschlange „sendQueue“ ersetzt. Der ausschließliche Vergleich der Kopfzeilen reduziert die Anzahl der Vergleiche in der Warteschlange von der Paketanzahl zur Nachrichtenanzahl. Diese liegen immer am Beginn einer Nachricht. Ferner werden die Paketköpfe zuerst gesendet und aus der Warteschlange entfernt, wodurch unnötige Vergleiche vermieden werden. Folglich werden alte noch nicht übertragene Nachrichten durch aktuellere Nachrichten desselben Typs ersetzt; jedoch werden die Datenfragmentpakete einer laufenden Übertragung auch nicht mehr überschrieben, wodurch die Korrektheit der Daten gewährleistet wird.

3.7.2.1. OnReceivePacket

Die Funktion „OnReceivePacket“ liest die Daten aus dem SRAM und setzt „preambleAdd“ auf Null, da Daten empfangen worden sind. Die Auswertung der Daten beginnt mit dem ersten Byte, das den Pakettypen definiert. Derzeit existieren vier verschiedene Typen: „AnnounceNode“ (siehe Abbildung 3.5), „BroadcastHeader“ (siehe Abbildung 3.6), „BroadcastFragment“ (siehe Abbildung 3.7) und „BroadcastAck“(siehe Abbildung 3.8). Jeder Typ wird innerhalb einer eigenen Rücksprungfunktion behandelt.

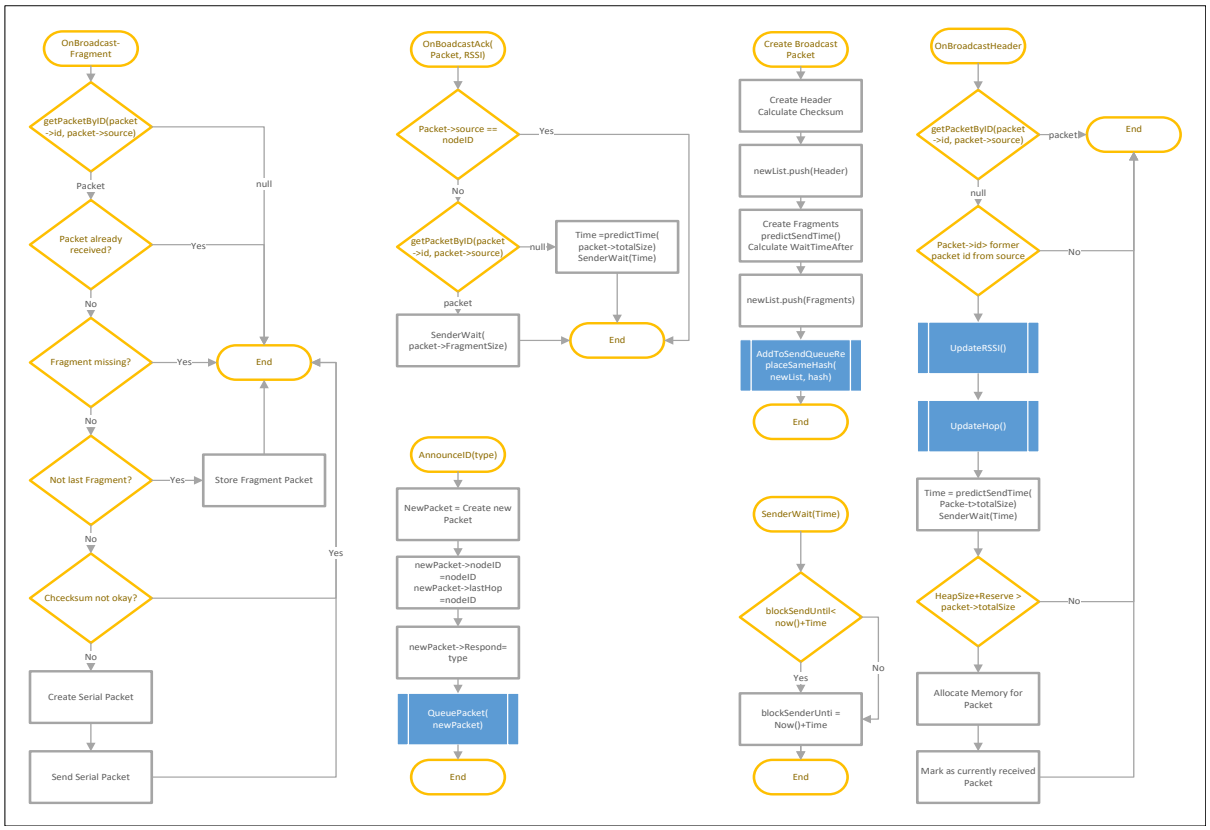


Abbildung 3.4.: Programmablaufplan der Unterfunktionen des „Prozesses 2“.

3.7.2.2. OnAnnounceNode

Zunächst wertet die *OnAnnounceNode*-Rücksprungfunktion den Absender der Nachricht aus. Wenn er sich von der eigenen Knoten-ID unterscheidet, wird die Routing-Tabelle aktualisiert und falls eine Antwort angefordert wird, wird die eigene Knoten-ID gesendet. Bei identischen IDs muss hingegen der Antwortcode überprüft werden. *IDused* zeigt an, dass eine verwendete Knoten-ID ausgewählt worden ist, sodass die Knoten-ID des Moduls inkrementiert wird und der Ankündigungsprozess erneut beginnt, bis die Konvergenz erreicht ist.

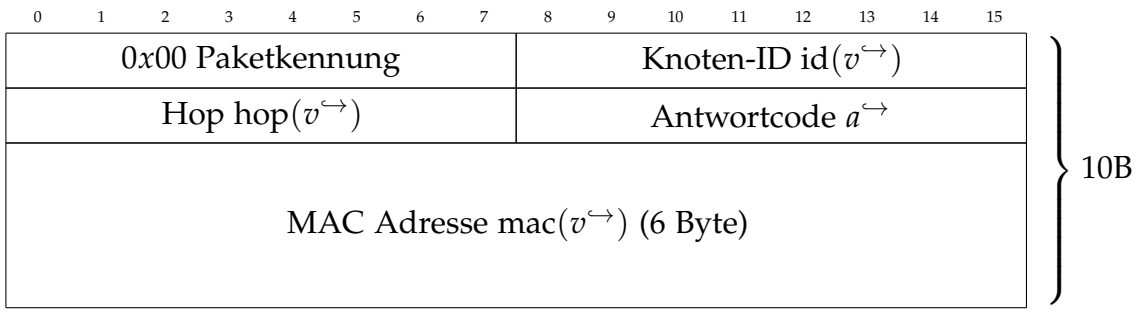


Abbildung 3.5.: Nachrichtenpaket: Knotenankündigung.

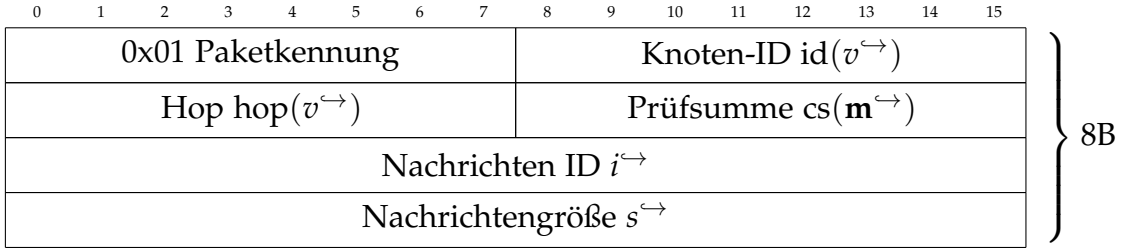


Abbildung 3.6.: Nachrichtenpaket: Nachbarschaftsrundfunknachrichtenankündigung.

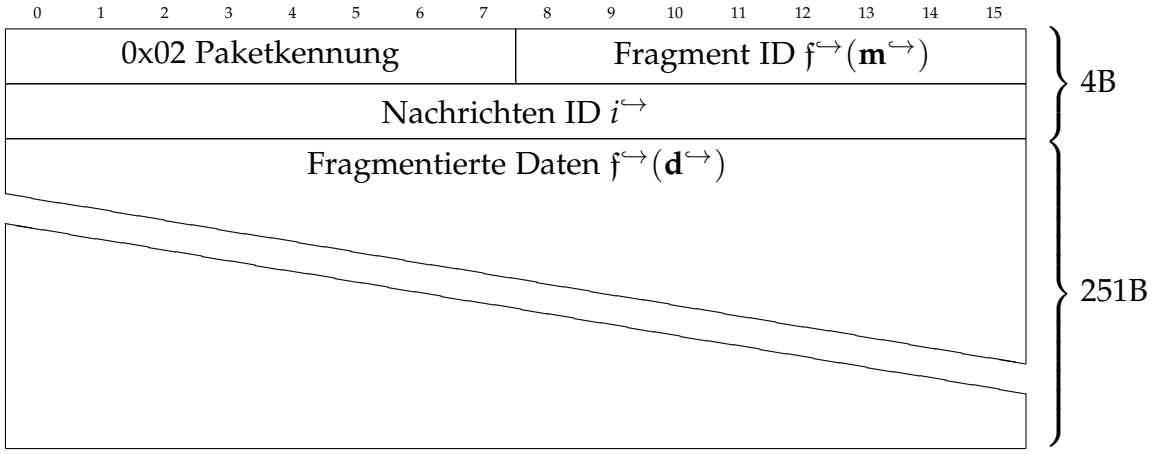


Abbildung 3.7.: Nachrichtenpaket: Nachbarschaftsrundfunkdatenfragment.

3.7.2.3. OnBroadcastHeader

Die Rücksprungfunktion für „OnBroadcastHeader“ verarbeitet die Nachricht und prüft, ob eine Nachricht mit der aktuellen ID bereits existiert. Dies bedeutet, dass die Nachricht bereits zuvor empfangen worden ist. Darüber hinaus können nur die Daten eines Moduls gleichzeitig empfangen werden. Daraus folgt, dass, falls ein anderer Paketkopf bereits empfangen worden ist, der diesem zugewiesene Speicher neu zugewiesen wird und nur die Daten des aktuellen Knotens verarbeitet werden. Andere Pakete werden verworfen. Wenn alle Prüfungen absolviert werden, werden der RSSI und die Hop-Informationen aktualisiert und der erforderliche Speicher zugewiesen. Zudem wird die Übertragungszeit für die gesamte Nachricht geschätzt und der eigene Sender für diese Dauer geblockt.

3.7.2.4. OnBroadcastFragment

Die Funktion *OnBroadcastFragment* prüft, ähnlich wie der bei der Verarbeitung des Paketkopfes, die Nachrichten, um Duplikate, Datenverluste und den Umgang mit Daten von verschiedenen Absendern zu verhindern (siehe Abbildung 3.4). Wird das letzte Fragment der Nachricht empfangen, wird die Prüfsumme cs($\mathbf{m}^{\hookrightarrow}$) der gesamten Daten geprüft. Anschließend werden die Daten als serielles Paket an den Host gesendet und die Blockierung der Sendeprozesses aufgehoben.

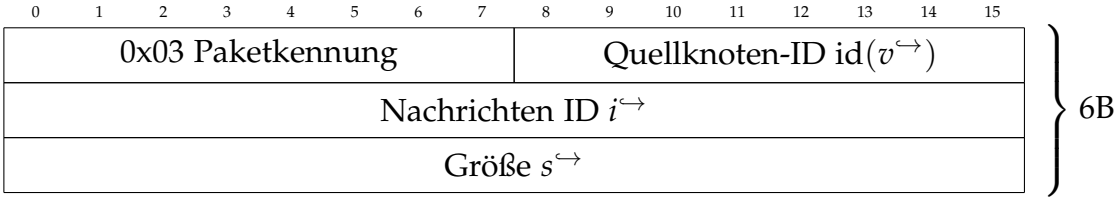


Abbildung 3.8.: Nachrichtenpaket: „Broadcast"-Bestätigung.

3.7.2.5. OnBroadcastAck

Das letzte Paket, das von *OnBroadcastAck* behandelt wird, versetzt alle Knoten, die nicht der Sender sind, in den Empfangsmodus, um Kollisionen auf dem Medium bei langen Paketübertragungen zu vermeiden.
Als Schlussfolgerung wird der „Carrier Sense Media Access / Collision Avoidance"-Ansatz (kurz CSMA / CA) implementiert. Dies wird durch die Vorhersage der Übertragungszeit für Pakete sowie exponentielle zufällige „Backoff"-Zeiten erreicht. Die wichtigsten Konzepte sind in Abbildung 3.2, 3.3 und 3.4 visualisiert.

3.8. Experimente und Analysen

Der entworfene Netzwerkprotokollstapel wird durch eine Experimentalreihe in zwei verschiedenen Gebieten analysiert, wobei simulierte Agenten auf dezentralen Hosts laufen. Jeder dieser Hosts ist mit einem LoRa-Modul für die Kommunikation zwischen den Simulationsinstanzen ausgestattet. Es wird das großstädtische Gebiet im Umfeld der Technischen Universität Dortmund und eine ländliche Region in der mittelgroßen Stadt Herten gewählt, um verschiedene Szenarien zu testen, z. B. Kommunikation innerhalb von Gebäuden und das Reflexionsverhalten an Wänden. Zusätzlich wird die Verlustleistung durch Vegetation in Wäldern und auf Feldern untersucht.

3.8.1. Aufbau

Als Testmodul wird das HelTec Board[1] mit ESP32[2] Chip und LoRa SX1276[3] Chip ausgewählt, da es sich um ein kostengünstiges und einfach zu beschaffendes Modul für Roboter handelt, das die Zweikernanforderungen erfüllt (siehe Abbildung 3.9). Es ist mit einer kleinen 0,5 dBi-Antenne ausgestattet und bietet eine maximale Sendeleistung von 20 dB mW. Die eingebaute Anzeige zeigt die Routingtabelle und zusätzliche Zeitdaten an. Die vier im Netzwerk verwendeten Module sind wie folgt konfiguriert:

$$\begin{aligned} f^{\leftrightsquigarrow} = 868\,\text{MHz}, \Delta f_c^{\leftrightsquigarrow} = 250\,\text{kHz} \\ SF = 7, CR = 4/5, \text{Präambellänge} = 12 \end{aligned} \tag{3.8.1}$$

[1]Chengdu Heltec Automation Technology Co., Ltd. https://heltec.org/product/wifi-lora-32-v2/
[2]ESPRESSIF SYSTEMS (SHANGHAI) CO., LTD. https://www.espressif.com/en/products/socs/esp32
[3]Semtech Cooperation https://www.semtech.com/products/wireless-rf/lora-core/sx1276

Abbildung 3.9.: Heltec „WiFi LoRa 32 (V2.1)“. Es zeigt die Routingtablle mit Informationen zu den anderen Knoten mit den IDs 168, 48, und 72. Diese entsprechen dem letzten Byte der MAC-Adresse. Die Spalten zeigen die IDs, Hops um die ID zu erreichen, die Zeit des letzten empfangen Pakets dieses Knotens und den RSSI zu dem Hop.

Aufgrund der benötigten hohen Übertragungsrate und der Verwendung dieser Technologie im Grenzbereich der Übertragungsleistung kann keine Konfiguration mit hohem Spreizfaktor SF oder geringer Kodierrate CR gewählt werden. Ebenfalls ist die Übertragungsfrequenz innerhalb Europas auf 868 MHz festgelegt. Diese Konfiguration ermöglicht eine theoretische Übertragungsrate von $10\,938\,\mathrm{bit\,s^{-1}}$ und eine maximale Empfängerempfindlichkeit von −124,5 dB mW bei einem niedrigen Rauschpegel von 3 dB mW. Jede Nachricht hat einen festen Paketkopf von 20 bit, der durch den LoRa-Standard definiert ist, und die Präambel von 12 Zirps. Diese werden immer mit $CR = \frac{4}{8}$ gesendet. Sie lassen sich als zusätzliche zu sendende Bytes pro Nachricht umwandeln:

$$12 \text{ Präambellänge: } 12.25 T_s \cdot SF \approx 86\,\mathrm{bit} \tag{3.8.2}$$

$$\frac{(86\,\mathrm{bit} + 20\,\mathrm{bit})}{(\rho^{\backsim})' \cdot \frac{4}{8}} \cdot ((\rho^{\backsim})' \cdot \frac{4}{5}) \approx 170\,\mathrm{bit} = 21{,}25\,\mathrm{B} \tag{3.8.3}$$

Die 21,25 B reduzieren die Nutzdatenrate für kompakte Nachrichten wie „Triangle Line Update“ or „Alive“ deutlich. Alle Module werden in einer Höhe von 1,5 m über dem Boden betrieben. Für das ländliche Szenario wird der erste Sender in einem Wald (siehe Abbildung 3.10) platziert und die erreichbaren Reichweiten und Bandbreiten im Wald, am Waldrand, auf verschiedenen Gras- und Maisfeldern und auf zwei Bauernhöfen gemessen. Die entsprechenden Markierungen und Entfernungen sind in Abbildung 3.10 dargestellt.
Für das großstädtische Szenario wird ein Sender in einem Büro und ein Sender in einem 30 m entfernten Raum an einem Fenster an der Ecke des Gebäudes im zweiten Stock angebracht. Das Büro und die Sender sind in Abbildung 3.11 mit einem blauen Rechteck und zwei blauen Piktogramme markiert. Das städtische Szenario wird für verschiedene Lastszenarien verwendet, die über jeweils fünf Messungen gemittelt werden. Die Dauer einer Messung beträgt immer 4 min und beginnt mit dem Neustart des Moduls, sodass immer eine Einwahl ins Netz erforderlich ist. Außerdem werden dadurch zufällige natürliche Effekte gemittelt.

3.8.2. Reichweitentests

Die ersten Experimente werden im Hinblick auf den maximalen Abstand zwischen den Knoten in zwei verschiedenen Gebieten analysiert, die in den Kartenausschnit-

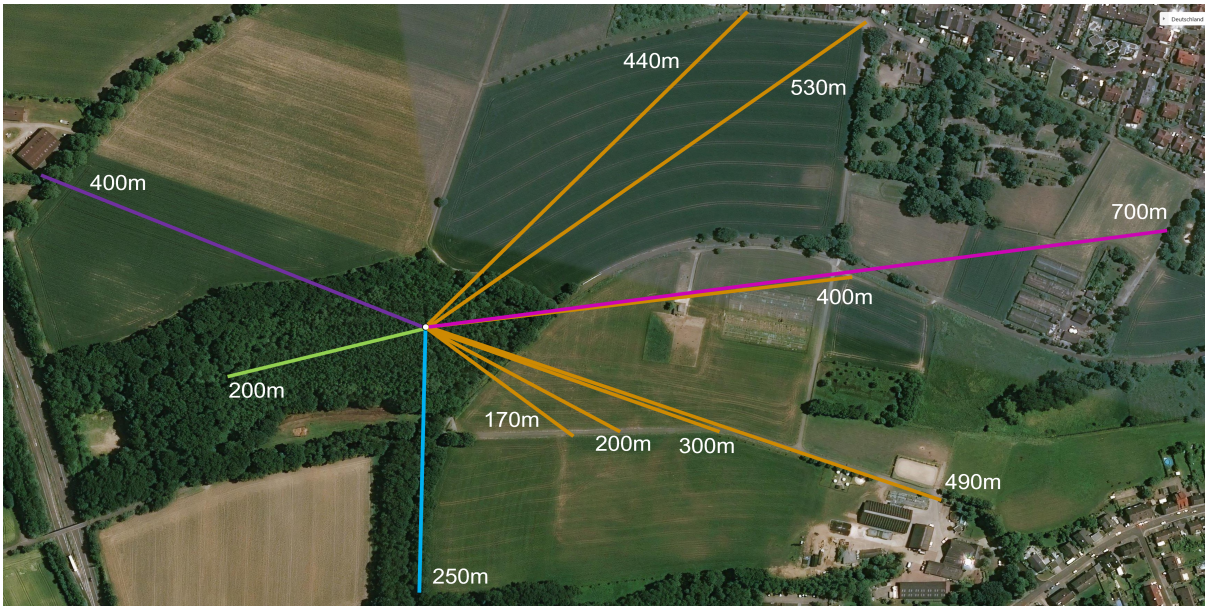


Abbildung 3.10.: Beschreibung der Messpunkte in einer ländlichen Gegend in Herten (51°36’40,7" Nord, 7°09’26,2" Ost), einer mittelgroßen Stadt. Die Messpunkte werden mit ihren Entfernungen zum Knoten in der Mitte des Waldes dargestellt.

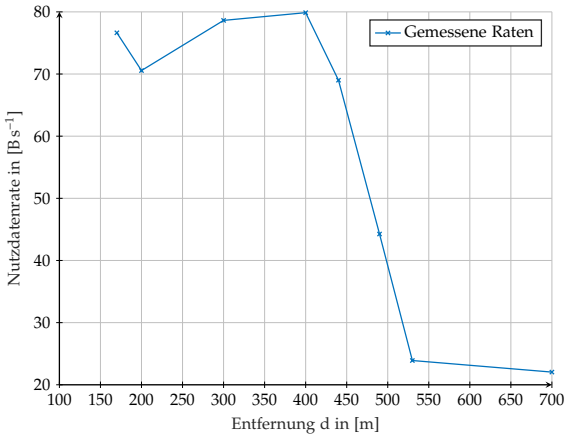
Abbildung 3.11.: Beschreibung der Messpunkte in der Großstadt Dortmund (51°29’22,5" Nord, 7°24’18,2" Ost). Die Messpunkte sind mit Abständen zum Knoten, der am Fenster des blau markierten Gebäudes montiert ist, bemaßt. Der zweite Knoten innerhalb des Gebäudes ist 30 m vom ersten entfernt, allerdings auf der gleichen Etage. LSR (Englisch: „Last Signal Received“) markiert die Position des letzten Signalempfangs. An diesem Punkt können kleine Netzwerkerhaltungsnachrichten empfangen werden, ein konstanter Datenaustausch ist nicht möglich.

Tabelle 3.5.: Übertragungsraten im ländlichen Raum (vgl. Abbildung 3.10).

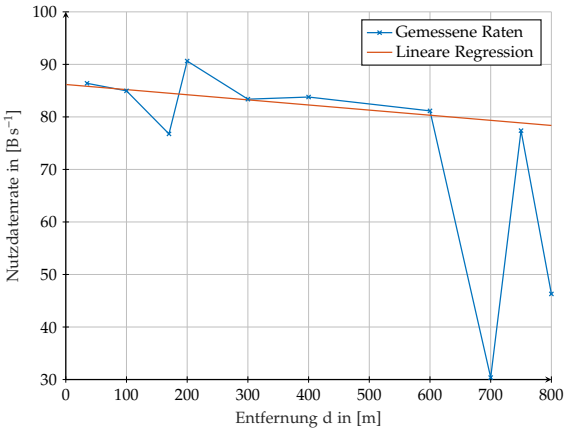
Gebiet	Entfernung	Nutzdatenrate $\rho^{\rightsquigarrow}$	RSSI
Übertragung innerhalb des Waldes (grün, cyan, lila)			
Wald	200 m	6,83 B s^{-1}	−122 dB mW
Waldrand	250 m	17,71 B s^{-1}	−122 dB mW
Gehöft	400 m	24,61 B s^{-1}	−123 dB mW
Übertragung aus dem Wald aufs Feld (orange, rot)			
Grasfeld	170 m	76,63 B s^{-1}	−116 dB mW
Grasfeld	200 m	70,56 B s^{-1}	−112 dB mW
Grasfeld	300 m	78,63 B s^{-1}	−121 dB mW
Grasfeld	400 m	79,87 B s^{-1}	−119 dB mW
Maisfeld	440 m	69,00 B s^{-1}	−122 dB mW
Bauernhof	490 m	44,26 B s^{-1}	−123 dB mW
Maisfeld	530 m	23,90 B s^{-1}	−121 dB mW
Gewächshaus	700 m	22,04 B s^{-1}	−123 dB mW

ten in Abbildung 3.11 und 3.10 visualisiert werden. Tabelle 3.5 hebt die beträchtliche Verlustleistung aufgrund der Flora hervor, die nicht nur niedrige RSSI-Werte, sondern auch viele Paketverluste verursacht. Dies zeigt sich in einer im Vergleich zu anderen Orten sehr geringen empfangenen Nutzdatenrate. Die Nutzdatenrate wird auf der Grundlage erfolgreicher Datenpaketübertragungen berechnet. Netzwerkbezogene Nachrichten und Pakete, z. B. Knoten-ID-Ankündigungen und Paketköpfe sowie Präambeln, werden nicht berücksichtigt. Weiterhin verdeutlicht Abbildung 3.12c, dass die Vegetation einen deutlich größeren Einfluss auf die Nutzdatenrate hat, denn trotz steigender Entfernung steigt die Nutzdatenrate bei abnehmender Vegetation. Dieser Zusammenhang wird auch in Tabelle 3.5 und Abbildung 3.12a bestätigt. Dort gilt, dass bei allgemein geringer Vegetation bis zu 79,87 B s^{-1} erreicht werden können. Blockiert ein Maisfeld den Sendebereich, sinkt die Rate hingegen wieder deutlich. Folglich lässt sich in ländlichen Gebieten mit geringer Vegetation eine maximale Knotenentfernung von 700 m erzielen. Im Vergleich zum städtischen Szenario, das in Abbildung 3.11 visualisiert und in Abbildung 3.12b dargestellt ist, sinken die erreichten Nutzdatenraten mit der Entfernung nichtlinear und stark ab.

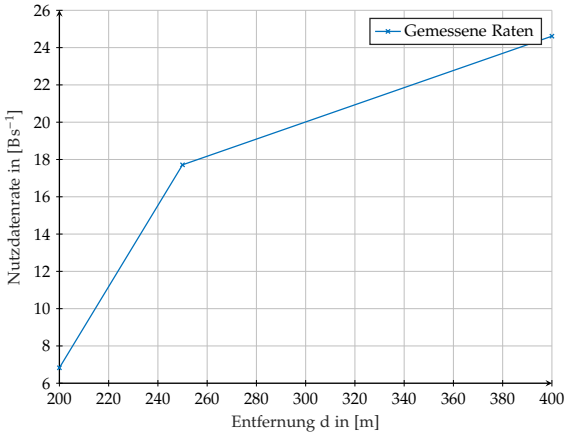
Im Gegensatz dazu zeigt die großstädtische Umgebung eine lineare Abhängigkeit zwischen der Rate und der Entfernung, sofern Ausreißer ausgeschlossen werden. Diese stark abweichenden Raten resultieren aus Interferenzen und Abschirmungen in städtischen Gebieten. In dem Testgebiet der Universität sind sogar Übertragungen in Gebäude hinein über 800 m möglich. Dies gilt auch dann, wenn keine direkte Sichtverbindung besteht, verursacht durch Signalreflexionen an ebenen Gebäudewänden, die in der ländlichen Umgebung nicht vorhanden sind. Dies wird durch ein Experiment bestätigt, bei dem das Signal durch ein großes Hindernis blockiert wird, z. B. durch das Gebäude in Abbildung 3.11. Trotz der fehlenden Sichtverbindung sind von einem Büro aus noch Reichweiten von 400 m möglich.



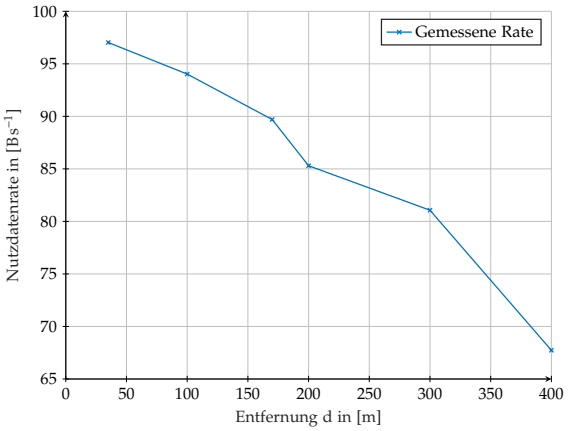
(a) Veranschaulichung der Nutzdatenrate in Abhängigkeit zur Entfernung in ländlichen Raum. Der Sender ist innerhalb des Waldes und die Messepunkte sind an Gras- und Maisfeldern (siehe Abbildung 3.10).



(b) Veranschaulichung der Nutzdatenrate in Abhängigkeit zur Entfernung in städtischem Raum. Abschirmungs- und Reflexionseinflüsse auf den ansonsten linearen Zusammenhang sind bei 170 m, 210 m 700 m, and 800 m zu erkennen.



(c) Veranschaulichung der Nutzdatenrate in Abhängigkeit zur Entfernung in Waldgebieten (siehe Abbildung 3.10): Im Wald (200 m, grüne Strecke), am Waldrand (250 m, cyane Strecke) und auf dem Feld hinter dem Wald (400 m, lila Strecke).



(d) Veranschaulichung der Nutzdatenrate in Abhängigkeit zur Entfernung für zwei Sender in der städtischen Umgebung. Die initiale Nutzdatenrate ist höher als für einen Sender, jedoch fällt die Rate stets linear deutlich stärker ab.

Abbildung 3.12.: Abstandsnutzdatenverhältnis.

Tabelle 3.6.: Nutzdatenrate im städtischen Raum (siehe Abbildung 3.11).

Senderanzahl	Entfernung	Nutzdatenrate $\rho^{\rightsquigarrow}$	RSSI in dB mW
1	35 m	86,40 B s^{-1}	−96
1	100 m	84,96 B s^{-1}	−86
1	170 m	76,77 B s^{-1}	−105
1	200 m	90,63 B s^{-1}	−105
1	300 m	83,38 B s^{-1}	−113
1	400 m	83,78 B s^{-1}	−110
1	600 m	69,05 B s^{-1}	−119
1	700 m	30,38 B s^{-1}	−123
1	750 m	77,39 B s^{-1}	−121
1	800 m	46,30 B s^{-1}	−121
2	35 m	97,03 B s^{-1}	−87, −101
2	100 m	94,03 B s^{-1}	−94, −105
2	170 m	89,70 B s^{-1}	−99, −111
2	200 m	85,30 B s^{-1}	−103, −112
2	300 m	81,07 B s^{-1}	−104, −118
2	400 m	67,73 B s^{-1}	−117, −123
3	35 m	93,94 B s^{-1}	−48, −51, −105
4	4 m	66,42 B s^{-1}	−48, −49, −51, −53

3.8.3. Analyse der Nutzdatenrate

Der zweite zu untersuchende Aspekt ist die steigende Zahl an Sendern in einem Knotenpunktnetzwerk (siehe Tabelle 3.6 und Abbildung 3.12d). Die Nutzdatenrate wird von einem passiven Knoten gemessen, der netzwerkbezogene Informationen, aber keine Datenpakete sendet mit Ausnahme des Aufbaus mit vier Sendern. Dies entspricht der maximalen Anzahl von Modulen, die für Experimente zur Verfügung stehen. Während dieser Tests ist sichergestellt, dass alle Knoten die Signale von allen anderen Knoten empfangen, um das Netzwerk möglichst stark zu belasten. Allerdings können Übertragungsfehler und widersprüchliche Übertragungen auftreten. Die Tabelle 3.6 verdeutlicht, dass die maximale Nutzdatenrate knapp unter 100 B s^{-1} liegt. Zudem sinkt die Datenrate nicht direkt mit zunehmender Senderanzahl. Der Anstieg der Nutzdatenrate bei zwei Sendern kann sowohl zufällig als auch ein Hinweis auf eine nun ausgeschöpfte Restkapazität des Netzwerks sein. Weiterhin ist aus den Daten ersichtlich, dass die Datenraten bei gleicher Entfernung bei drei Sendern nicht deutlich reduziert sind. Dies ist ein Indiz für eine funktionierende Umsetzung des CMSA/CA-Ansatzes, der Paketkollisionen reduziert und die Übertragungen koordiniert. Die signifikante Reduktion der Nutzdatenrate bei vier Sendern resultiert daraus, dass der Empfänger bei dieser Messung auch als Sender fungiert. Da sich die Module nur entweder im Empfangs- oder im Sendezustand befinden können, ist die Zeit, in der das Modul empfangsbereit ist, geringer als zuvor. Wird von einer gleichverteilten

Nutzung der Kapazität ausgegangen, ergibt sich eine Datenrate von:

$$\frac{4}{3} \cdot 66{,}42\,\mathrm{B\,s^{-1}} = 88{,}56\,\mathrm{B\,s^{-1}} \tag{3.8.4}$$

Allerdings sinkt die Datenrate mit zunehmender Entfernung sehr viel stärker als zuvor (siehe Abbildung 3.12d), weil der RSSI zum zweiten Sender sehr viel niedriger ist. Das führt zu häufigeren Übertragungsfehlern. Wenn die Datenraten für zwei Sender in 400 m Entfernung auf einer Kombination von Raten bei vergleichbaren RSSI-Werten für einen Sender basieren würde, wäre die resultierende Nutzdatenrate:

$$\frac{1}{2} \cdot 69{,}05\,\mathrm{B\,s^{-1}} + \frac{1}{2} \cdot 30{,}38\,\mathrm{B\,s^{-1}} \approx 45{,}22\,\mathrm{B\,s^{-1}} \tag{3.8.5}$$

Die geringeren RSSI-Werte resultieren aus dem Standort des zweiten Senders innerhalb eines geschlossenen Raumes in einer Ecke des Gebäudes, der nicht an einem Fenster montiert ist und keine Sichtverbindung bietet. Somit wird das Netzwerk stark durch inhomogene Signalstärken gestört. Die Nutzlastrate pro Sendemodul wird hingegen reduziert und kann maximal wie folgt durch die Senderanzahl $\#s^{\rightsquigarrow}$ ausgedrückt werden:

$$\frac{97{,}03\,\mathrm{B\,s^{-1}}}{\#s^{\rightsquigarrow}} \tag{3.8.6}$$

Um nun eine prädiktive Schätzung der Nutzdatenraten für die Verwendung während der Missionen zu bestimmen wird, ein empirisches Modell auf Grundlage der Abbildung 3.13 gebildet:

$$\rho^{\check{\rightsquigarrow}}(d) = \tag{3.8.7}$$

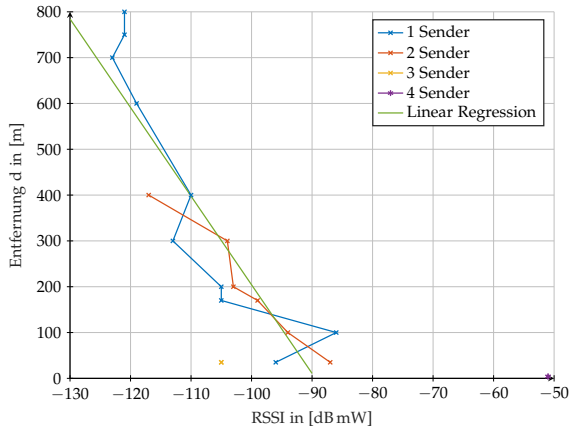
$$\rho^{\check{\rightsquigarrow}}(P_{r^{\rightsquigarrow}}(d)) = \begin{cases} 91{,}2629\,\mathrm{B\,s^{-1}}, \text{falls } d \leq 0\,\mathrm{m} \\ -0{,}024\,59\,(\mathrm{B/s})/(\mathrm{m}) \cdot d + 91{,}2629\,\mathrm{B\,s^{-1}}, \text{falls } 0\,\mathrm{m} \geq d \geq 750\,\mathrm{m} \\ 0\,\mathrm{B\,s^{-1}}, \text{sonst} \end{cases}$$

$$\rho^{\check{\rightsquigarrow}}{}_s(\#s^{\rightsquigarrow}, d) = \frac{\rho^{\check{\rightsquigarrow}}(d)}{\#s^{\rightsquigarrow}} \tag{3.8.8}$$

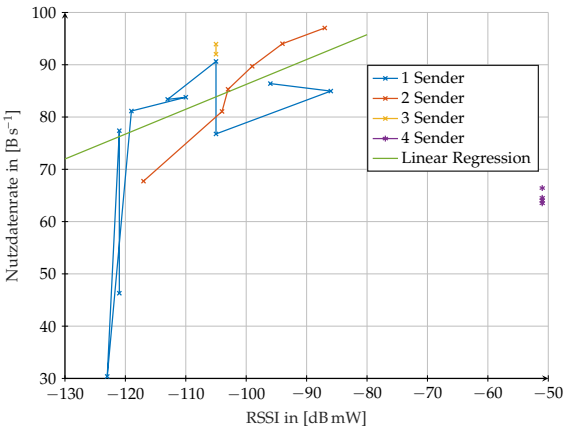
3.8.4. Bewertung der Nutzbarkeit

Abschließend werden die erreichten Raten mit der geforderten Nutzdatenrate verglichen und die Implikationen anhand des Beispiels 3.8.1 analysiert. Aus Abschnitt 3.6 ergibt sich, dass bei normalen Nachrichten eine Übertragungsrate von mehr als $4894\,\mathrm{B\,s^{-1}}$ erforderlich ist. Dies ist mit dem LoRa-Mesh-Netzwerk definitiv nicht möglich. Daher wird die Nutzbarkeit anhand eines typischen Einsatzszenarios evaluiert.

Beispiel 3.8.1: *Drei Roboter werden für 15 min in einem Einsatzgebiet betrieben und erhalten zwei Aufträge: Einer soll das Gebiet erkunden, die anderen eine Linienformation mit einem Abstand von 35 m einhalten. Folglich werden zwei Missionsnachrichten gesendet, eine Formationsnachricht alle 10 s und eine Synchronisationsnachricht alle 10 min.*



(a) Veranschaulichung der möglichen Entfernung in Abhängigkeit zum RSSI. Die Daten können zur empirischen Nutzdatenratevorhersage verwendet werden.



(b) Veranschaulichung der Nutzdatenrate in Abhängigkeit zum RSSI. Die Daten können zur empirischen Nutzdatenratevorhersage verwendet werden.

Abbildung 3.13.: Abstands-RSSI-Nutzdatenverhältnis für verschiedene Senderanzahlen.

$$2 \cdot 28\,\mathrm{B} + 28\,\mathrm{B} \cdot \frac{900\,\mathrm{s}}{10\,\mathrm{s}} + 32\,\mathrm{B} \cdot \frac{6300\,\mathrm{s}}{600\,\mathrm{s}} = 2642\,\mathrm{B} \tag{3.8.9}$$

Nun kann auf Basis der Nutzdatenratenschätzung 3.8.7 mittels des maximalen Interroboterabstandes die Nutzdatenrate $\rho^{\tilde{\rightsquigarrow}}$ bestimmt werden. Anschließend wird diese Übertragungsleistung um die zuvor berechneten Daten reduziert.

$$900\,\mathrm{s} \cdot \rho^{\tilde{\rightsquigarrow}}(70\,\mathrm{m}) - 2642\,\mathrm{B} = 80\,587\,\mathrm{B} - 2642\,\mathrm{B} = 77\,945\,\mathrm{B} \tag{3.8.10}$$

Die verbleibende Kapazität steht für die Roboter-Kurznachrichten zur Verfügung, wodurch sich eine Aktualisierungsrate von 0,2005 Hz pro Roboter ergibt:

$$\frac{77\,945\,\mathrm{B}}{3 \cdot 144\,\mathrm{B}} \cdot \frac{1}{900\,\mathrm{s}} \approx 0{,}2005\,\mathrm{Hz} \tag{3.8.11}$$

Diese Rate ist akzeptabel. Der ermittelte Planungshorizont des modellprädiktiven Reglers entspricht 30 s und die konfigurierten Roboter fahren höchstens $3\,\mathrm{m\,s^{-1}}$ (siehe Kapitel 6), sodass maximal 14,96 m zurückgelegt werden, bevor die Trajektorie aktualisiert wird. Folglich können bis zu fünf konsekutive Paketverluste pro Roboter auftreten, bevor der Regler mit extrapolierten Werten auf Grundlage veralteter Daten arbeiten muss. Die Aktualität der Daten ist also nicht gut, aber immer noch brauchbar. Es ist aber ebenfalls klar, dass der Einsatz im Wald über 50 m nicht praktikabel ist, wie in Tabelle 3.5 dargelegt wird.

3.9. Zwischenfazit und Ausblick

Dieses Kapitel stellt die Grundlagen der ad hoc Netzwerkkommunikation bezogen auf den Anwendungsfall der autonomen Schwarmrobotik in Katastrophengebieten zur Erkundung oder zur Rettung dar. Um diese Operationen durchführen zu können,

benötigen die Roboter, auch wenn sie autonom sind, ein Minimum an Kommunikation, die über ein ad hoc Netzwerk realisiert werden muss. Dafür werden zwei physikalische Übertragungsmodelle für verschiedene Protokolle aufgestellt. Für den IEEE 802.11-Standard wird ein modellbasierte Ansatz zur Schätzung der Übertragungsrate gewählt. Dieser Standard ermöglicht hohe Übertragungsraten, jedoch nur eine geringe Übertragungsreichweite. Das entwickelte Modell lässt sich dabei für verschiedene Frequenzen und Hardware konfigurieren und berücksichtigt ebenfalls Hindernisse sowie thermisches Rauschen und aktive Störsender in der Berechnung der Übertragungsrate. Dem gegenüber bietet der LoRa-Standard eine kostengünstige Alternative für große Reichweiten. Allerdings mit deutlich geringeren Nutzdatenraten, die im kritischen unteren Bereich für eine sinnvolle Verwendung liegen. Die Experimente zeigen, dass die Nutzlast mittels LoRa, die unter realistischen Bedingungen übertragen werden kann, nur weniger als 10 % der möglichen Datenrate von LoRa erreicht. Gründe dafür liegen sowohl in der Wahl der ad hoc Netzwerktopologie, den verwendeten Protokollverfahren als auch in der Implementierung und Hardware. Weiterhin wird gezeigt, dass Umgebungsbedingungen einen großen Einfluss auf die Paketverluste und somit auf die Nutzdatenrate haben. Bei langsamen Robotern, die oft in Katastrophensituationen eingesetzt werden und stark komprimierte und dichte Daten verwenden, ist jedoch ein Vorhersagehorizont von 30 s ausreichend und könnte von den Kommunikationsmodulen erreicht werden. Außerdem ist der erforderliche Abstand zwischen den Robotern im Katastrophenfall oft begrenzt. Die Messungen zeigen jedoch auch, dass diese Netzwerktopologie die LoRa-Hardware im Grenzbereich betreibt.
Es sind jedoch Versuche mit handelsüblicher und günstiger Hardware durchgeführt worden. Etwas teurer sind verbesserte Antennen mit einem viel höheren Antennengewinn. Antennen mit einem 20-fach höheren Antennengewinn, z. B. 10 dBi, würden den RSSI um $2 \cdot 10 = 20\,\mathrm{dB\,mW}$ erhöhen. Dies ermöglicht die Verwendung des Protokollstapels auch in Situationen, in denen die Umgebungsbedingungen wesentlich problematischer sind, zum Beispiel bei starker Vegetation, Abschirmung oder Staub und Rauchentwicklung. Schließlich werden die Daten aus den Messungen verwendet, um realistischere Modelle für die Datenübertragung in einem Simulator zu entwickeln. Dies wird die Simulationsergebnisse verbessern und auch die Möglichkeit bieten, Steuerungssoftware in simulierten Umgebungen zu testen und zu bewerten. Weiterhin sind die Regeln zur zeitlichen Belegung der Funkfrequenzen für einen industriellen Einsatz zu beachten. Diese verbieten die dauerhafte Nutzung einer Funkfrequenz durch einen Sender.

4

Autonomiekern

Im folgenden Kapitel werden die Struktur und die Konzepte des entwickelten Steuerungsprogramms, als „Autonomiekern“ bezeichnet, für die autonomen Agenten beschrieben. Die Ausgangsbasis dieses Programms bietet die in der Abschlussarbeit Puzicha [141] entwickelte Simulation. Mit dieser lassen sich Missionen leicht testen und die Daten der Agenten sowie der Umwelt einfach auslesen, weil alle Komponenten einen Verweis zu dem übergeordneten Szenarioobjekt besitzen. Ebenso besitzt dieses Verweise zu jedem in ihm existierenden Objekt. Diese gegenseitigen Verweise sind für Simulationen üblich und ermöglichen direktes Auslesen und Manipulieren der Zustände und Umwelteinflüsse. In einige Simulationen können auch zuvor aufgezeichnete reale Daten eingespeist werden. Jedoch können diese Ansätze keine Realzeitintegration realer Agenten ermöglichen, denn die realen Agenten können kein Szenarioobjekt besitzen, da dies die gesamte reale Welt wäre. Weiterhin ist auch das Teilen von Hindernisobjekten im Speicher nicht möglich. Daher wird ein von Grund auf neues System entwickelt, welches lediglich einige Darstellungsfunktionen übernimmt.

Der allgemeine Aufbau sowie die untergeordneten Komponenten sind mit dem Ziel einer möglichst hohe Modularität und Kapselung bei gleichzeitiger Laufzeiteffizienz entwickelt worden (siehe Abbildung 4.1). Somit wird die Wiederverwendbarkeit, die Anpassungsfähigkeit und der Export einzelner Komponenten begünstigt, sodass diese sowohl in realen Agenten als auch innerhalb von Simulationen verwendet werden können. Dazu werden der eigentliche Regler sowie die mathematische Lösung der modellprädiktiven Regelung in eine eigenständige Programmbibliothek ausgelagert und als Optimierung bezeichnet (siehe Abbildung 4.1). Der Autonomiekern der Agenten baut auf dieser Bibliothek auf, kann jedoch ebenso alternative Regler und Optimierungsbibliotheken verwenden. Eine implementierte Alternative bietet die „pagmo2“ Optimierungsbibliothek (vgl. [15]), die für Leistungsvergleiche verwendet wird. Das gesamte Programm wird mittels des Bausystems „CMAKE“ für die Programmiersprache C++ aufgesetzt. Dadurch wird die modulare und plattformunabhängige Programmübersetzung ermöglicht, sodass es Microsoft® Windows™ und Ubuntu Linux Betriebssysteme auf X86, X64 und ARM Prozessorarchitekturen unterstützt.

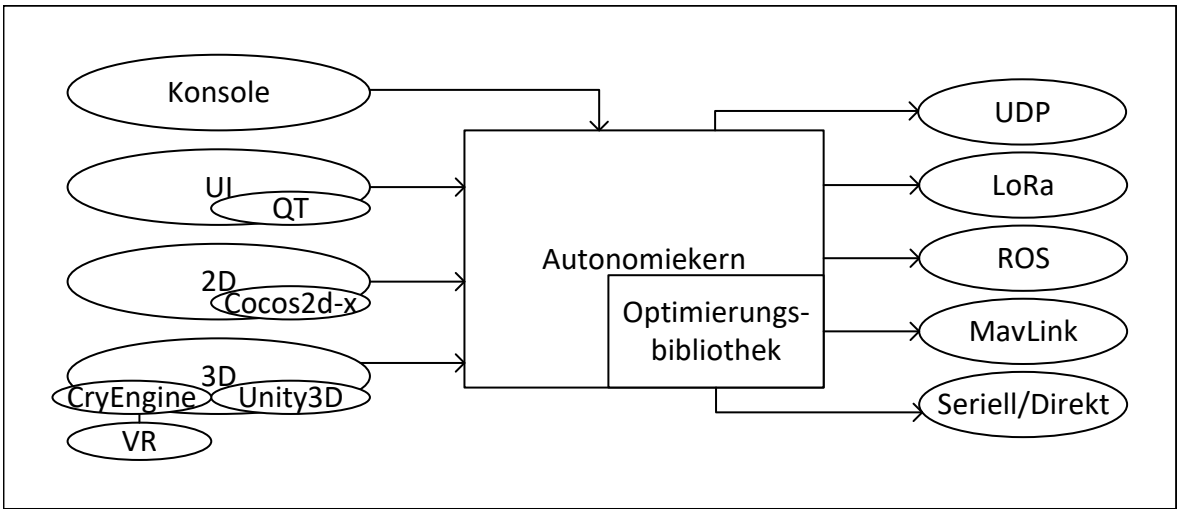


Abbildung 4.1.: Modulare Architektur und Schnittstellenintegration des Autonomiekerns. Die linke Seite zeigt die realisierten Benutzerschnittstellen und die rechte Seite die Kommunikationsmöglichkeiten mit realen Agenten.

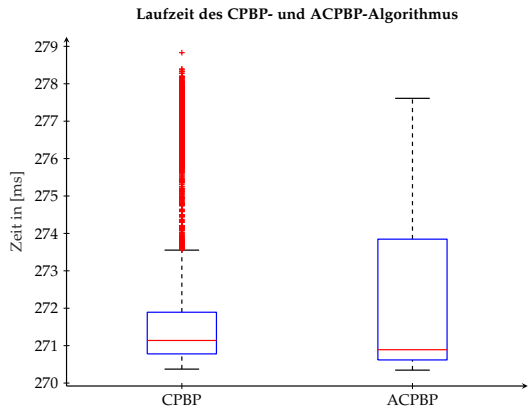


Abbildung 4.2.: Laufzeitvergleich der Reglerimplementierungen (siehe Abschnitt 2.4.3).

4.1. Optimierungsbibliothek

Die Optimierungsbibliothek beinhaltet den zur Veröffentlichung Hämäläinen, Rajamäki und Liu [57] zugehörigen CPBP-Quelltext als auch die erweiterte Version ACPBP, die im Abschnitt 2.4.3 vorgestellt wird. ACPBP ist als C++ Vorlagenklasse (Englisch: „Template“) entwickelt worden, um eine möglichst hohe übersetzerseitige Optimierung zu ermöglichen. Die Vorlage legt die Stellgrößendimension, Zustandsdimension sowie die Anzahl an Irrfahrten und Zeitschritten fest. Dadurch erhalten die meisten Schleifen eine feste Durchlaufanzahl und die Matrixoperationen feste Dimensionen, sodass der Speicher zu Initialisierung angefordert werden kann. Weiterhin kann in der Vorlage der Datentyp und somit die Präzision der Berechnungen bestimmt werden. Das Resultat der Quelltextoptimierung äußert sich in einer gleichen Laufzeit bei deutlich gesteigertem Berechnungsaufwand des Optimierers (siehe Abbildung 4.2).
Die fixierten Dimensionen ermöglichen ebenfalls die Berechnung einiger Matrixopera-

tionen auf der Grafikkarte mittels des CUDA-Grundgerüsts. Die Verwendung der Grafikkarte eignet sich jedoch hauptsächlich nur für große Matrixoperationen, bei denen der Speicher zuvor angefordert worden ist. Ansonsten verzögert die Speicheranforderung und die Kopie der Daten vom Direktzugriffsspeicher des Prozessors zu dem des Grafikprozessors die Berechnung zu lange, sodass in Summe die gesamte Operation länger dauert als ohne Grafikkartenunterstützung. Untersuchungen zur parallelen Optimierung dieser Regelung auf Grafikkarten werden in der Abschlussarbeit Eigenseher [32] begonnen. Die Optimierungsbibliothek besitzt nur eine Abhängigkeit zu „Eigen3“ (vgl. [51]). Diese Bibliothek stellt Funktionen und Datentypen für lineare Algebra zur Verfügung.

4.2. Simulation und Regelungskern

Das Ziel dieser Arbeit ist es, Verhaltensweisen und Lösungsstrategien für autonome Roboterschwärme zu entwickeln und zu analysieren. Dazu können Experimente mit realen Roboterschwärmen durchgeführt werden oder es werden analysezweckgebundene Simulationen als Modell dieser Systeme verwendet. Zur Beschleunigung der Entwicklung durch eine einfachere Fehlereingrenzung und -analyse sowie aufgrund der Kosten für reale Roboterschwärme wird zunächst mit Simulationen gearbeitet. Anschließend erfolgen Anpassungen an die realen Systeme.

4.2.1. Roboter- und Netzwerksimulatoren

Es existieren verschiedene Simulatoren für mobile Roboter, Roboterschwärme (vgl. [82], [116], [19], [88], [134], [135] und VEROSIM) und für Netzwerke, z. B. OMNet++ oder ns-3 (vgl. [151]). Jedoch ist das Problem dieser Simulatoren, dass sie in der Regel Hardware und deren Dynamik simulieren oder zu stark abstrahieren. Das Problem der direkten physikalischen Simulation ist, dass nur eine sehr geringe Anzahl an Agenten zeitgleich simuliert werden kann. Dem gegenüber vereinfachen die stark abstrahierten Simulationen die Regelung oder arbeiten in gitterbasierten Umgebungen. Weiterhin wird vor allem das autonome Missionsmanagement nicht abgedeckt und die Kommunikation über ideale Verbindungen oder externe Programme abgewickelt, die nicht für Echtzeitanwendungen gedacht sind. Daher wird in dieser Arbeit versucht diese Lücke mit dem entworfenen Ansatz zu schließen. Jeder Agent betreibt eine vollständig eigenständige Einheit des Autonomiekerns, die auch unabhängig von der Simulation laufen kann. Die Daten innerhalb eines Kerns können über die Prozesse hinweg geteilt werden, jeder Kern generiert eigene Instanzen; z. B. von den Hindernisobjekten. Die Kommunikation erfolgt über ein echtzeitfähiges virtuelles Kommunikationsnetz mit physikalischer Ausbreitung (siehe Kapitel 3), welches mit realen Netzwerken verbunden werden kann. Weiterhin wird keine Reibung oder physikalische Interaktion ermöglicht, jedoch werden physikalische Dynamikmodelle als Regelungssysteme verwendet. Dadurch kann eine große Anzahl an Agenten simuliert werden, ohne wesentliche physikalische Einschränkungen durch die Kinematik oder die Datenübertragung weg zu abstrahieren.

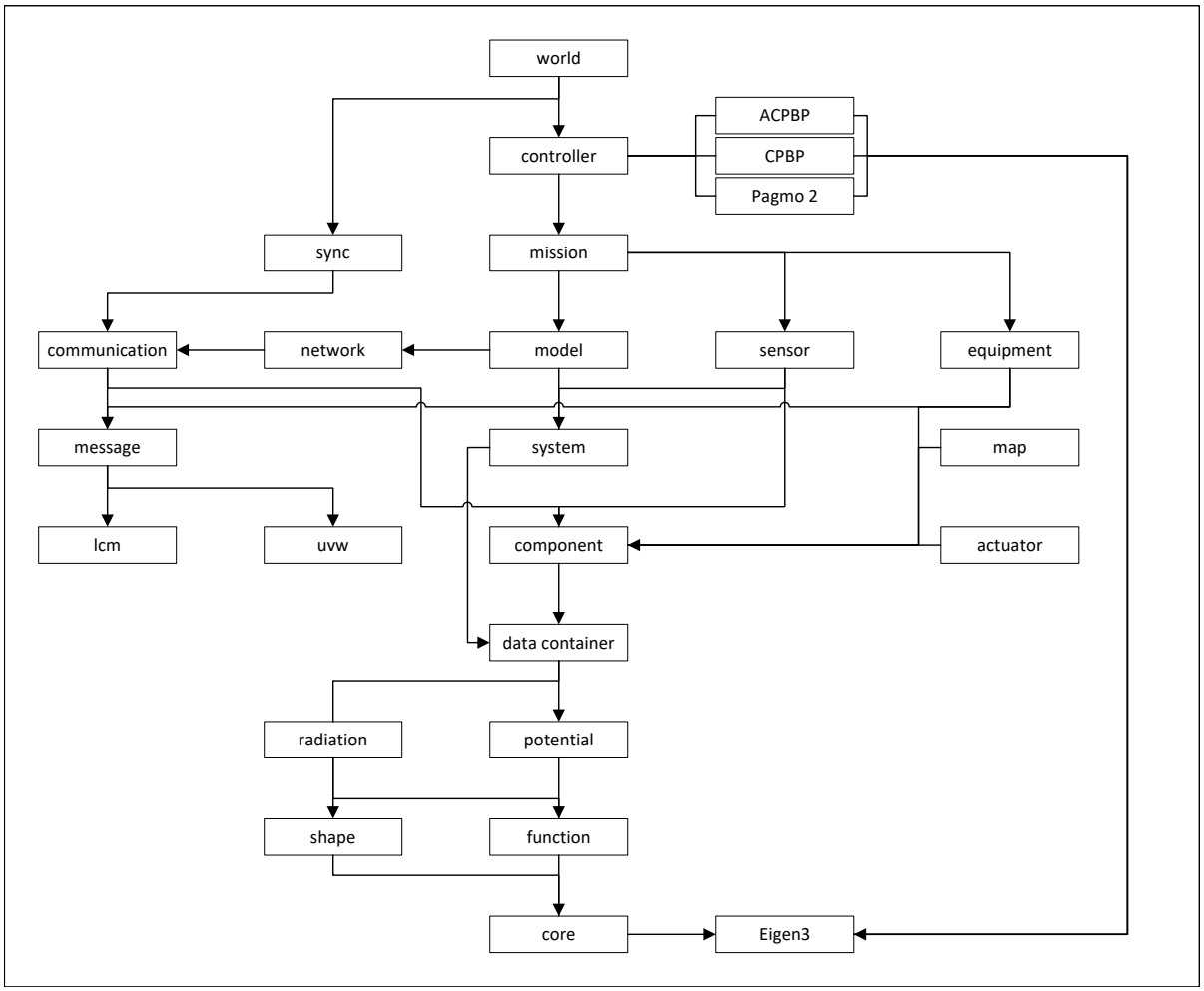


Abbildung 4.3.: Abhängigkeitsstruktur der Bibliotheken des Autonomiekerns.

4.2.2. Aufbau des Autonomiekerns

Der Aufbau und die internen Abhängigkeiten des Autonomiekerns lassen sich mittels Abbildung 4.3 visualisieren. Dabei ist hervorzuheben, dass die Graphstruktur zyklenfrei ist, die fast einen Baum bildet. Folglich lassen sich für die unterschiedlichen Anforderungen von kleinen beschränkten Realsystemen bis hin zu großen Simulationen modular nur die tatsächlich benötigen Komponenten einbinden. Dadurch wird ein minimaler Speicher- und Bauzeitverbrauch wie auch eine hohe Modularität und Wiederverwendbarkeit erreicht. Dies wird durch die konsequente Beachtung der Softwarekonstruktionsmetriken der geringen Kopplung zwischen den Modulen und einer hohen Kohäsion innerhalb der Module bewirkt. Jedes Modul bildet dabei eine statische C++ Bibliothek. Statische Bibliotheken erhöhen die Bauzeit und die Programmlänge im Speicher, sind dafür allerdings signifikant schneller zur Laufzeit.

4.2.3. „core“-Modul

Das „core“-Modul besteht aus Basisklassen und Hilfsfunktionen für die Ausgabe in der Konsole oder die einfache Graphgenerierung mittels „Gnuplot“. Die „Semaphore“-

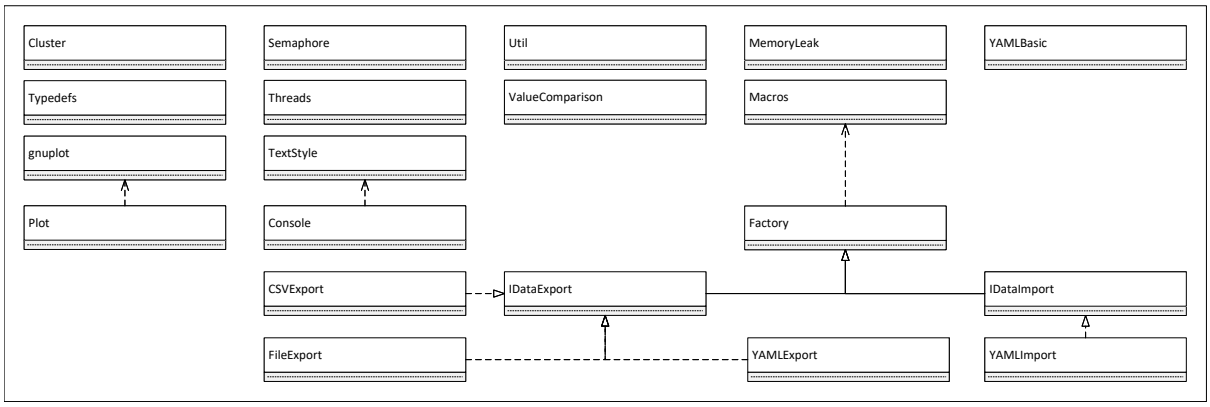


Abbildung 4.4.: Elemente des „core“-Moduls.

Implementierung wird für die Subprozessverwaltung genutzt, um aktives Warten der Prozesse zu vermeiden. Die wichtigsten beiden Dateien umfassen die „Macros“ und „Factory“. Makros sollten generell vermieden werden, ermöglichen jedoch die automatische textuelle Quelltexterzeugung. Diese wird genutzt um standardisierte Zugriffsfunktionen, Variablennamen und Komponentenfunktionen zu erzeugen (siehe Quelltext 4.1 und Abschnitt 4.2.22). Die wichtigste Funktionalität ist jedoch der automatische Import und Export einer Klasse, die in das YAML-Format serialisiert wird. Dies wird durch ein einzeiliges Makro erreicht. Alternativ zum YAML-Format wird ein Textformat und ein CSV-Format angeboten. Dazu müssen die Klassen eine „encode“- und „decode“-Funktion implementieren.

Quelltext 4.1: Getter und Setter für eine automatische Klassengenerierung

```
#define GETTER_REF(type, Name, attribute)
\
    inline type &puzi_CAT(get, Name)()
\
    {
\
        return puzi_CAT(m_, attribute);
\
    }

#define SETTER_REF(type, Name, attribute)
\
    inline void puzi_CAT(set, Name)(const type &attribute)
\
    {
\
        puzi_CAT(m_, attribute) = attribute;
\
    }

#define GETTER_SETTER_REF(type, Name, attribute)
\
    GETTER_REF(type, Name, attribute)
\
    SETTER_REF(type, Name, attribute)
```

Fabrik Das Entwurfsmuster der Fabrik (siehe „Factory“ und vgl. [89]) wird angewendet, um auf Basis des Klassennamens zur Laufzeit Objekte zu erzeugen, die erst

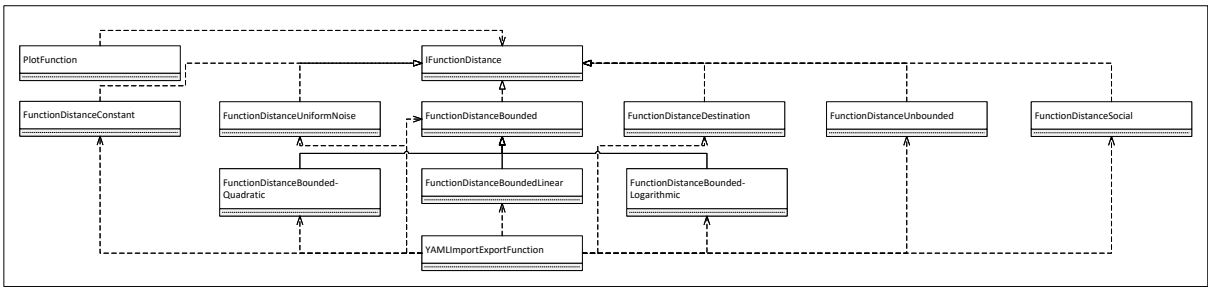


Abbildung 4.5.: Elemente des „function“-Moduls.

durch weitere Module oder externe Bibliotheken hinzugefügt werden und zur Bauzeit des statischen Kerns nicht bekannt sind. Dafür werden klassischer Weise Objekte bei einer entsprechenden Fabrik registriert. Bei der Registrierung wird von jedem Typ ein Objekt mittels des Standardkonstruktors erzeugt. Dieses Vorgehen hat in dem vorliegenden Projekt den großen Nachteil, dass die Klassen zum einen nur mit dem Standardkonstruktor erzeugt werden können und zum anderen, dass große Menge an Speicherplatz für die Speicherung der Objekte einiger komplexer Klassen verbraucht wird. Daher wird dieses Entwurfsmuster um eine weitere Stufe erweitert, die fortan als „Maschine“ bezeichnet wird. Diese bilden eine Erzeugerklasse, die eine geringere Speichersignatur aufweist. Zudem können auch verschiedene Konstruktoren der tatsächlichen Objekte genutzt und Parameterlisten in diese Erzeuger geladen werden. Die „Maschinen“ produzieren somit die Produkte bzw. Objekte innerhalb der Fabrik. Eine Standardimplementierung der Fabrik arbeitet auf dem statischen Speicher und verwaltet eine unsortierte Abbildung auf diesem. Die Abbildung wird in den dynamischen Speicher verschoben, wodurch sich die Ladezeiten verlängern. Allerdings wird der statische Speicher und der Cache zur Laufzeit entlastet, sodass eine hohe Ausführungsgeschwindigkeit erreicht werden kann. Dies ist besonders sinnvoll, da mehrere verschiedene Fabriken erzeugt werden.

4.2.4. „function“-Modul

Das „function“-Modul umfasst verschiedene Funktionstypen, die zur Erzeugung von Potenzialfeldern benötigt werden (vgl. [141]). Diese bilden anschließend Missionen (siehe Kapitel 5), Hindernisse oder Übertragungshindernisse ab, die das Netzwerk stören. Die Basis der distanzbasierten Kostenfunktionen bietet die „IFunctionDistance“-Schnittstelle (siehe Abbildung 4.5). Für diese existiert eine generische Zeichenfunktion und sie deklariert die Funktionen „getValueByDistance“ und „getValueByNonZeroDistance“ (siehe Quelltext 4.2).

Quelltext 4.2: Funktionsdeklaration der Funktionsschnittstelle

```
const float getValueByDistance(const float &distance) const;
const float getValueByNonZeroDistance(const float &distance) const;
```

Die Unterscheidung der Funktionen für den Spezialfall, dass die Distanz $d = 0$ ist, kann vor allem für gebrochenrationale Funktionen und logarithmische Funktionen zu effizienteren Implementierungen führen. Ansonsten ruft diese Funktion nur die

eigentliche „getValueByDistance“-Funktion auf. Die Funktionsklasse „FunctionDistanceConstant“ realisiert eine einfach parametrisierbare konstante Funktion:

$$\begin{aligned} l_{\text{konst}}(d) &= c_l \\ c_l &\in \mathbb{R} \end{aligned} \tag{4.2.1}$$

Eine um ein mit a_l skaliertes gleichverteiltes Rauschen erweiterte Variante der konstanten Funktion bildet „FunctionDistanceUniformNoise“ ab:

$$\begin{aligned} l_{\text{rauschen}}(d) &= c_l + \epsilon_l \\ c_l \in \mathbb{R}, \epsilon_l &\sim \mathcal{U}(-\frac{a_l}{2}, \frac{a_l}{2}) \end{aligned} \tag{4.2.2}$$

Mathematisch unbegrenzte Funktionen („FunctionDistanceUnbounded“), z. B. gebrochenrationale Funktionen, die folgendes Verhalten aufweisen, werden durch einen Maximalwert a_l beschränkt, um numerisch stabil zu bleiben.

$$l_{\text{unbegrenzt}} : \mathbb{R}_0^+ \to \mathbb{R}$$

$$\lim_{d \to \infty} l_{\text{unbegrenzt}}(d) = 0 \tag{4.2.3}$$

$$\lim_{d \to 0} l_{\text{unbegrenzt}}(d) = \pm\infty \tag{4.2.4}$$

$$l_{\text{unbegrenzt}}(d) = \begin{cases} \frac{b_l}{d} & \text{, falls} |\frac{b_l}{d}| \leq |a_l| \\ a_l & \text{, sonst} \end{cases} \tag{4.2.5}$$

Eine besondere Form der gebrochenrationalen Funktionen bietet die Robotersozialfunktion (vgl. [149]), die mit der Funktionsklasse „FunctionDistanceSocial“ implementiert wird.

$$\begin{aligned} l_{\text{sozial}}(d) = \begin{cases} \frac{b_{1l}}{d^{\gamma_{1l}}} - \frac{b_{2l}}{d^{\gamma_{2l}}} & \text{, falls} |\frac{b_1}{d^{\gamma_1}} - \frac{b_2}{d^{\gamma_2}}| \leq |a_l| \\ a_l & \text{, sonst} \end{cases} \\ \text{mit } b_{1l}, b_{2l} \geq 0, \gamma_{1l} > \gamma_{2l} > 0 \end{aligned} \tag{4.2.6}$$

Diese bietet bei der passenden Parametrisierung ein globales Minimum L_- bei einer gewünschten Distanz $d_{\text{wunsch}} \in \mathbb{R}^+$ und ein lokales mit a_l begrenztes Maximum.

$$b_{1l} = -2 \cdot L_- \cdot d_{\text{wunsch}}^3 \tag{4.2.7}$$

$$\gamma_{1l} = 3 \tag{4.2.8}$$

$$b_{2l} = -3 \cdot L_- \cdot d_{\text{wunsch}}^2 \tag{4.2.9}$$

$$\gamma_{1l} = 2 \tag{4.2.10}$$

Zuletzt bleiben noch die Funktionen, die für jede rellwertige Distanz $d \in \mathbb{R}$ auch begrenzte rellwertige Werte bestimmen. Die grundlegenden Eigenschaften sind in der Klasse „FunctionDistanceBounded“ zusammengefasst und werden mit der jeweiligen

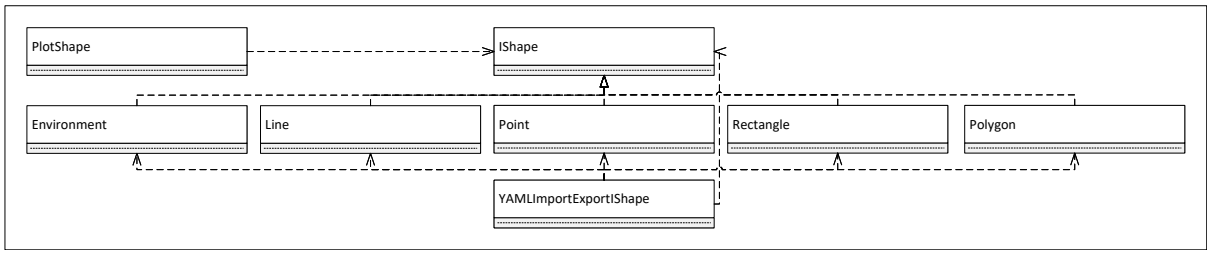


Abbildung 4.6.: Elemente des „shape“-Moduls.

Funktion realisiert.

$$l_{\text{linear}}, l_{\text{quadratisch}} : \mathbb{R}_0^+ \to \mathbb{R}$$

$$l_{\text{linear}}(d) = \begin{cases} \frac{-a_l}{r_l} \cdot d + a_l & \text{, falls } d \leq r_l \\ 0 & \text{, sonst} \end{cases} \tag{4.2.11}$$

$$l_{\text{quadratisch}}(d) = \begin{cases} \frac{-a_l}{r_l^2} \cdot d^2 + a_l & \text{, falls } d \leq r_l \\ 0 & \text{, sonst} \end{cases} \tag{4.2.12}$$

Die Klassen „FunctionDistanceLogarithmic“ und „FunctionDistanceDestination“ werden in den Abschnitten 5.1.1 und 5.8 genauer erläutert.

4.2.5. „shape“-Modul

Mit Hilfe des „function“-Moduls können Potenzialfelder für Hindernisse erzeugt werden, sodass Agenten diese Hindernisse aufgrund steigender Kosten meiden. Dazu wird vor allem der Abstand der Agenten zu diesen Hindernissen benötigt. Dieser minimale Abstand wird über die Position und die geometrischen Form der Hindernisse bestimmt. Somit umfasst dieses Modul (siehe Abbildung 4.6) Basisformen, aus denen Hindernisse bestehen können, die anschließend die entsprechende Distanz als Eingabe für die distanzbasierten Funktionen liefern. Diese Formen sind Punkte, Linie, Rechtecke, Vielecke und Umgebungen. Umgebungen, die mit der Klasse „Environment“ realisiert werden, umfassen das gesamte Operationsgebiet der Agenten und besitzen deshalb stets den Abstand $d = 0$, welcher nicht kompatibel mit der Funktion „getValueByNonZeroDistance“ ist. Auch diese Klassen und ihre Parameter können mittels Im- und Export als YAML-Dateien serialisiert werden. Ferner können die Formen graphisch dargestellt werden.

4.2.6. „potential“-Modul

Das Hauptelement des „potential“-Moduls bildet die Schnittstelle „IPotentialObject“ (siehe Abbildung 4.7), welche die Kernfunktion „getPotential“ deklariert. Diese gibt abhängig vom übergebenen Agentenzustand einen Kostenwert zurück. Potenziale, die selber eine eigene Position besitzen, werden in „IPotentialWithPosition“ abstrahiert. Ein Realisierung dafür bildet die „IPotentialShape“-Klasse, welche aus einer Funktion (siehe Modul „function“) und einer Form (siehe Modul „shape“) besteht. Diese Klasse

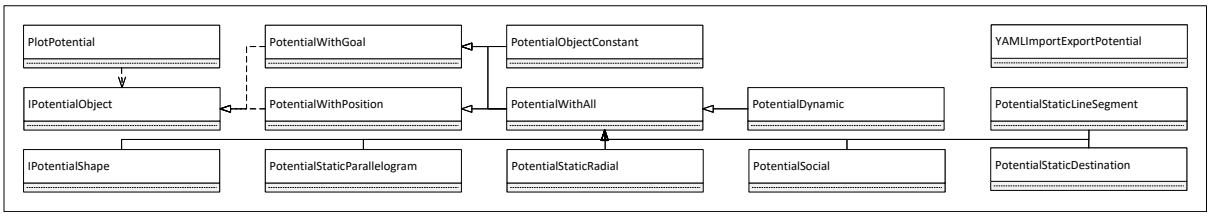


Abbildung 4.7.: Elemente des „potential“-Moduls.

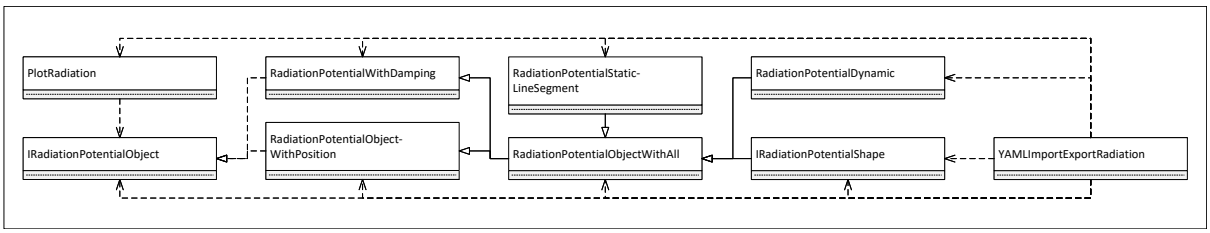


Abbildung 4.8.: Elemente des „radiation“-Moduls.

löst die meisten anderen Potenzialklassen ab, die aus Kompatibilitätsgründen von der Arbeit Puzicha [141] übernommen werden. Für „IPotentialShape“ gibt es sowohl eine dynamische Implementierung als auch eine statische mittels Vorlagenklassen. Dieses Modul dient der Erzeugung von physischen Hindernissen wie auch als Grundlage jeder Mission (siehe Kapitel 5), die von „IPotentialObject“ indirekt erbt, in der Optimierung.

4.2.7. „radiation“-Modul

Das „radiation“-Modul ist ein Analogon des „potential“-Moduls für netzwerkbezogene Hindernisse und Umgebungen, die die Funkübertragung beeinflussen. Daher ist auch die Benennung analog (siehe Abbildung 4.8). Ein Störsender kann beispielsweise als radiales begrenztes punktförmiges Netzwerkhindernis betrachtet werden, das den Signal-Rausch-Abstand (SNR) verschlechtert. Um einen Einfluss auf die Empfangsfeldstärke (RSSI) auszuüben, muss das Hindernis die Sichtverbindung vom Sender zum Empfänger schneiden. Dies unterschiedet die Bibliotheken, denn das „potential“-Modul arbeitet nur auf Zustandspunkten und das „radiation“-Modul berechnet Schnittpunkte zwischen Strecken und Objekten und gibt Dämpfungswerte zurück.

4.2.8. „data container“-Modul

Eine der größten Weiterentwicklungen bilden die Datencontainer bzw.-speicher. Diese kapseln zusammengehörige Daten in kontinuierlichen Speicherblöcken und ermöglichen lesenden und schreibenden Zugriff mittels Zeigern. Um Konflikte durch konkurrierende Zugriffe zu vermeiden, gibt es häufig einen einzigen Schreiber und eine unbegrenzte Anzahl an Lesern oder einen Schutz mittels des gegenseitigen Ausschlusses (Englisch: „mutex“). Unter bekannten Voraussetzungen kann ein Überschreiben der Werte zur Laufzeitverbesserung akzeptiert werden. Ein Beispiel dafür bildet der

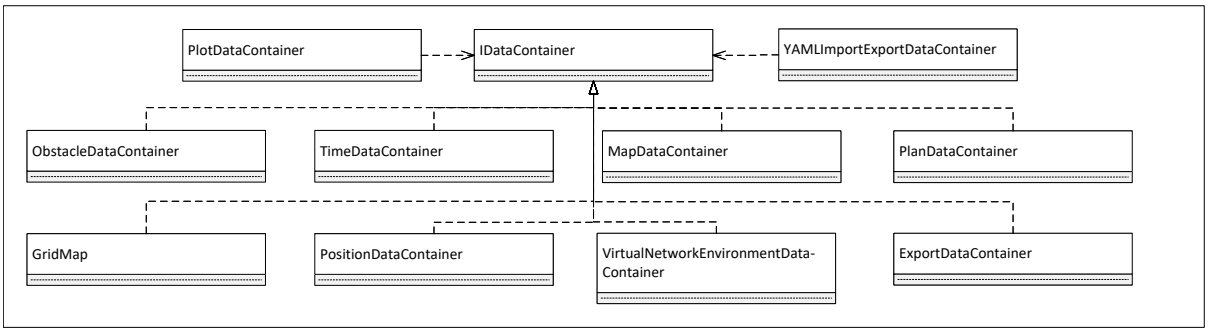


Abbildung 4.9.: Elemente des „data container“-Moduls.

Positionsdatencontainer (siehe Komponente „PositionDatacontainer“). Die Position wird hauptsächlich anhand der schnell zu berechnenden Koppeldaten aktualisiert. Das satellitenbasierte System hat eine deutlich geringere Aktualisierungsfrequenz, liefert aber präzise Daten und überschreibt die Positionsdaten ohne gegenseitigen Ausschluss, sodass die Kopplung zurückgesetzt wird und mit den neuen Daten im Speicher arbeitet. Der „TimeDatacontainer“ speichert die Zeit des Systems und kann mittels der „TimeSyncComponent“ (siehe Abschnitt 4.2.20) für alle Netzwerkteilnehmer synchronisiert werden. Realsysteme können eine synchrone Zeit von Satelliten erhalten. „PlanDatacontainer“ speichert die geplanten Zustands-, Stellgrößen- und Zeittrajektorien. „ObstacleDatacontainer“ und „MapDatacontainer“ kapseln jeweils Hindernisdaten. Im Vergleich zur vorherigen Version wird klar zwischen Kartendaten und sensorbasierten Hindernisdaten unterschieden. Die sensorbasierten Hindernisse werden in drei Kategorien unterteilt, die je in einer eigenen Liste verwaltet werden. Es wird zwischen statisch lokal, statisch global und dynamisch unterschieden. Dynamische Hindernisse werden nur verwaltet, falls sie im Sichtbereich der Sensoren sind. Statische Hindernisse werden jenseits des Erfassungshorizontes in einer globalen Liste gespeichert. Die lokale Liste ist eine temporäre Liste für die Sensoren und die Nahfeldplanung. Die Karte wird ebenfalls in einer globalen und lokalen Hindernisliste verarbeitet und besitzt eine Rasterkarte (siehe Komponente „GridMap“) als Abbild der Bodengegebenheiten, die die Bewegungsfähigkeit der Agenten beschreibt. Die Trajektorienplanung des Optimierers verwendet nur die lokalen Kartendaten, deren Auswahl in Abschnitt 4.2.11 beschrieben wird. Der „VirtualNetworkEnvironmentDatacontainer“ ist ähnlich zum Kartenspeicher und enthält eine Rasterkarte mit den Rauschleistungen der Umgebung. Diese ist z. B. in Städten deutlich höher als im ländlichen Raum oder wird durch nukleare Katastrophen und Störsender bestimmt.

4.2.9. „component“-Modul

Das „component“-Modul beschreibt die Grundlage aller technischen Komponenten bzw. Teile im System. Dies beinhaltet sowohl Software als auch Hardware. Diese können mittels eines Zustandsübergangsgraphen beschrieben werden (siehe Abbildung 4.11). Eine Komponente ist nach ihrer Erzeugung in einem uninitialisierten Zustand. Dieser wird über eine Initialisierungsmethode verlassen, danach kann sie gestartet und aktiviert werden. Rückwärtig erfolgt nach der Deaktivierung und dem

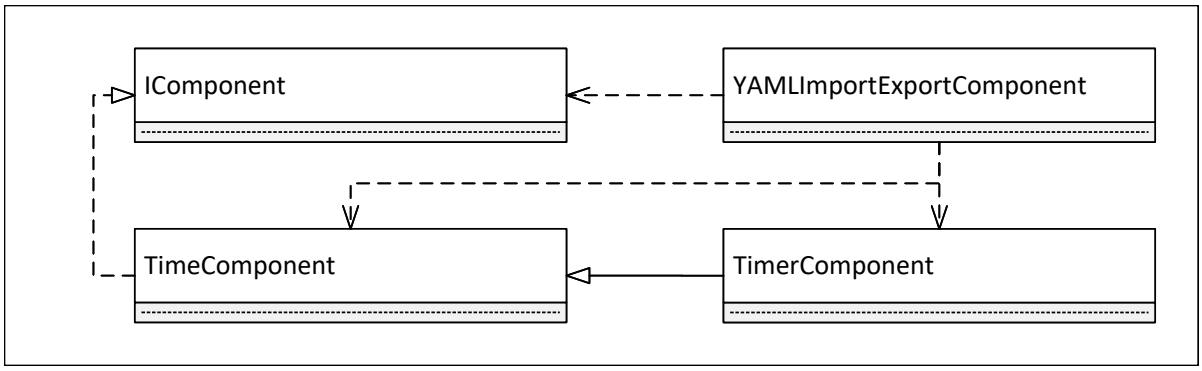


Abbildung 4.10.: Elemente des „component“-Moduls.

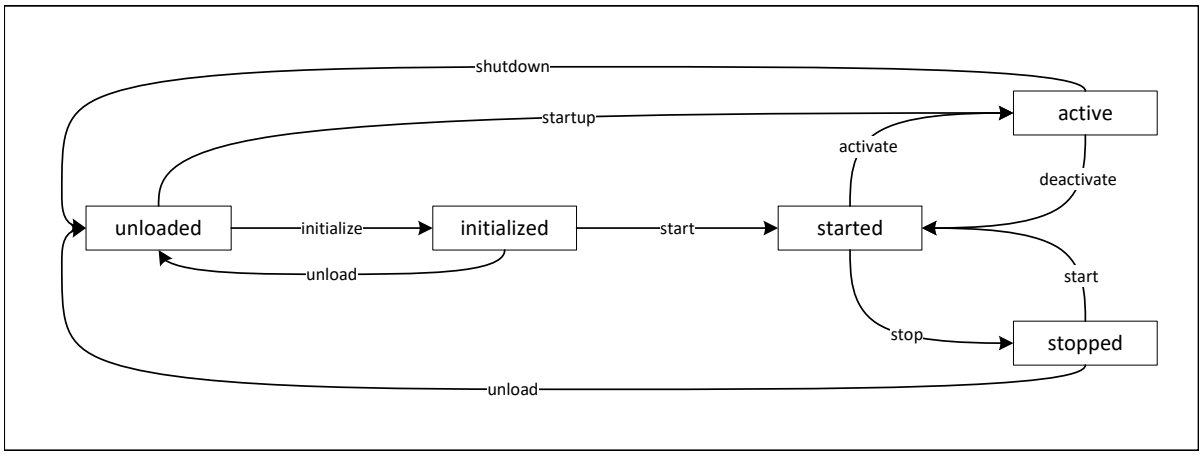


Abbildung 4.11.: Zustandsübergangsgraph der Komponenten.

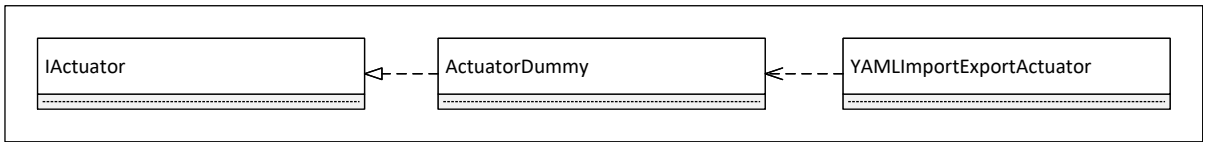


Abbildung 4.12.: Elemente des „actuator“-Moduls.

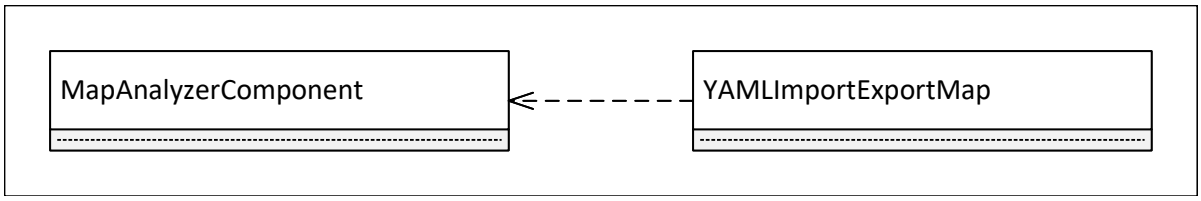


Abbildung 4.13.: Elemente des „map“-Moduls.

Stoppen der Komponente der Entladevorgang. Der Startvorgang wird mit „startup“ und der Beendigungsvorgang mit „shutdown“ zusammengefasst. Eine Erweiterung bieten die zeitbasierten Komponenten, denen ein Frequenz übergeben wird, mit der eine Rücksprungfunktion aufgerufen wird. Es existiert je eine Rücksprungfunktion für den aktivierten und den deaktivierten Zustand. Letzterer kann aus dem Bauprozess ausgeschlossen werden. Die Stoppuhrkomponente („TimerComponent“) realisiert eine Stoppuhrfunktionalität, indem mit Hilfe der Rücksprungfunktion die vergangene Zeit summiert wird.

4.2.10. „actuator“-Modul

Ein Aktor kann beispielsweise der Antriebsstrang eines Agenten sein (siehe Abschnitt 6.2.1). Diese Funktionalität und die notwendigen Daten werden im Allgemeinen durch die „IActuator“-Schnittstelle zur Verfügung gestellt. Sie beinhaltet eine zeitbasierte Komponente, die einen Zeiger auf den „PlanDatacontainer“ besitzt. Daraus lassen sich die geplanten Stellgrößen herauslesen und mit der gewünschten Rate auf ein System geben. Ein Beispiel bieten die ROS und ROS2-Erweiterungen, die die Stellgrößen für ROS konvertieren und verschicken (sieh Abschnitt 4.2.25).

4.2.11. „map“-Modul

Das „map“-Modul umfasst, bis auf die Import- und Exportfunktionen, eine Komponente. Die „MapAnalyzerComponent“ reduziert bzw. selektiert diejenigen Hindernisse, die in der Trajektorienberechnung berücksichtigt werden sollen. Da die Berechnung der „getPotential“-Funktion für sehr viele Hindernisse, die weit außerhalb des Prädikitionshorizontes liegen, nur die Laufzeit der Optimierung verschlechtern, werden diese herausgefiltert. Dabei existieren verschiedene Ansätze zur Reduktion der Funktionsauswertungen. Dies kann über eine Multiskalenrepräsentation (Englisch: „Level Of Detail“, kurz LOD) erreicht werden. Bei diesem auf „Mipmapping“ basierenden Verfahren wird die Auflösung der Hindernisse reduziert und durch Verschmelzen dieser groben Konturen wird die Anzahl der Hindernisse und somit der zu evaluierenden Potenzialobjekte ebenfalls reduziert. Der Grad der Auflösungsreduktion soll

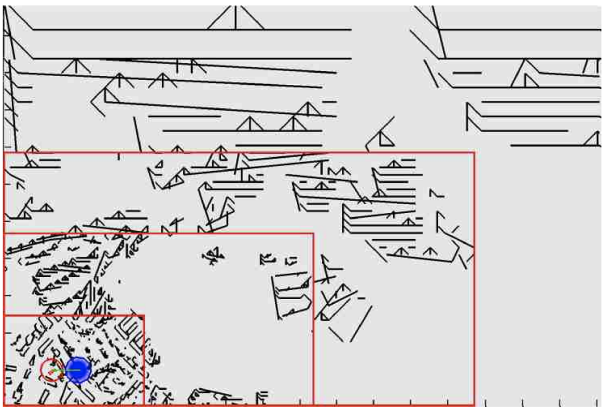


Abbildung 4.14.: Multiskalenrepräsentation der Hindernisse aus der Abschlussarbeit Kaluzny [77], die aus einer Karte von Dortmund geladen werden. Es werden vier Auflösungsstufen (LOD) dargestellt. Jedoch sind die gröberen Auflösungsstufen nicht für eine realistische Simulation brauchbar. Ein schwarzer Strich der unteren und linken Skala markiert 100 m.

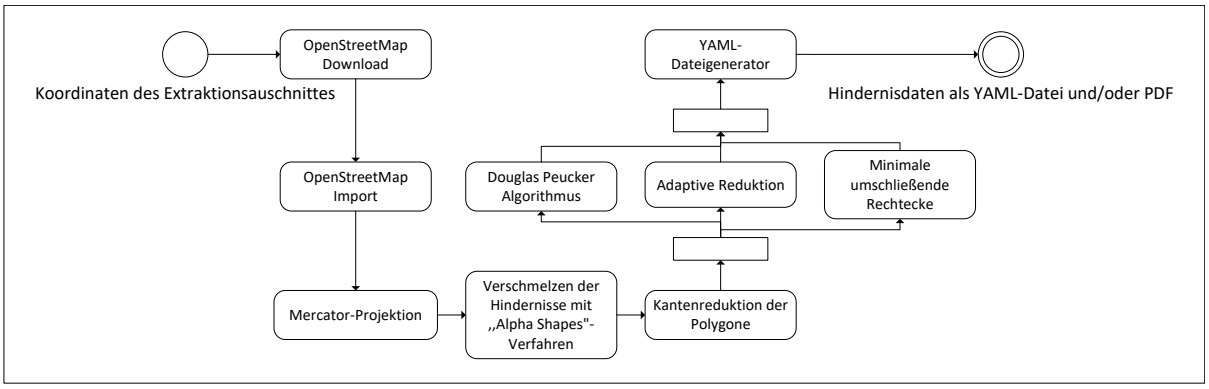


Abbildung 4.15.: Verarbeitungsfluss des Programms von Koordinaten bis hin zu Szenariobeschreibungen (vgl. [28]).

dabei von der Entfernung vom Agenten abhängen (vgl. [77]). Aufgrund der geringen Ergebnisqualität wird jedoch eine Filtertechnik angewendet (siehe Abbildung 4.14), die zyklisch eine Liste aller Hindernisse im Umkreis aktualisiert. Damit allerdings auch globale Pfade oder Missionen berechnet werden können, erhalten diese stets den Zugriff auf alle verfügbaren globalen statischen Hindernisse (siehe Abschnitt 5.15). Eine Alternative bietet das „Level-Set"-Verfahren, bei dem die Komplexität bzw. die Auflösung eines Hindernisses implizit durch eine Funktion repräsentiert wird (vgl. [4]). Das Verfahren eignet sich besonders für Polygonzüge.

4.2.11.1. Kartendatenextraktion

Um tatsächlich reale Katastrophenszenarien zu simulieren und zu untersuchen, werden auch reale Umgebungen benötigt. Dazu wird eine Exportprogramm entwickelt, welches automatisch öffentlich verfügbare Kartendaten von „OpenStreetMaps" extrahiert, reduziert und in adäquate Potenzialobjekte transformiert (siehe Abbildung 4.15 und vgl. [28]). Das Programm beginnt mit dem Herunterladen und dem Importieren der Kartendaten von „OpenStreetMap". Anschließend wird die Merkatorprojektion

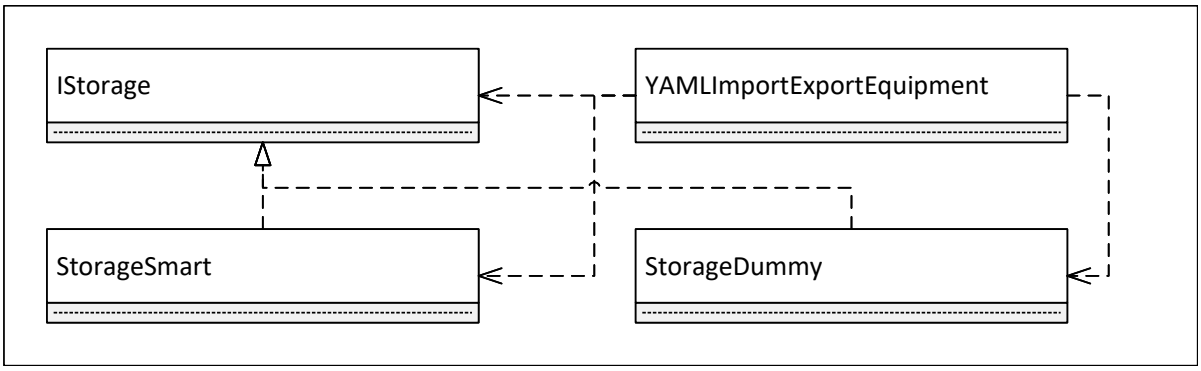


Abbildung 4.16.: Elemente des „equipment“-Moduls.

genutzt, um die geographischen Koordinaten der Polygonzüge der Hindernisse in eine kartesische Darstellung umzuwandeln. Nun werden diese Polygon mit Hilfe des „α-Shape“-Verfahrens unter der Berücksichtigung der Agentenbreite verschmolzen. Die Breite legt fest, welchen Abstand Hindernisse aufweisen müssen, um für den Agenten als getrennt beachtet zu werden. Trotz dieser Reduktion sind die Kartendaten der Polygone deutlich zu detailreich, um sie zur Wegplanung zu verwenden. Daher werden diese mit drei verschiedenen Reduktionsalgorithmen geglättet und weiter reduziert. Die generierten Szenariobeschreibungen werden als YAML-Datei (siehe Quelltext 4.3) gespeichert und von dem „MapDataContainer“ in den Speicher für globale statische Hindernisse geladen.

4.2.12. „equipment“-Modul

Das „equipment“-Modul soll perspektivisch verschiedene Ausrüstungskomponenten für die Agenten bieten. Es umfasst zur Zeit eine Schnittstelle zur Aufnahme und Ablage von Objekten. Dies wird in der Schnittstelle „IStorage“ beschrieben. Die Realisierung „StorageDummy“ lagert ein Objekt nur ein und aus, wohingegen die intelligente Variante „StorageSmart“ eine Nachricht in das Netzwerk sendet. Diese Nachricht beinhaltet die Aktion, Aufnahme oder Ablage, und die Position des Agenten anhand des „PositionDatacontainer“. Die Nachricht wird dazu an einen Sender übergeben.

4.2.13. „message“-Modul

Die in Abschnitt 3.6 bereits genannten Nachrichten realisieren alle die „IMSG“-Schnittstelle. Diese Schnittstelle fasst Funktionen und Daten zur Berechnung der Größe der Nachricht und ihrer Attribute, die Identifizierungsnummern, die Zeitstempel und den Nachrichtentyp zusammen. Zudem deklariert sie die Kodierungs- und Dekodierungsfunktionen für die „Lightweight Communications and Marschalling“ (kurz LCM) Bibliothek (vgl. [66]). LCM bietet eine gute Funktionen zur Kodierung und Dekodierung von Klassen und Daten auf Byte-Ebene, die dann von Netzwerkbibliotheken als Datenströme oder Datenpakete verschickt werden können. Als Netzwerkbibliothek wird „libuv“ mit der C++ Erweiterung „uvw“ genutzt. Zur Nutzung von LCM wird für jede Nachricht eine Nachrichtendefinition erzeugt, die anschließend in eine C++-Klasse

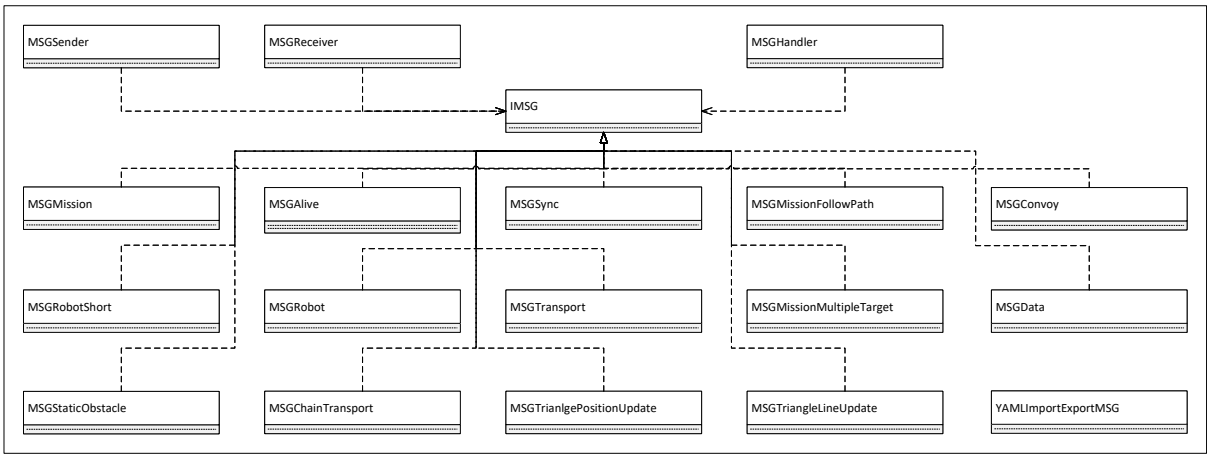


Abbildung 4.17.: Elemente des „message“-Moduls.

kompiliert und mit der eigentlichen Klasse verknüpft wird.
Die wichtigsten Nachrichten sind „MSGMission“, „MSGRobot“, „MSGStaticObstacle“, „MSGAlive“ und „MSGSync“. „MSGMission“ ist der Anfang und das Ende jeder Mission. Sie beschreibt, welche Mission mit welchen Parametern gestartet oder beendet werden soll. Wenn die jeweiligen Missionen gestartet sind und weitere Informationen und zyklische Aktualisierungen benötigen, wird dies durch dedizierte Nachrichten der jeweiligen Mission (siehe Kapitel 5) umgesetzt. Diese Nachrichten werden nicht von der Missionsbehandlung (siehe Abschnitt 4.2.19) verarbeitet, sondern direkt an die jeweilige Mission weitergereicht. „MSGRobot“ ist die zentrale Nachricht der Agenten, die über ihren Zustand und die geplanten Trajektorien des Zustands und der Stellgröße sowie der Zeit informieren. Dies ist mit Abstand die größte Nachricht (siehe Tabelle 3.4), die im Unterschied zu ereignisbasierten Nachrichten zyklisch gesendet wird. Eine dieser ereignisbasierten Nachrichten ist die „MSGStaticObstacle“. Sie wird gesendet, sobald die Sensoren ein nicht kartographiertes, statisches Hindernis entdecken. Somit können die anderen Agenten dieses Hindernis in ihren Speicher aufnehmen, ohne es selbst entdecken zu müssen. Dies lohnt sich allerdings nur bei statischen Hindernissen. „MSGAlive“ beinhaltet den Typ des Senders, sodass auf der Basis dieser Nachricht ein passendes Modell des Senders im Modelldatencontainer (siehe Abschnitt 4.2.18) generiert werden kann. Die „MSGSync“-Nachricht beinhaltet einen Zeitstempel, um die Uhren der Agenten im Netzwerk synchron zu halten. Die logische Ordnung der Nachrichten wird mittels der erweiterten Lamport-Uhren erreicht (siehe Algorithmus 3.1). Mit „MSGData“ lassen sich beliebige Daten über das Netzwerk verschicken und die Nachricht ist vor allem für das Weiterleiten von einsatzspezifischen Daten gedacht, die das Netzwerk nur als Datentransportmedium zu einer Basisstation nutzen oder für Softwareaktualisierungen.
Um die erstellten Nachrichten zu versenden, zu empfangen und zu verarbeiten werden drei Schnittstellen angeboten. So erhalten Missionen, die ihre eigenen Nachrichten versenden wollen, einen Zeiger auf eine Realisierung der „MSGSender“-Schnittstelle und sie nutzen die deklarierte Funktion „sendMSG“, welche die Nachricht in einen Puffer schreibt. Die tatsächliche Übertragung der Nachricht über ein Medium oder nur virtuell wird von Realisierungen der „ICommunication“-Schnittstelle übernom-

men (siehe Abschnitt 4.2.16). Als Nachrichtenempfänger können sich Realisierungen der „MSGReceiver“-Schnittstelle registrieren. Diese sind in der Regel dieselben Module bzw. Klassen wie die Sender. Sie nehmen die Bytefolge in Empfang und erzeugen daraus wieder die passenden Nachrichtenobjekte. Die Verarbeitung dieser geschieht jedoch in Realisierungen der „MSGHandler“-Schnittstelle, die sich anhand des gewünschten Nachrichtentyps bei „MSGReciever“-Objekten registrieren können. Diese besitzen eine Hashfunktion, die eine effiziente Abbildung eines Nachrichtentyps auf den passenden Verarbeiter ermöglicht. Ein Beispiel einer Realisierung bietet das „MissionModule“ (siehe Abschnitt 4.2.19).

4.2.13.1. Modellbasierte Nachrichtenkompression

Um Netzwerke mit niedrigen Nutzdatenraten wie LoRa (siehe Abschnitt 3.5) nutzen zu können, muss die von der Anwendung benötigte Nutzdatenmenge auf wenige Bytes reduziert werden. Für die Anwendung innerhalb eines Schwarms sind insbesondere die „MSGRobot“-Nachrichten (siehe Tabelle 3.4) das Hauptproblem aufgrund der großen Größe und der kurzen Senderate von $\frac{1}{0{,}3}$Hz. Diese Nachrichten erfordern eine Übertragungsrate von 39 227 bit s^{-1} pro Knoten, wobei alle anderen Nachrichtentypen vernachlässigt werden. Daher werden zwei Verfahren eingeführt, um große Datenpakete zu verschicken. Zum einen werden Datenpakete von bis zu 60 kB fragmentiert verschickt. Damit das Netzwerk nicht verstopft wird, dürfen diese aber nur eine geringe Ankunftsrate aufweisen. Die Nachrichten mit einer ebenfalls hohen Ankunftsrate müssen zum anderen zuvor komprimiert werden. Die Nachricht „MSGRobot“ besteht aus:

- 12 B für den Nachrichtenkopf
- 1 B für die Agentennummer
- 12 B für den Zustand $\mathbf{x}_0$ des Agenten
- 720 B für die Zustandstrajektorie
- 480 B für die Stellgrößentrajektorie
- 240 B für die Zeittrajektorie

In Kombination mit dem Modell der Netzwerkteilnehmer, das auf Grundlage der „MSGAlive“-Nachricht erzeugt wird, sind einige Informationen redundant. Diese werden als erstes entfernt. Da die Zeitschrittfolge $\Delta\mathrm{t} = [\Delta t_i]_{i=1,2,\ldots,N}$ bekannt ist, kann die Zeittrajektorie ausschließlich mit der Startzeit t_0 bestimmt werden:

$$t_k = t_0 + \sum_{j=1}^{k} \Delta t_j \tag{4.2.13}$$

Die Anwendung dieser Reduktion reicht nicht aus, um die Daten in einer LoRa-Nachricht mit höchstens 255 B einschließlich eines Paketkopfes unterzubringen. Jeder Wert der zweidimensionalen Stellgrößentrajektorie $\mathbf{u}_k = [u_{[1],k}, u_{[2],k}]$ wird durch eine

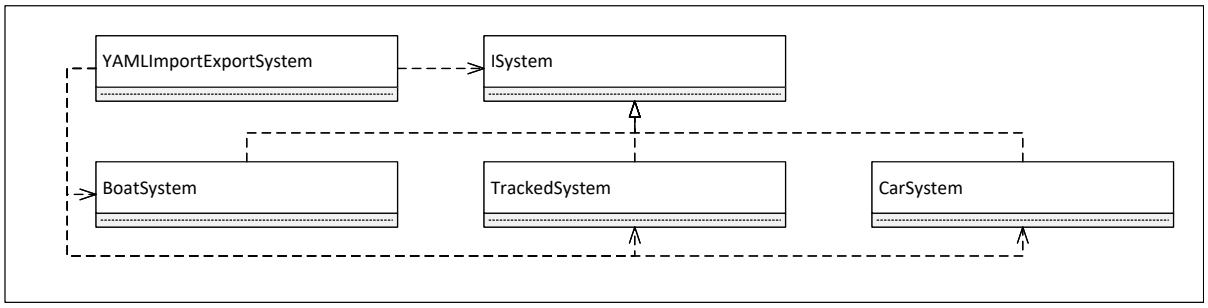


Abbildung 4.18.: Elemente des „system“-Moduls.

32 bit-Gleitkommazahl kodiert. Da die meisten Aktorsysteme jedoch keine Digital-zu-Analog-Wandler besitzen, die 2^{32} verschiedenen digitalen Aktionen pro Dimension unterstützen, werden die Steuersignale in ein Intervall zwischen 0 und 1 transformiert. Dies wird durch die Division der Intervallgröße, beschrieben durch die minimale ${}^{\diamond,\dashv}\mathbf{u}_{-}$ und die maximale ${}^{\diamond,\dashv}\mathbf{u}_{+}$ Stellgröße pro Dimension, erreicht:

$$\mathbf{i}_k = \mathbf{i}_{[j],k} = \frac{u_{[j],k} - u_{[j]-}}{u_{[j]+} - u_{[j]-}} \text{ mit } j \in \{1, 2\} \tag{4.2.14}$$

Dann wird dieses Intervall $\mathbf{i}$ mit 1 B abgetastet:

$$\mathrm{i}_k = \lfloor \mathbf{i}_k \cdot 256 \rfloor \tag{4.2.15}$$

Die Komprimierung reduziert die Stellgrößentrajektorie auf 120 B. Die Stellgrößentrajektorie definiert mit der aus dem Startzeitpunkt t_0 gewonnen Zeittrajektorie und dem Startzustand die Zustandstrajektorie eindeutig. Somit werden diese beiden Trajektorien verworfen. Der präzise Startzustand t_0 benötigt 4 B. Die Positionsdaten werden in ähnlicher Weise auf 2 B für jede der drei Dimensionen reduziert. Dies ist möglich, weil die Agenten so konfiguriert sind, dass sie in einem Bereich von 25 km^2 arbeiten. Daher sind auch die Positionsdaten beschränkt. Der Autonomiekern verwendet anschließend die Mercator-Projektion, um die Koordinaten auf der Grundlage der metrischen Daten neu zu berechnen. Alle diese Komprimierungstechniken verringern die Nachrichtengröße von 1468 B auf 144 B. Die kodierten Größen sind 1471 B im Vergleich zu 149 B. Folglich sind alle Nachrichten auch für Netzwerke mit geringer Übertragungsrate geeignet.

4.2.14. „system“-Modul

Das „system“-Modul enthält die drei in Abschnitt 2.3 präsentierten Systeme und bietet deren Dynamikfunktion sowie eine Simulationsfunktion für Zustandsübergänge bei gegebenem Startzustand, Stellgrößenfolge und Zeit an. Die Funktionen werden abstrakt in der Schnittstelle „ISystem“ deklariert. Dadurch kann jeder Regler jedes System verwenden.

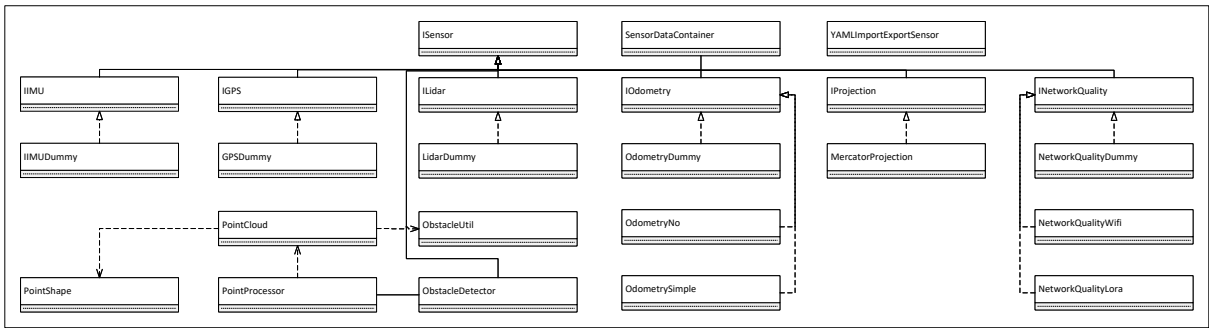


Abbildung 4.19.: Elemente des „sensor“-Moduls.

4.2.15. „sensor“-Modul

Das „sensor“-Modul deklariert eine übergeordnete Schnittstelle für alle Sensoren. Diese Schnittstelle „ISensor“ erbt von „IComponent“. Die Sensoren werden in ihrem eigenen Datencontainer verwaltet, der modular ausgetauscht werden kann. Dadurch muss nicht jeder Agent für jeden Sensortypen einen Verweis verwalten. Das Modul umfasst Schnittstellen für vier physische Sensoren und einen kalkulatorischen Sensor. „IOdometry“ beschreibt die Schnittstelle zur Berechnung von Koppeldaten. Dabei beschreibt die Ergänzung „Dummy“, dass die Daten invers aus dem Positionsdatencontainer gewonnen werden und anschließend zur Aktualisierung dieses angeboten werden. Diese Einheitsabbildung aus inverser Funktion und Funktion ist zum Testen der Funktionalität gedacht. Klassen mit der Bezeichnung „No“ liefern keine Daten. „Simple“ bezeichnet eine einfache, aber reale Implementierung der Sensoren. „IIMU“ kapselt Trägheits-, Rotations-, Beschleunigungssensoren und Magnetometer (Englisch: „Inertia Measurement Unit“, kurz IMU). Geographische Koordinaten werden mittels Satellitendaten über die Schnittstelle „IGPS“ erzeugt. Um diese in kartesische Koordinaten für die Regelung umzuwandeln, gibt es verschiedene Projektionsmethoden, die in „IProjection“ gekapselt werden. Die aktuelle Implementierung enthält nur die Mercator-Projektion als Standardreferenz. Die Schnittstelle „INetworkQuality“ bietet den Netzwerkkomponenten und den netzwerkbezogenen Missionen die Daten über die Qualität der Verbindungen. Die Schnittstelle wird für LoRa und IEEE 802.11 realisiert. Auch für „Light Detection And Ranging“-Sensoren (kurz LIDAR) ist eine Schnittstelle implementiert. Diese deklariert die Funktion „getPoints“, welche eine Liste an Tupeln, bestehend aus Abstand und Winkel, zurückgibt. Diese Liste wird anschließend von der Hindernisdetektionskomponente (siehe Komponente „ObstacleDetector“), unter der zur Hilfenahme mehrerer Punktverarbeitungsklassen, in Hindernisse repräsentierende Potenzialobjekte konvertiert.

4.2.15.1. Hindernisdetektion

Die Hindernisdetektion besteht aus drei Hauptschritten (siehe Abbildung 4.20). Zunächst werden die Tupel der Punktliste gefiltert. Die Filterung entfernt Tupel, die den minimalen oder maximalen Abstand der Sensoren unter- bzw. überschreiten und daher einen Distanzwert von 0 m aufweisen. Anschließend werden nah beieinanderliegende Punkte entfernt, um die Datenmenge zu reduzieren bei gleichzeitiger Wahrung

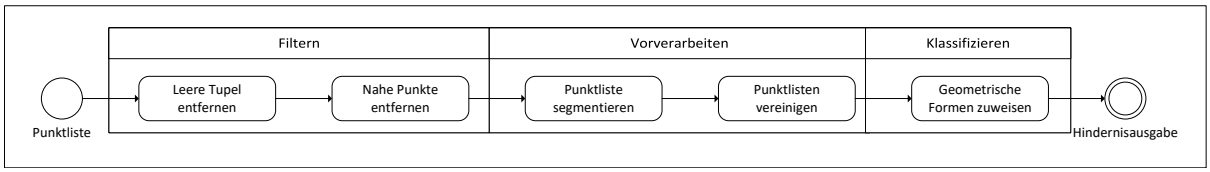


Abbildung 4.20.: Verarbeitungsfluss der Hindernisdetektion (vgl. [162]).

der signifikanten Positionen (vgl. [162]). Dies stellt eine Erweiterung der Standardmethodik dar (vgl. [199]). Die zweite Verarbeitungsstufe segmentiert die Punktliste konsekutiv anhand eines Abstandsschwellwerts und vereinigt danach naheliegende Punktlisten. Die resultierenden Punktlisten, die auch als Punktwolken bezeichnet werden, werden anschließend in kreisförmige, linienförmige oder rechteckige Potenzialobjekte transformiert. Zunächst wird geschaut, ob die Punktliste mehr als eine maximale Punktanzahl besitzt. Falls dies nicht der Fall ist, wird ein Kreis generiert, sonst wird eine Ausgleichsstrecke berechnet. Überschreitet die Distanz eines Punktes der Liste zur Strecke einen Schwellwert, so wird anstatt einer Strecke eine rechteckige Repräsentation gewählt. Diese ist auf Grund der zuvor bestimmten Strecke und der Orthogonalen durch den Punkt des maximalen Abstandes einfach zu bestimmen (vgl. [162]). Zur Demonstration der Funktionsfähigkeit wird ein Experiment aus der Abschlussarbeit Rütter [162] durchgeführt.

Experiment 4.2.1: *Vermesse einen Gehweg mit einem LIDAR und lasse aus den aufgenommenen Daten Potenzialobjekte erzeugen.*

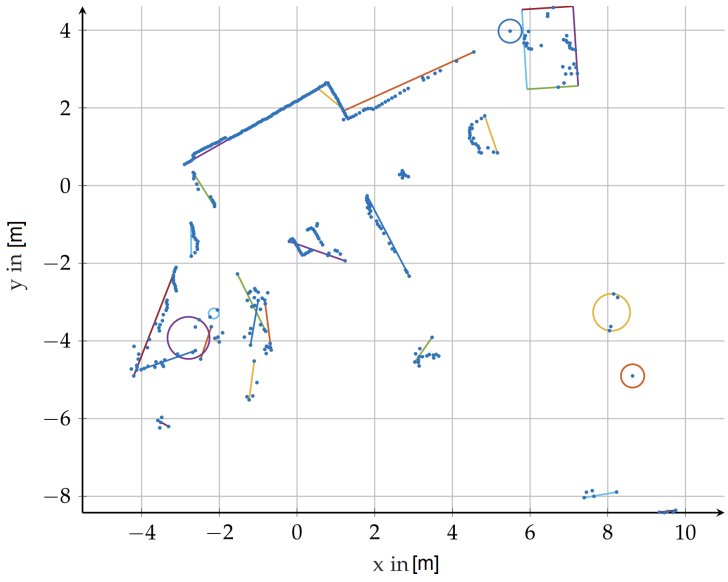


Abbildung 4.21.: Darstellung der Eingangspunktwolke und der resultierenden Objekte des Experimentes 4.2.1, übernommen aus der Abschlussarbeit Rütter [162].

Punkte	
Eingabe	802
Gefiltert	386
Punktlisten	
Segmentiert	36
Vereinigt	19
Potenzialobjekte	
Kreise	5
Strecken	24
Rechtecke	1
Laufzeit	
Filtern	1,37 ms
Segmentieren	2,73 ms
Vereinigen	1,46 ms
Klassifizieren	49,55 ms

Tabelle 4.1.: Numerische Auswertung des Experimentes 4.2.1 (vgl. [162]).

Das Experiment 4.2.1 zeigt, wie in sehr kurzer Zeit aus einer gemischt strukturierten Punktliste passende Potenzialobjekte erzeugt werden können (siehe Abbildung 4.21

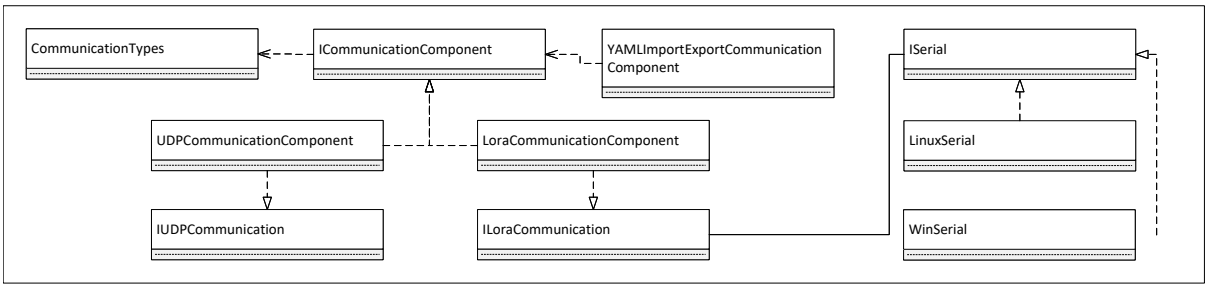


Abbildung 4.22.: Elemente des „communication“-Moduls.

und Tabelle 4.1). Einige Schwächen weist dieses Verfahren vor allem im unteren linken Bereich der Abbildung 4.21 auf. Dort werden einige überlappende Strecken erzeugt, wohingegen andere Bereich sehr passend konvertiert werden. Eine ausführliche Analyse erfolgt in der Arbeit Rütter [162]. Damit die Liste der Hindernisse nicht mit jeder neuen Punktliste wächst, werden die Potenzialobjekte vor dem Speichern mit Hilfe der Koppeldaten durch die Odometriesensoren auf Duplikate in der existierenden Hindernisliste überprüft. Da die Kopplung stets ungenau ist, wird nur auf Ähnlichkeit und nicht auf die Gleichheit der Objekte überprüft.

4.2.16. „communication“-Modul

Das „communication“-Modul umfasst Komponenten und Auflistungen zum Betrieb von Funkstationen. Unter den „CommunicationTypes“ werden Modulationsverfahren, MIMO-Verfahren, Modultypen und Klassifizierer zur Behandlung der Nachrichten zusammengeführt. Die Schnittstelle „ICommunication“ deklariert die Basisfunktionen zum Senden und Empfangen sowie Attribute für die verwendeten Trägerfrequenzen und Kanäle, die alle Funkmodule gemein haben. Die anderen beiden Schnittstellen „IUDPCommunication“ und „ILoraCommunication“ definieren den Zugriff auf das zugehörige reale Netzwerk und die dafür relevanten Funktionen. So beinhaltet die LoRa-Kommunikation eine serielle Schnittstelle, um die Module anzusprechen. Diese werden betriebssystemabhängig umgesetzt. Die Kombination der Schnittstellen ergibt nutzbare Kommunikationsknoten. Auf das „User Datagram Protocol“ (kurz UDP) wird mittels der Netzwerkbibliothek „libuv“ zugegriffen.

4.2.17. „network“-Modul

Um nicht nur reale Kommunikationsknoten zu betreiben wird ein simuliertes Netzwerk auf Grundlage der Erkenntnisse aus Kapitel 3 entworfen. Im Zentrum dieser Bibliothek steht die „VirtualNetwork“-Komponente, die aus einer Aggregation virtueller Kommunikationsknoten („VirtualCommunicationComponent“) und virtueller Verbindungen (siehe Modul „VirtualLink“) zwischen diesen besteht. Die Verbindungen simulieren ihre Qualität und Übertragungsleistung und verzögern dementsprechend das Weiterleiten der über sie verschickten Nachrichten. Jeder Teilnehmer des Netzwerks muss über eine „VirtualCommunicationComponent“ verfügen. Um nun reale Netzwerke mit diesem virtuellen Netz zu koppeln, werden Netzwerkbrücken für

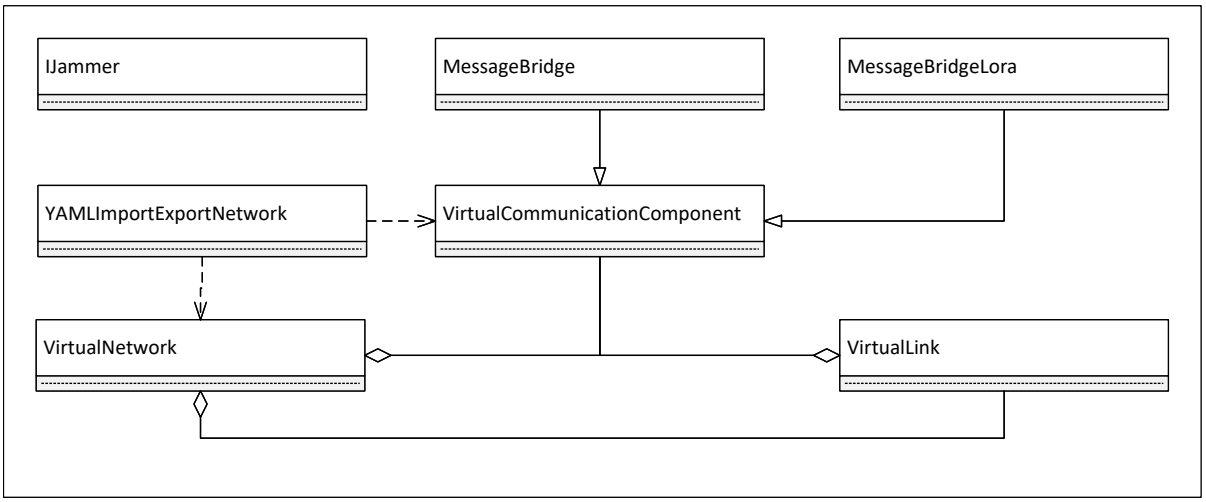


Abbildung 4.23.: Elemente des „network“-Moduls.

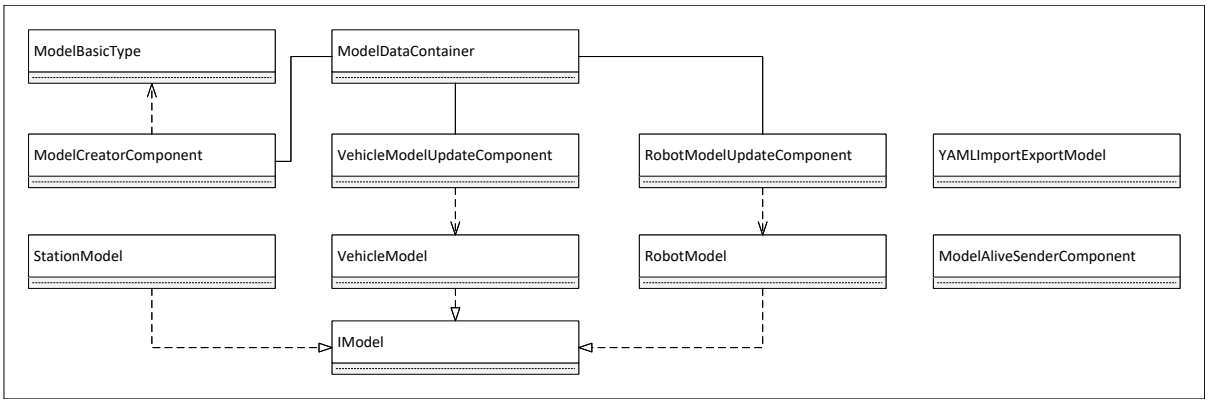


Abbildung 4.24.: Elemente des „model“-Moduls.

jede der beiden Technologien angelegt. Diese erben auf der einen Seite von „Virtual-CommunicationComponent“ und auf der anderen Seite von „IUDPCommunication“ oder „ILoraCommunication“, wodurch sie mit beiden Netzen verknüpft sind. Diese Komponenten erhalten, so wie alle Kommunikationsknoten im Netzwerk, eine Identifizierungsnummer. Erhalten sie nun eine Rundfunknachricht oder eine Nachricht, die für die entsprechende Nummer bestimmt ist, so wird diese in das jeweils andere Netzwerk übertragen. Somit bildet jede dieser Komponenten genau einen Knoten im realen und virtuellen Netzwerk ab. Weiterhin werden die Positionsdaten der entsprechenden Identifizierungsnummer mitgelesen, um eine korrekte Darstellung und Netzwerkqualitätsberechnung im virtuellen Netzwerk zu erzielen.

4.2.18. „model“-Modul

Eines der wichtigsten Elemente für emergentes, dezentrales Schwarmverhalten bildet die Prädiktion der Verhalten der anderen Schwarmagenten. Dazu erhält jeder Agent einen „ModelDatacontainer“, der die jeweiligen Modelle speichert. Es gibt Agentenmodelle (siehe Modul„RobotModel“), Fahrzeugmodelle („VehicleModel“) und Funkstationsmodelle („StationModel“). Die ersten beiden Modelle werden als dynamische Mo-

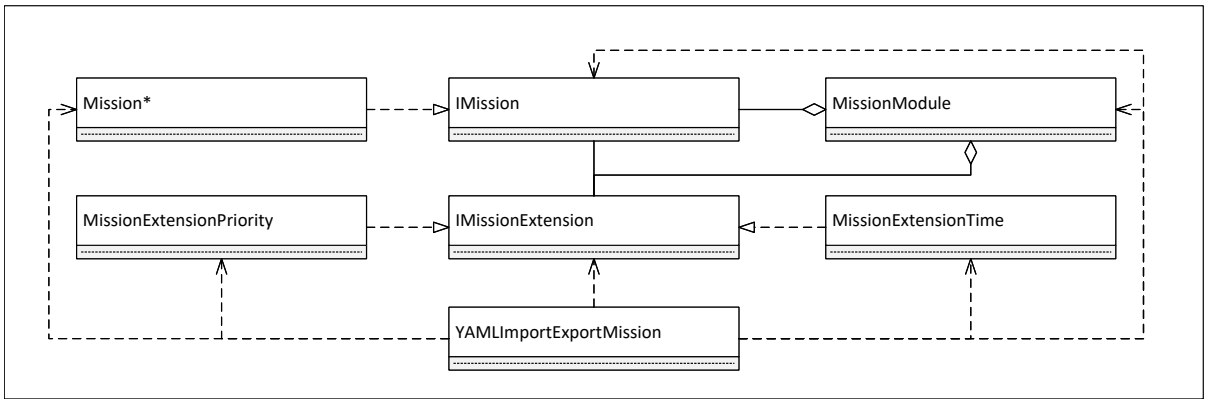


Abbildung 4.25.: Elemente des „mission“-Moduls.

delle betrachtet und erhalten eine Aktualisierungskomponente („VehicleModelUpdateComponent“ und „RobotModelUpdateComponent“), die eine Rücksprungfunktion für Nachrichten der entsprechenden Typen enthalten. Anhand der Nachrichteninformationen werden der tatsächliche Zustand sowie die geplanten Aktionen aktualisiert und Prädiktionen ermöglicht. Dabei besitzen alle Modelle die gemeinsame Schnittstelle „IModel“, welche z. B. Positionsdaten in Form eines „PositionDatacontainer“ umfasst. Die „ModelAliveSenderComponent“ sendet zyklisch eine Typinformationsnachricht in ein Netzwerk, sodass später hinzukommende Knoten die Nachrichten der bereits im Netzwerk vorhandenen Knoten einordnen können und entsprechende Modelle anlegen können.

4.2.19. „mission“-Modul

Die Konzepte und Funktionsweisen des Missionsmanagements sowie der einzelnen Missionen werden detailliert in Kapitel 5 erläutert. Dabei verwaltet das „MissionModule“ eine Liste gleichwertiger Standardmissionen, eine Liste prioritätsbasierter Missionen und eine Liste für zeitlich getaktete Missionen, die mittels eines Arbitrieres verwaltet werden. In welchem dieser drei Modi sich die Missionsverwaltung befindet, kann zur Laufzeit beliebig bestimmt und gewechselt werden. Dabei muss jede Mission die „IMission“-Schnittstelle realisieren, welche von „IPotentialObject“ erbt. Des Weiteren erweitern die „IMissionExtension“-Realisierungen „MissionExtensionPriority“ und „MissionExtensionTime“ die eigentlichen Missionen um Prioritäten oder Zeitpläne. Auch diese Klassen können durch YAML-Serialisierungen im- und exportiert werden.

4.2.20. „sync“-Modul

Das „sync“-Modul enthält eine erweiterte Stoppuhrkomponente („TimeSyncComponent“), die mit einer gegebenen Frequenz den eigenen Zeitstempel im Netzwerk propagiert. Liegt dieser hinter dem eines anderen Knoten zurück, so sendet dieser eine Antwort mit seinem Zeitstempel. Der Zeitstempel wird dann von der initial sendenden Komponente übernommen. Ansonsten übernehmen die Netzwerkteilnehmer den

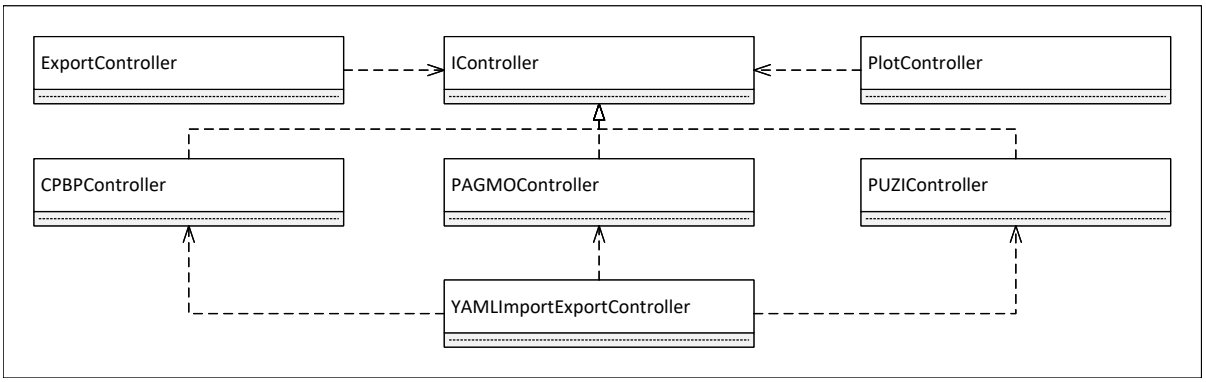


Abbildung 4.26.: Elemente des „controller“-Moduls.

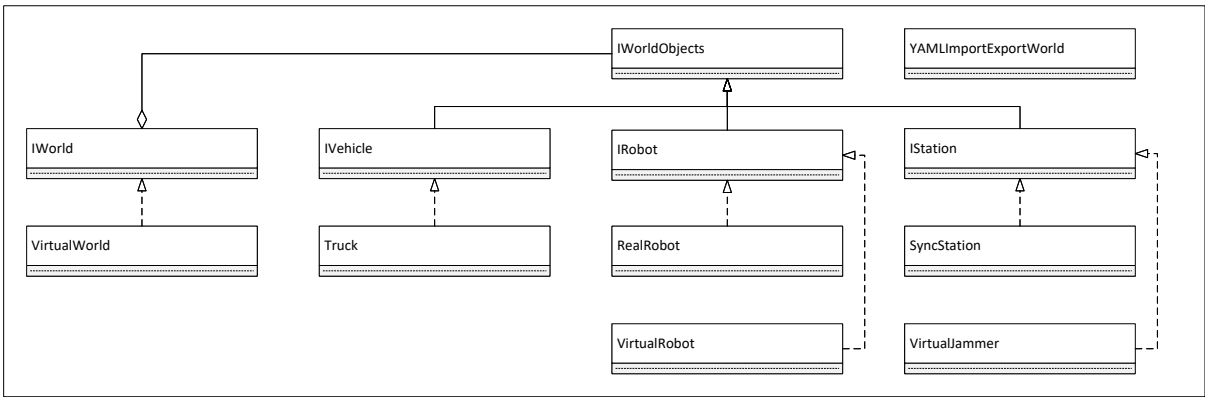


Abbildung 4.27.: Elemente des „world“-Moduls.

gesendeten Zeitstempel unter Beachtung der Netzwerkverzögerung. Somit wird eine synchrone Zeit in einem dezentralen Netzwerk erreicht.

4.2.21. „controller“-Modul

Die verschiedenen Regler und deren Optimierungsbibliotheken werden durch je eine Klasse gekapselt. Diese wiederum realisieren die einheitliche „IController“-Schnittstelle, die einen einfachen Austausch und die Integration weiterer Regler ermöglicht.

4.2.22. „world“-Modul

Das komplexeste Modul ist das „world“-Modul, da es alle einzelnen Bibliotheken und Komponenten zu einer Simulation zusammensetzt. Die Basisklasse einer Simulation oder Ko-Simulation bildet die „IWorld“-Schnittstelle. Sie verwaltet eine Aggregation verschiedener „IWorldObjects“. Dies beinhaltet die Erzeugung der Komponenten, das Laden, Starten und Aktivieren dieser in der korrekten Reihenfolge und das Beenden der Komponenten. Dabei erfolgt das Einlesen eines Szenarios aus einer YAML-Datei, die im Vergleich zur Arbeit Puzicha [141] um folgende Punkte erweitert worden ist:

1. „gatheringpoints“: Diese Auflistung enthält alle Ablageorte für eine Transportmission (siehe Abschnitt 5.11).

2. „goods“: Diese Liste enthält die Positionen der zu transportierenden Pakete (siehe Abschnitt 5.11).

3. „radiostations“: Dies ist die Auflistung aller Funkstationen im Szenario inklusive der Störsender und der Zeitsynchronisationsstationen.

4. „messagebridges“: Diese Auflistung erzeugt Nachrichtenbrücken des gegebenen Typs an der initialen Position, bis sich ein Netzwerkteilnehmer mit derselben Identifizierungsnummer mit dieser verbindet.

Die Dateien haben nun folgendes Format:

Quelltext 4.3: Beispiel einer vollständigen Szenariobeschreibung

```
environment: # Bodengegebenheiten
    - type: "RadialBoundedLinear"
      pos: [50.0, 200.0]
      goal_value: 36.0
      radius: 150.0
radiation: # Rauschniveau
    - type: "ConstantWithNoise"
      goal_value: 0.0
      noise_scale: 10.0
    - type: "RadialBoundedLinear"
      pos:
        - 175.0
        - 325.0
      goal_value: 30.0
      radius: 150.0
staticobstacles: # Statische Hindernisse
    - type: "RectangleQuadratic"
      pos:
        - 100.0
        - 100.0
      goal_value: 600.0
      width:  40.0
      height: 20.0
      radius: 10.0
gatheringpoints: # Ablageorte
    - pos:
      - 200.00
      - 50.00
goods: # Pakete
    - pos:
      - 100.00
      - 75.00
dynamicobstacles: # Dynamische Hindernisse
    - type: "MoveLinear"
      velocity:
        - 2.0
        - 0.0
      rotation_velocity: 0.07
      staticobject:
        type: "RadialBoundedQuadratic"
        pos:
          - 255.0
          - 155.0
        goal_value: 600.0
        radius: 20.0
radiationstaticobstacles: #Netzwerkhindernisse
    - type: "RadiationStaticLineSegment"
      pos_start:
        - 100.0
        - 100.0
      pos_end:
        - 200.0
        - 150.0
      damping: -20.0
radiationdynamicobstacles: #Netzwerkhindernisse
    - type: "RadiationMoveLinear"
      velocity:
        - 3.0
        - 0.0
      rotation_velocity: 0.0
      staticobject:
        type: "RadiationStaticLineSegment"
        pos_start:
          - 50.0
          - 100.0
        pos_end:
          - 150.0
          - 150.0
        damping: -20.0
convoy: # Fahrzeuge
    - type: "CommunicationTruck"
      pos:
        - 200.0
        - 175.0
        - 3.0
      velocity:
        - 0.0
        - 0.0
        - 0.0
robots: # Agenten
    - pos:
        - 200.0
        - 200.0
        - 0.0
radiostations: # Funkstationen
    - pos:
        - 30.0
        - 30.0
        - 0.0
messagebridges: # Nachrichtenbruecken
    - type: "UDP"
      id: 66
      sender_port: 4242
      receiver_port: 4242
      pos:
        - 100.0
        - 100.0
        - 0.0
    - type: "UDP"
      id: 67
      sender_port: 4242
```

```
      receiver_port: 4242
      pos:
        - 110.0
        - 110.0
        - 0.0
    - type: "LORA"
      id: 77
```

```
      port: "COM3"
      pos:
        - 120.0
        - 120.0
        - 0.0
```

Die aktiven Elemente eines Szenarios erben von der „IWorldObject“-Schnittstelle. „IStation“ repräsentiert alle möglichen Formen eines realen Kommunikationsknotens inklusive einer realen Position. Die direkten Realisierungen bilden eine Zeitsynchronisationsstation („SyncStation“) sowie Störsenderstationen („VirtualJammer“), die zeitgleich von „IJammer“-Schnittstelle erben. Weiterhin existiert die übergeordnete Schnittstelle aller bewegten Objekte und Fahrzeuge, die in „IVehicle“ zusammengefasst wird. Ein Kommunikationsfahrzeug zur Koordinierung des Schwarms, welches in der Realität auch eine Rettungskraft sein kann, realisiert nun die Kombination aus einem Fahrzeug und einem Kommunikationsknoten in der Klasse „Truck“. Dieselbe Kombination erzeugt die „IRobot“-Schnittstelle die aus einer Ansammlung aus allen zuvor beschriebenen Komponenten besteht (siehe Quelltext B.6 im Anhang B.4). Mit welchen Realisierungen und in welcher Startreihenfolge die Verweise der „IRobot“-Komponenten gefüllt und aktiviert werden, unterscheidet sich von Agentenkonfiguration zu Agentenkonfiguration. Es existieren zwei mögliche Realisierungen, eine für rein virtuelle Agenten („VirtualRobot“) und eine für reale Agenten („RealRobot“). Ein Weiteres Beispiel einer Funkstation sind die Nachrichtenbrücken („MessageBridge“ und „MessageBridgeLora“) mit denen sich ebenfalls simulierte und reale Roboter verbinden können.

4.2.23. Testung des Autonomiekerns

Um eine hohe Qualität der Implementierung zu erreichen, sind Modultests (Englisch: „Unit tests“) für jede Komponente eingeführt worden. Dazu wird das Grundgerüst „GoogleTest“ verwendet. Nach jeder Änderung oder Erweiterung wird der Quelltext übersetzt und jeder Test ausgeführt, bevor eine neue Version auf die Agenten übertragen wird.

4.2.24. Datenim- und Datenexport

Der Export der Daten bzw. Zustände wird durch einen rekursiven Aufruf der Komponenten erzielt. Dieses Vorgehen nutzt den Vorteil aus, dass alle komplexeren Elemente in der Simulation oder in einer realen Umsetzung aus einer Auflistung oder rekursiven Struktur dieser Komponenten bestehen. Dazu implementiert jede Komponente und jeder Datencontainer die folgenden zwei Funktionen:

Quelltext 4.4: Rekursive Import- und Exportfunktionen

```
#ifdef YAML_SUPPORT
/**
 * Loads the configuration from a file (yaml)
 */
inline virtual bool loadConfiguration(const YAML::Node& node) = 0;
```

```
/**
 * Saves the configuration to a file (yaml)
 */
inline virtual bool saveConfiguration(YAML::Node& node) const = 0;
#endif
```

Diese „saveConfiguration“-Funktion schreibt alle Basisvariablen und ruft untergeordnete Komponenten mit seinem eigenen „node“-Objekt auf. Somit muss keine Komponente ihre übergeordnete Komponente kennen, wodurch ein flexibler Einsatz ermöglicht wird. Nachteilig hingegen ist das vollständige Einlesen einer solcher Struktur, weil einige Komponenten dieselben Datencontainer teilen. Diese werden korrekterweise mehrfach in die Datei geschrieben, sodass auch nur Teilkomponenten korrekt geladen werden können. Jedoch müssen die komplexeren Komponenten beim Einlesen darauf achten, dass die ehemals selben Datencontainer nur noch datengleich sind, aber nicht auf denselben Speicher verweisen. Somit müssen nach dem Einlesen einige Datencontainer entladen werden und die Verweise angepasst werden. Dies erfolgt ebenfalls rekursiv.

4.2.25. ROS und ROS2

Zur Demonstration der Kopplung des Autonomiekerns mit anderen Simulatoren und der Kosimulationsfähigkeit wird die exemplarische Integration des Kerns in das ROS- und ROS2-Grundgerüst erklärt, sodass eine Ko-Simulation mit Gazebo (vgl. [82]) für einen einzelnen Agenten erzielt werden kann (siehe Abschnitt 6.3.1). Als erstes wird die Kommunikationsrichtung vom Autonomiekern zum ROS-Grundgerüst durch eine Realisierung der „IActuator“ Schnittstelle erzeugt. Die „ActuatorROS“- und „ActuatorROS2“-Komponenten werden als zeitbasierte Komponenten ausgelegt und konvertieren mit der gewünschten Frequenz die Stellgrößentrajektorie in eine ROS-Nachricht des Typs „geometry_twist“. Diese wird anschließend unter dem „ROS topic“ „\cmd_vel“ veröffentlicht. Für die Richtung werden je eine Realisierung für die Schnittstelle „IIMU“, „IGPS“, „IOdometry“ und „ILidar“ implementiert. Diese abonnieren die jeweiligen „ROS topics“ der zugehörigen Sensoren und wandeln sie in passende Formate für die Datencontainer um. Die Implementierung dieser Realisierungen bilden eigene nachladbare Bibliotheken, die abhängig vom Vorhanden sein des jeweiligen Gerüsts, gebaut werden und den Autonomiekern optional ergänzen. Dies ist ebenfalls mit anderen Simulatoren möglich und verdeutlicht die Modularität.

4.3. Visualisierung und Interaktion

Die eigentliche Simulation (siehe Kapitel 4.2) und die Berechnung der Aktionen (siehe Kapitel 2) sind vollständig beschrieben. Jedoch soll es möglich sein, schnell und verständlich neue Missionen und Funktionen für die Agenten zu entwickeln und die berechneten Lösungen des Schwarms mit Rettungsteams zu teilen und Anpassungen vorzunehmen. Dazu sind eine Visualisierung des Verhaltens sowie Interaktionsmöglichkeiten unabdingbar. Es sind verschiedene Visualisierungen und Interaktionsmög-

lichkeiten entwickelt worden, die je nach verfügbaren Ressourcen und Echtzeitanspruch eingesetzt werden können.

4.3.1. Konsole und „QT-Framework“

Die einfachste Interaktionsmöglichkeit mit den Agenten kann über ein Konsolenfenster geschehen, in dem Agenten konfiguriert und Daten ausgelesen werden können. Dazu bedarf es jedoch stets einer Eingabe. Dieses Verfahren eignet sich vor allem für die eingebettete Systeme der realen Roboter, damit möglichst wenig Rechenleistung verbraucht wird und diese dem Autonomiekern zur Verfügung steht. Eine etwas bessere Darstellung der Informationen einzelner Agenten ermöglicht die implementierte Schnittstelle zum „QT-Framework“, welches die Visualisierung mittels einfacher Oberflächen auch auf eingebetteten Systemen ermöglicht.

4.3.2. 2D-Grafik

Die am meisten verwendete Visualisierung ist die zweidimensionale Darstellung des gesamten Szenarios. Diese wird mit Hilfe des „Cocos2d-X“-Spielegrundgerüsts umgesetzt. Die grundlegenden Elemente und Verfahren werden aus der Arbeit Puzicha [141] übernommen und um die fehlenden Objekte, wie z. B. die Funkstationen, Pakete, Ablageorte und Ladestationen, erweitert. Ferner sind auch die Datenquellen auf die Datencontainer umgeleitet worden. Unterschiedliche Anzeigemenüs für die Agenten und Stationen werden ergänzt und neben einer Übersichtskarte ist ein Hilfemenü realisiert worden, das den Großteil aller verfügbaren Funktionen aufzeigt (siehe Abbildung 4.28). Bei einer realen Kopplung kann nur ein reduzierter Datensatz der Agenten gezeigt werden, da nur eine beschränkte Menge an Daten in den Modellen und den Nachrichtenbrücken gespeichert ist. Dem gegenüber können nahezu alle Daten der simulierten Agenten angezeigt werden. Diese beinhalten die Position, die Stellgrößen, die Lamport-Uhren des Netzwerks und die Liste der aktiven Missionen. Der Vorteil dieser Visualisierung ist, dass sie ebenfalls sehr wenig Ressourcen benötigt sowie echtzeitnah arbeiten kann und dennoch eine übersichtliche, nutzerfreundliche Visualisierung und Interaktion ermöglicht.

4.3.3. 3D-Grafik

Der Datencontaineransatz ermöglicht allerdings auch die Einbindung des Autonomiekerns in deutlich leistungsfähigere Visualisierungen, ohne die Software des Kerns zu ändern. Exemplarisch wird die Integration in die „CryEngine“ und in „Unity“ beschrieben.

4.3.3.1. CryEngine

Die „CryEngine“ Version 5.6 ist eine hocheffiziente Spiele-„Engine“, geschrieben in C++, und diese benutzt das gleiche Bausystem „CMAKE“, wodurch eine einfache Integration möglich ist. Der Vorteil der dreidimensionalen Darstellung ist, dass ebenfalls

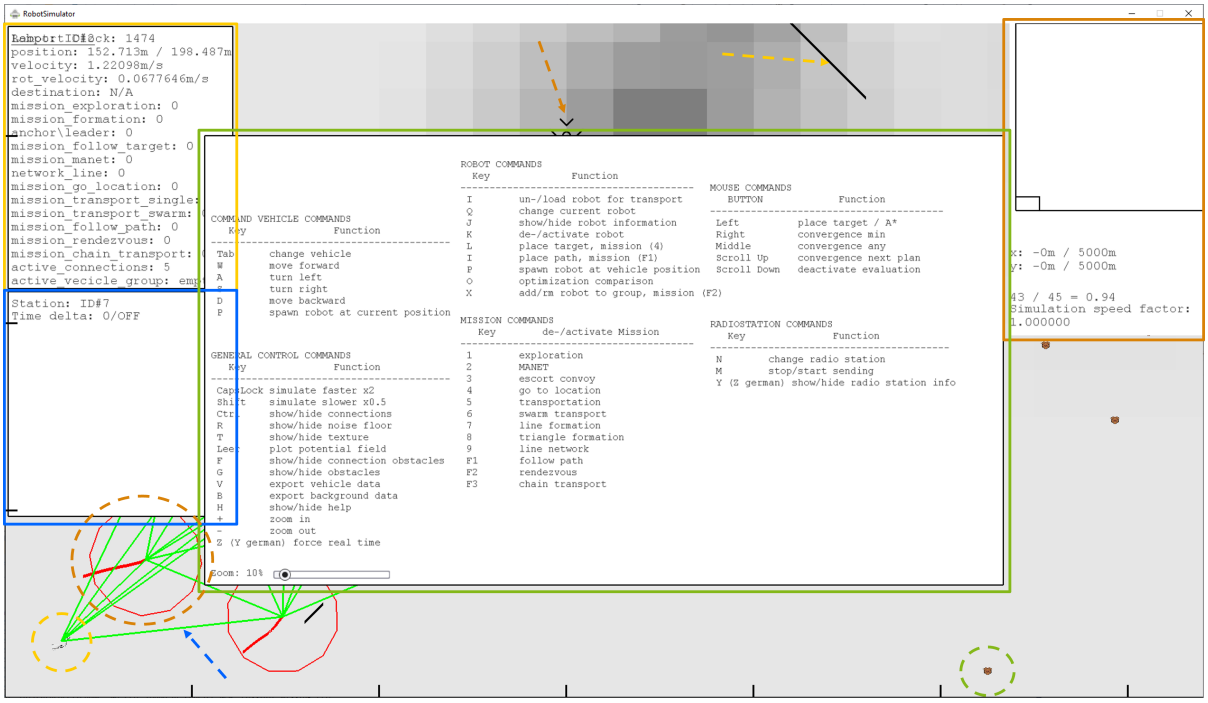


Abbildung 4.28.: Darstellung der neuen zweidimensionalen Benutzeroberfläche. In dem gelben Kasten werden die Daten des aktuell selektierten Agenten gezeigt. Diese umfassen Position, Stellgröße, Missionen und Missionsziele und Netzwerkdaten. Der blaue Kasten zeigt die Daten der in gelb gestrichelt eingekreisten Sendestation, die Netzwerktransaktionen starten kann. Der orange Kasten zeigt eine Karte, den aktuellen Kartenausschnitt und Zeitinformationen. Der grüne Kasten beschreibt fast alle verfügbaren Tastaturkommandos und die Skalierungsstufe. Grün gestrichelt eingekreist ist ein abzuholendes Paket durch einen in orange gestrichelt eingekreisten Agenten. Dieser befindet sich im Zentrum und visualisiert eine rote Trajektorie für die kommenden 30 s. Der blaue gestrichelte Pfeil zeigt auf eine Netzwerkverbindung mit hoher Qualität, die sich von grün über rot zu schwarz färbt. Der gelb gestrichelte Pfeil zeigt auf eine statische Hindernislinie und der orange gestrichelte Pfeil auf ein einen möglichen Paketablageort. Die unterschiedliche Graufärbung des Hintergrunds beschreibt die Bodengegebenheiten- Die schwarze Skaleneinteilung am unteren und linken Rand markiert 100 m pro Strich.

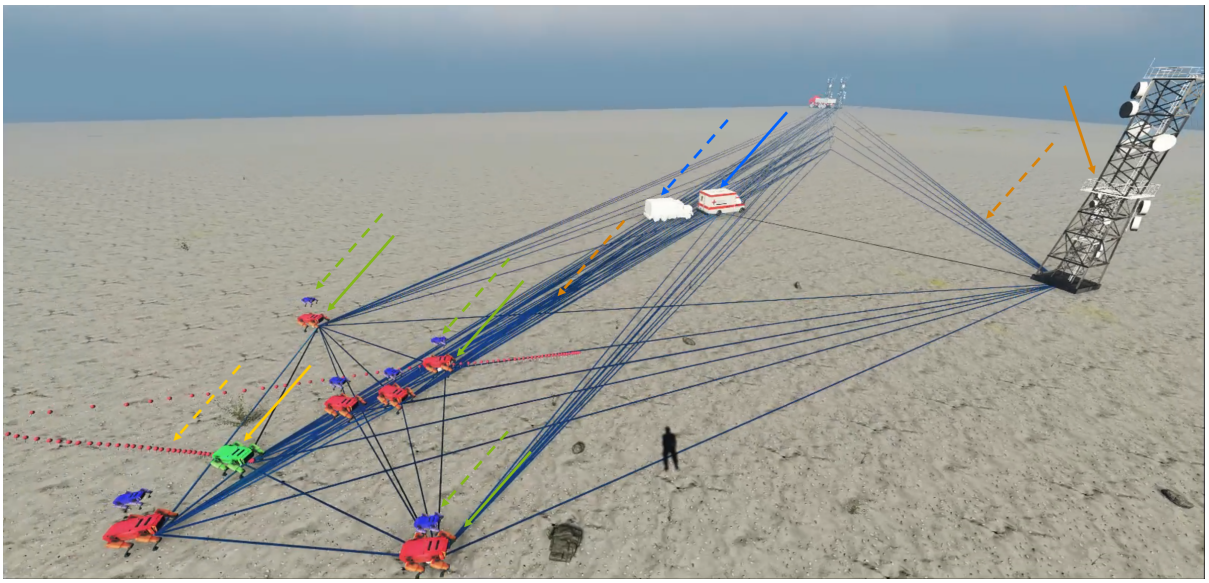


Abbildung 4.29.: Darstellung der dreidimensionalen Ansicht. Der gelbe Pfeil zeigt auf den aktuell selektierten grünen Agenten, dessen Sicht zu sehen ist. Der gestrichelte gelbe Pfeil zeigt dessen geplante Trajektorie für die kommenden 30 s mit Hilfe der Stecknadeln. Die grünen Pfeile zeigt auf andere Agenten, deren Modelle, aus der Sicht des selektierten Agenten, in blau über diesen visualisiert werden und durch grüne gestrichelte Pfeile hervorgehoben werden. Der blaue Pfeil zeigt auf ein Kommando- bzw. Rettungsfahrzeug, dessen Modell, angezeigt durch den gestrichelten Pfeil, sich kurz hinter dem Fahrzeug befindet. Dies resultiert aus der Netzwerkverzögerung. Die orangen Pfeile zeigen auf eine Sendestation im Netzwerk sowie auf die blau dargestellten Netzwerkverbindungen. Die Bodengegebenheiten werden hier nicht in Graustufen, sondern durch die Textur visualisiert.

die simulierten Objekte mit der Modellsicht eines Agenten überlagert werden können (siehe Abbildung 4.29). Dadurch lassen sich Netzwerkverzögerungen gut erkennen und Implementierungsfehler oder Missionsplanungsfehler eher entdecken. Weiterhin wird durch die eingesetzte Unterstützung der virtuellen Realität (kurz VR) ein immersives Erlebnis geboten. Mit diesem können Rettungs- und Einsatzkräfte die Interaktion mit Roboterschwärmen trainieren. Als Hardware für die VR wird die Brille „Quest“ der Firma Oculus verwendet.

4.3.3.2. Andere Spiele-„Engines“ und Visualisierungen

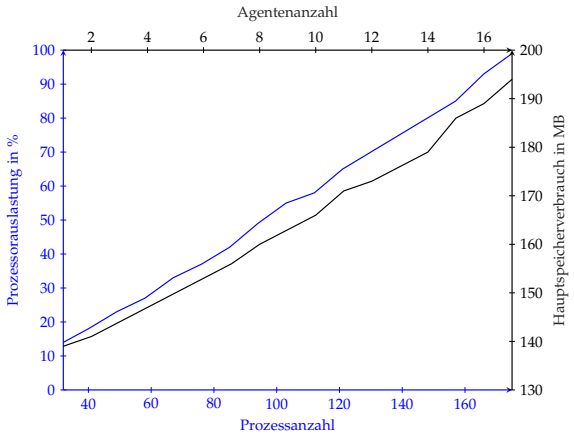
Eine Integration in andere Spiele-„Engines“ kann durch die Generierung einer dynamischen Bibliothek erreicht werden, die die statischen Bibliotheken des Autonomiekerns kapselt. Die dynamische Programmbibliothek (Englisch: „Dynamic Linked Library“, kurz DLL) kann anschließend von anderen Programmiersprachen geladen werden. So bindet die „Unity“ „Engine“ die DLL als in C# geschriebene Erweiterung ein.

4.4. Leistungsanalyse

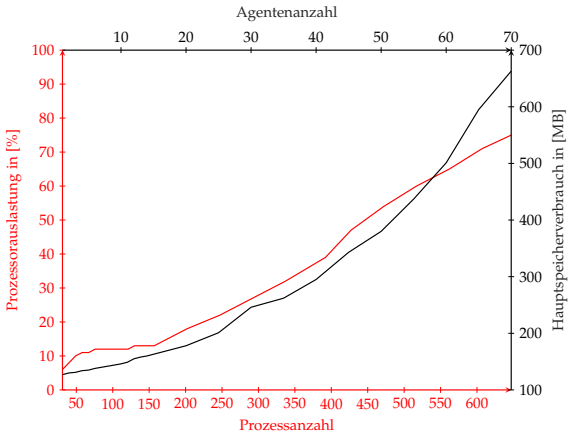
Die Speichereffizienz und die Parallelität des Autonomiekerns und einer gesamten Szenariosimulation wird anhand des Hauptspeicherverbrauchs und der Prozessorauslastung untersucht.

Experiment 4.4.1: *Starte ein Szenario mit einem simulierten Agenten, drei Nachrichtenbrücken, einem Kommunikationsfahrzeug und einem Hindernis innerhalb der zweidimensionalen Visualisierung. Füge daraufhin nach und nach weitere simulierte Agenten hinzu bis der Prozessor vollständig ausgelastet ist. Miss dabei den Hauptspeicherverbrauch, Prozessfadenanzahl und die Prozessorauslastung über 30 min.*

Das Experiment 4.4.1 wird für den „Debug“- und „Release“-Modus ausgeführt, um den Effekt des Quelltextentwurfs, der für übersetzungsseitige Optimierung ausgelegt ist, neben den Leistungsgrenzen mit zu identifizieren. Die Konfiguration des Rechners für die Experimente ist im Anhang B.1.2 aufgeführt. Aus Abbildung 4.30 ist für den unoptimierten Fall ein nahezu linearer Anstieg der Prozessorauslastung und des Hauptspeicherverbrauchs in Abhängigkeit der Agenten- und somit der Prozessanzahl ersichtlich. Dabei erzeugt jeder neue Agent neun Prozesse. Das Basisszenario inklusive der Visualisierung besitzt 31 Prozesse. Der Speicherverbrauch liegt bei etwa 3 MB pro weiteren Agenten. Eine vollständige Auslastung des Prozessors wird bei 17 Agenten erreicht. Die Agenten führen keine komplexe Mission aus, verwalten allerdings die Modelle der Agenten, des Fahrzeugs und der Nachrichtenbrücken und prädizieren deren Verhalten. Dies ist eine geringe Anzahl für einen Schwarm- und Missionssimulator, obgleich sehr viele Optimierungsprobleme gelöst werden. Doch wird nun das Design des Quelltextes durch Quelltextoptimierung zur Bauzeit ausgenutzt, so lassen sich bis zu 70 Agenten simultan auf einem Rechner simulieren. Der begrenzende Faktor ist nicht die Prozessorauslastung, sondern die Zeichnungsfunktion (Englisch: „Render pipeline“) der Visualisierung. Diese schafft es nicht die Linienanzahl zu zeichnen, da



(a) „Debug“-Modus ohne Quelltextoptimierung.



(b) „Release“-Modus mit maximaler Quelltextoptimierung.

Abbildung 4.30.: Darstellung der Beziehungen zwischen Prozessanzahl und Prozessorauslastung sowie Agentenanzahl und Hauptspeicherverbrauch, wobei die Messungen gekoppelt sind und jeder weitere Agent neun weitere Prozesse erzeugt.

jeder Agent zu jedem Agenten, dem Fahrzeug und den Brücken eine Verbindungslinie besitzt. Zudem wird die Zustandstrajektorie jedes Agenten aufgrund der $k = 60$ Zeitschritte (siehe Abschnitt 2.4.3) mit 60 Linien gezeichnet. Allein dies ergibt zusammen bei 70 Agenten 6905 Linien, die stetig aktualisiert werden müssen. Folglich muss für größere Simulationen ein anderer leistungsstärkerer Visualisierer verwendet oder vollständig auf eine Visualisierung verzichtet werden. So können die restlichen Kapazitäten des Prozessors dazu genutzt werden, komplexere Szenarien mit vielen Hindernissen und komplexen Missionen auszuführen. Trotz der Limitierung wird gezeigt, dass der neue Ansatz die Leistungsfähigkeit der vorherigen Simulation mit bis zu 60 Agenten deutlich überschritten hat (vgl. [145]). Weiterhin ist zu erkennen, dass der im „Debug“-Modus scheinbar linear steigende Speicherbedarf mit der Anzahl der Agenten und den damit erzeugten Modellen pro Agenten korrekterweise quadratisch ansteigt, da jeder Agent für jeden Agenten ein Modell im Speicher verwaltet. Trotzdem ist der Hauptspeicherbedarf sehr gering und eignet sich daher auch für eingebettete und ressourcenbeschränkte Systeme.

4.5. Zwischenfazit

Dieses Kapitel beschreibt die internen Konzepte, den Entwurf und die Implementierung des neu entwickelten dezentralen Autonomiekerns für Schwarmagenten. Es wird vor allem die Modularität, Wiederverwendbarkeit und die Laufzeiteffizienz forciert. Das Entwurfsmuster der Fabrik wird an die Anforderungen angepasst, indem es um eine Maschinenkomponente weiter abstrahiert wird. Weiterhin werden Interaktionsmöglichkeiten mit anderen Simulationen bzw. Softwaregrundgerüsten anhand von ROS und ROS2 exemplarisch gezeigt. Aufbauend auf den einzelnen Autonomiekernen der Schwarmagenten lassen sich ganze Szenarien effizient und parallel simulieren. Die Simulation erreicht durch das neue Konzept eine deutlich höhere Leistungsfähig-

keit im Vergleich zu einer vorherigen Realisierung. Zudem lassen sich verschiedene Visualisierer einbinden, um die Daten anzuzeigen, ohne den Quelltext anzupassen. Diese reichen von einfachen Oberflächen und zweidimensionalen Darstellungen, die auch auf eingebetteten System und ARM Prozessoren funktionieren, bis hin zu dreidimensionalen Darstellungen in der virtuellen Realität.

5

Roboterverhaltensmodellierung

Nachdem der Programmaufbau, die Netzwerktechnik zur Kommunikation sowie die zugrundeliegende Regelung und Optimierung behandelt worden sind, beschreibt dieses Kapitel die Verhaltensmodellierung. Nach der Erläuterung der Potenzialfunktionen als mathematische Basis werden systematisch kurz verschiedene Verhaltensstrategien präsentiert, die stets ein Ziel erreichen sollen. Diese Strategien werden als „Mission“ für die Agenten bezeichnet. Zwei der wichtigsten Aspekte sind die Autonomie der Agenten und die Dezentralität der Lösungen, um zentrale Ausfallpunkte in dynamischen Umgebungen zu verhindern. Der Laufzeiteffizienz der Potenzialfunktionsauswertung pro Stützstelle kommt eine zentrale Bedeutung zu, um die Zeitgarantien bis zum Erhalt einer neuen Trajektorie zu fixieren. Die Konfiguration des Rechners für die durchgeführten Experimente ist im Anhang B.1.3 aufgeführt.

5.1. Potenzialfunktionen

Potenzialfunktionen bzw. Kostenfunktionen sind neben den Verhaltensbäumen (vgl. [81]) eine der Modellierungsmethoden für Verhaltensstrategien (vgl. [95, 35, 85, 70, 26] und [141]). Sie werden bereits seit 1998 zur Formationsregelung (vgl. [181]) verwendet. Die Grundidee ist die Verwendung von Funktionen, die erweiterte Zustandsräume, Aktionen oder die Kombination dieser auf skalare Werte abbilden. Dabei wird die Konvention getroffen, dass positive Werte abstoßende, repulsive Verhaltensweisen und negative Werte attraktive, wünschenswerte Verhaltensweisen beschreiben. Sind alle Potenziale und Aggregationen bekannt, so kann durch eine Transformation Null als absolutes Minimum verwendet werden. Anschließend versucht ein Optimierer oder ein künstliches neuronales Netz (kurz KNN, siehe Abschnitt 2.1.3) die Aggregation verschiedener Potenzialfunktionen zu minimieren.
Im Vergleich dazu nutzt die Regelungstechnik mit expliziter Referenz einen Arbeitspunkt, einen Zielzustand oder sogar eine Referenztrajektorie. Diese werden mittels einer Distanzmetrik zum aktuellen Zustand als Regelabweichung verrechnet. Vorteile der Regelungstechnik sind die Stabilitätsbeweise für die gesamten Systeme oder die Arbeitspunkte (vgl. [106, 107]). Jedoch ist die Beweisbarkeit der Systemstabilität auf Kostenfunktionen aus der Menge der Ljapunov-Funktionen beschränkt (vgl. [50]).
Die Wahl der im Allgemeinen gradientenfreien und stützstellenbasierten Optimierer

innerhalb der modellprädiktiven Regelung ermöglicht die Anwendung deutlich größerer Funktionsklassen. Allerdings geschieht dies zu Kosten der analytisch beweisbaren Stabilität bzw. Optimalität der Lösung (siehe Kapitel 2).
Die der Regelung zugrunde liegende Kostenfunktion wird innerhalb der Definition des Minimierungsproblems 2.4.1 angegeben:

$$\mathfrak{L}^{\mathfrak{r}}(T^{\mathfrak{r}}_{\overset{\diamond,\dashv^{\mathfrak{r}}}{\mathbf{x}}_0}, \mathfrak{R}, \mathfrak{t}, \Delta\mathfrak{t}) = \sum_{k=1}^{K} L^{r_k}(\overset{\diamond,\dashv^{\mathfrak{r}}}{\mathbf{x}}_0, \overset{\diamond,\dashv^{r_k}}{\mathbf{x}}_k, \overset{\diamond,\dashv^{r_k}}{\mathbf{u}}_k, \mathcal{R}_k, t_k, \overset{\diamond}{\Delta t}_k) \tag{5.1.1}$$

$T^{\mathfrak{r}}_{\overset{\diamond,\dashv^{\mathfrak{r}}}{\mathbf{x}}_0} = [\overset{\diamond,\dashv^{r_k}}{\mathbf{x}}_k \overset{\diamond,\dashv^{r_k}}{\mathbf{u}}_k]_{k=1,2,\ldots,K} \in (\mathcal{X} \times \mathcal{U})^K$ bezeichnet die Trajektorie eines Agenten $\mathfrak{r}$, beginnend im Startzustand $\overset{\diamond,\dashv^{\mathfrak{r}}}{\mathbf{x}}_0$, über einen diskreten, nicht zwangsweise äquidistant verteilten Zeithorizont der Länge $K \in \mathbb{N}$ mit Zeitschrittvektor $\Delta\mathfrak{t}$ und Zeitvektor $\mathfrak{t}$. Die anderen Agenten $[r_k]_{k=1,2,\ldots,K} = \mathfrak{r} \in \mathfrak{R} = [\mathcal{R}_k]_{k=1,2,\ldots,K}$ im Schwarm $\mathfrak{R}$ werden mit Hilfe der netzwerkbasierten Modelle (siehe Abschnitt 4.2.18) geschätzt. Daher wird im Folgenden die Modellmenge $\hat{\mathfrak{R}}$ verwendet. Weiterhin kann die Modellmenge auch Entitäten enthalten, die keine Agenten, jedoch Netzwerkteilnehmer sind. Beispiele für diese Entitäten sind Synchronisationsstationen oder andere Fahrzeuge, die eskortiert werden sollen.

5.1.1. Geometrische Potenzialfunktionen

Die am meisten verwendeten Potenzialfunktionen sind jene, die Ziele oder Hindernisse im Raum darstellen oder eine geometrische räumliche Ausbreitung über dem Positionsraum eines Agenten bilden. Diese können in zwei Teilfunktionen separiert werden, die anschließend geschachtelt werden. Der eine Teil beschreibt die geometrische Form der Kostenfunktion und der andere Teil den Verlauf der Kosten, wie z. B. linearer oder quadratischer Verlauf (siehe Abschnitt 4.2.4).

Beispiel 5.1.1: *Ein Haus kann als sein Grundriss in einer zweidimensionalen Karte als Rechteck oder Vieleck repräsentiert werden. Dies beschreibt die geometrische Form. Das Verhalten bei einer Annäherung kann über eine weitere Funktion über den Abstand zum Grundriss bestimmt werden. So können die Kosten bei Annäherung beispielsweise quadratisch ansteigen.*

Aufgrund dieser Trennung existieren für die folgenden Formen Vorlagenklassen, die zur Übersetzungszeit berechnet werden:

- Punkt
- Linie
- Rechteck
- Vieleck
- Umgebung

Mit „Umgebung“ ist eine sich über den gesamten Raum ausbreitende Form gemeint, sodass der Abstand zu dieser stets Null beträgt. Als Abstand wird in der Regel die euklidische Norm zum nächsten Punkt auf der Formkontur gewählt. Die Orientierung der Agenten wird bei der Berechnung in der Regel vernachlässigt.

Die Verhaltensfunktionen werden aus der Abschlussarbeit Puzicha [141] übernommen und um eine logarithmische Funktion, die auf dem Abstand $d \in \mathbb{R}_0^+$ definiert ist, ergänzt:

$$l_{\log}(d) = \begin{cases} a_{\log} \cdot \left(-\log(b_{\log} + d) + c_{\log}\right), \text{falls } d \leq r_{\log} \\ 0, \text{sonst} \end{cases} \tag{5.1.2}$$

Dabei beschreibt $a_{\log}$ eine Skalierung der Funktion, $b_{\log} > 0$ einen Versatz, um eine Definitionslücke bei $d = 0$ zu verhindern. $c_{\log}$ definiert eine Verschiebung der Funktion. Die Variablen werden anhand eines gewünschten Zielwertes $e_{\log}$ und eines maximalen Einflussradius $r_{\log}$ der Funktion automatisch bestimmt:

$$c_{\log} = \log(b_{\log} + r_{\log}) \tag{5.1.3}$$

$$a_{\log} = \frac{e_{\log}}{-\log(b_{\log}) + c_{\log}} \tag{5.1.4}$$

Als Versatz wird $b_{\log} = 0{,}5$ als Standard gewählt.

Durch die freie Kombinationsmöglichkeit von fünf Formen und acht Verhaltensfunktionen lassen sich 40 verschiedenen Potenzialfunktionen erzeugen. Jedoch sind nicht alle Kombinationen sinnvoll. Denn eine „Umgebung“ ist in der Regel nur mit konstanten oder zeitabhängigen Kosten zu nutzen.

5.2. Missionsschnittstelle

Die Missionen basieren auf der Klasse der zyklisch zeitbehafteten Zeitkomponente (siehe Komponente 4.2.9 „TimeComponent“) sowie auf der Schnittstelle des Potenzialobjekts (siehe Komponente 4.2.6 „IPotentialobject“) und implementieren deren Spezifikationen. Das Klassendiagramm weist folgenden Aufbau auf (vgl. Abbildung 5.1). Die verwendete Schnittstelle kann durch zwei mögliche Erweiterungen spezialisiert werden. Einerseits existiert die Erweiterung zur Prioritätsmission (siehe Komponente 4.2.19 „MissionExtensionPriority“) und andererseits die Erweiterung zur getakteten Zeitmission (siehe Komponente 4.2.19 „MissionExtensionTime“). Erstere erhalten eine Priorität und es wird stets die Mission mit der höchsten Priorität verarbeitet. Die getaktete Zeitmission besitzt einen Start- und Endzeitpunkt, anhand derer der Missionsarbitrierer (Englisch: „Scheduler“) die Mission erzeugt, aktiviert, deaktiviert und löscht. Allgemein kapselt jede Missionserweiterung (siehe Komponente 4.2.19 „IMissionExtension“) ein Objekt der Missionsschnittstelle. Die Vererbungshierarchie wird in Abbildung 5.2 dargestellt.

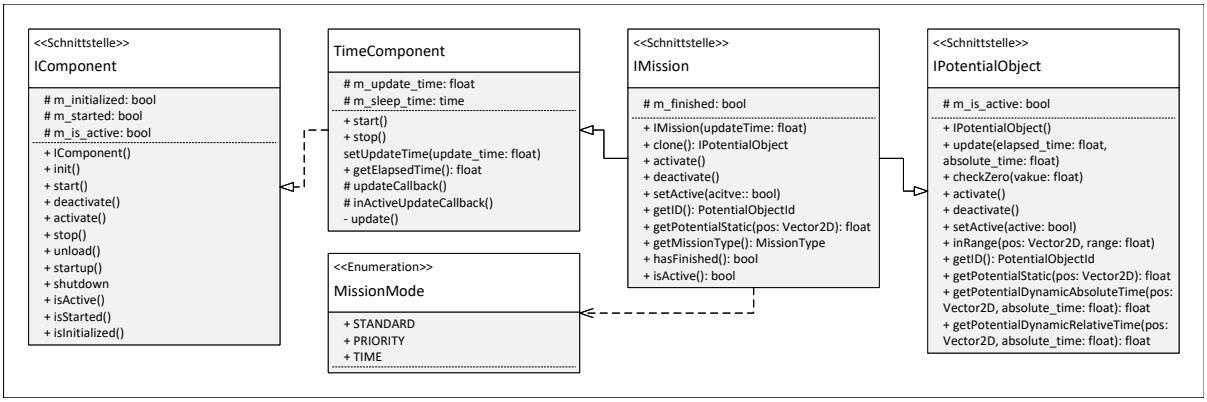


Abbildung 5.1.: Klassendiagramm der Missionsschnittstelle (siehe Komponente 4.2.19 „IMission“).

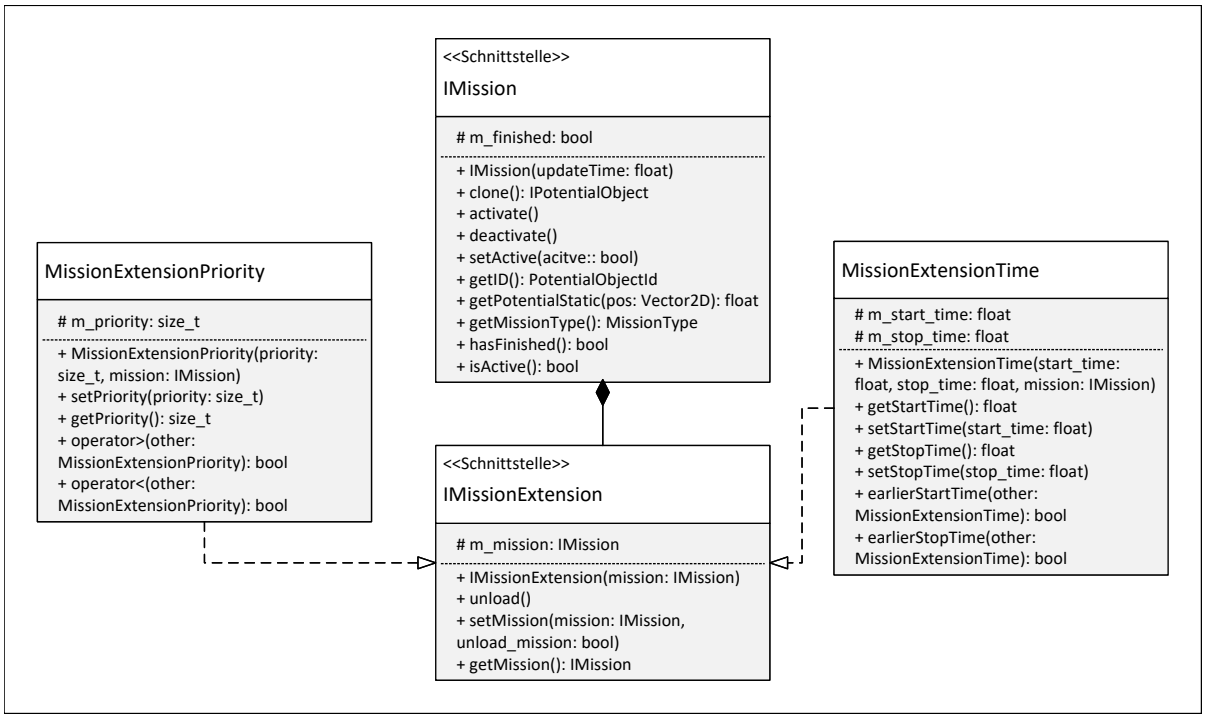


Abbildung 5.2.: Klassendiagramm der Missionserweiterung (siehe Komponente 4.2.19 „IMissionExtension“).

5.3. Missionsbehandlungsmodul

Die Klasse Missionsbehandlungsmodul (siehe Komponente 4.2.19 „MissionModule“) bildet eine wesentliche Komponente des gesamten Autonomiesystems der Agenten. Sie erbt von dem Potenzialobjekt (siehe Komponente 4.2.6 „IPotentialObject“), der Zeitkomponente (siehe Komponente 4.2.9 „TimeComponent“) und der Schnittstelle Nachrichtenbehandlung (siehe Komponente 4.2.13 „MSGHandler“). Die Vererbung der ersten Klasse ermöglicht es das Modul als eine Potenzialfunktion an den Regler weiterzugeben, ohne dass er die spezielle Struktur kennen muss. Die periodische Zeitkomponente dient als Rücksprungfunktion für den Missionsarbitrierer, der je nach gewähltem Modus - „Standard“, „Priorität“ oder „Zeit“ - die Missionen auf ihren Abschluss, ihre zeitliche Gültigkeit oder die aktuelle Priorität prüft. Dadurch lässt sich die Laufzeit für die Berechnung des aktuellen Potenzials auf das Minimum der aktuell aktiven Missionen reduzieren. Dies ist entscheidend, da das Potenzial für jeden Stütz- bzw. Abtastpunkt des Optimierers berechnet werden muss. Durch die Implementierung der Nachrichtenbehandlungsschnittstelle („MSGHandler“, siehe Abschnitt 4.2.13) lässt sich das Modul als Rücksprungroutine für empfangene Nachrichten des Kommunikationsmodul (siehe Komponente 4.2.16 „ICommunicationComponent“) registrieren. Somit kann dieses Modul eigenständig Missionen erstellen und löschen, sobald die entsprechenden Nachrichten empfangen worden sind. Dafür benötigt es jedoch den Zugriff auf alle Datencontainer, da die Missionen alle verschiedenen Datenkombinationen benötigen können.

5.4. Mission: Exploration

Die Explorationsmission dient dem Zweck eine unbekannte Gegend zu erkunden und dabei möglichst weder das eigene noch das von anderen Agenten beobachtete Gebiet erneut zu durchfahren. Um eine zentrale Koordination der Trajektorien zu verhindern, wird ein zufallsbasierter Ansatz gewählt, bei dem jede vorherige Position mit einem positiven, repulsiven Potenzial belegt wird. Dies baut auf der Idee aus der Masterarbeit Puzicha [141] auf. Dazu kann als Kegelfunktion eine lineare oder quadratische Funktion gewählt werden, die mit dem Abstand vom Beobachtungspunkt $\overset{\diamond,\dashv}{\mathbf{x}}$ bis hin zu einem Punkt auf dem Rand des durch Sensoren abgedeckten Erfassungskreises das Nullpotenzial erreicht. Weiterhin können diese Beobachtungspunkte zeitlich gewichtet werden, da dynamische Umgebungen eine zyklische Neuerfassung der Umgebung erfordern. Diese zeitliche Gewichtung begünstigt ebenfalls, dass nur eine gewisse Anzahl an Beobachtungspunkten gespeichert werden müssen, bis deren Potenzial keinen wesentlichen Einfluss mehr aufweisen. Somit kann der benötigte Speicherplatz als auch die Durchlaufzeit der Funktion begrenzt werden.

5.4.1. Abhängigkeiten

Diese Funktion benötigt zum einen den Zugriff auf den eigenen Zustand $\overset{\diamond,\dashv}{\mathbf{x}}$ in Form des Positionsdatencontainers. Zum anderen wird der Modelldatencontainer $\hat{\mathfrak{R}}$ benö-

tigt, um Zugriff auf die Positionsdaten der anderen Agenten $\hat{\mathfrak{r}} \in \hat{\mathfrak{R}}$ zu erhalten. Ergänzend wird der Sensordatencontainer benötigt, um die Reichweite des erkundeten Gebietes mittels Sensors $\overset{\ltimes}{\mathbf{s}}$ zu bestimmen.

5.4.2. Funktionsweise und Formalisierung

Die Mission arbeitet zyklisch auf Basis einer Aktualisierungszeit mit Periode $t^{\dashrightarrow}$. Zu jedem dieser Zeitpunkte wird für jedes Modell $\hat{r} \in \hat{\mathcal{R}}_k$ in dessen zugehörige FIFO-Liste $\mathbf{q}^{\hat{r}} =< \mathfrak{p}_1^{\hat{r}}, \mathfrak{p}_2^{\hat{r}}, \ldots, \mathfrak{p}_N^{\hat{r}} >$ (Englisch: „First In First Out“, kurz FIFO) ein Potenzialobjekt $\mathfrak{p}^{\hat{r}}$ eingefügt. $\mathfrak{p}^{\hat{r}}$ verwendet den aktuellen Zustand $\overset{\diamond,\dashv^{\hat{r}}}{\mathbf{x}}(t^{\dashrightarrow})$ als zentrale Stützstelle und die Reichweite $r(\overset{\ltimes^{\hat{r}}}{\mathbf{s}})$, um ein repulsives, radiales Potenzial mit Radius $r(\overset{\ltimes^{\hat{r}}}{\mathbf{s}})$ und maximalem Potenzial p^+ an dieser Position zu erzeugen. Die FIFO-Liste wird in ihrer Kapazität auf N beschränkt, sodass nach N Aktualisierungszeiten das erste Element entfernt und alle weiteren Indizes $n' = n - 1$ verschoben werden. Damit die zeitliche Genauigkeit der Explorationsinformation widergespiegelt wird, werden die Potenzialfunktionen $\mathfrak{p}_n^{\hat{r}}$ mit einer Gewichtungsfunktion $w(n)$ gewichtet. Dadurch verblasst der Einfluss der Funktion auf das Roboterverhalten mit der Zeit.

Beispiel 5.4.1: *$w(n)$ kann linear mit $w(n) = \frac{n}{N}$, hyperbolisch mit $w(n) = \frac{1}{N-n+1}$ oder exponentiell $w(n) = x^{-N+n}$ mit $x > 1$ gewählt werden.*

Die vollständig formale Beschreibung ist demnach:

$$\mathfrak{L}^{\mathfrak{r}}(T^{\mathfrak{r}}_{\overset{\diamond,\dashv^{\mathfrak{r}}}{\mathbf{x}}_0}, \hat{\mathfrak{R}}, \mathfrak{t}, \Delta\mathfrak{t}) = \sum_{k=1}^{K} L^{r_k}_{\text{EXPLORE}}(\overset{\diamond,\dashv^{\mathfrak{r}}}{\mathbf{x}}_0, \overset{\diamond,\dashv^{r_k}}{\mathbf{x}}_k, \overset{\diamond,\dashv^{r_k}}{\mathbf{u}}_k, \hat{\mathcal{R}}_k, t_k, \overset{\diamond}{\Delta t}_k) \tag{5.4.1}$$

$$L^{r_k}_{\text{EXPLORE}}(\overset{\diamond,\dashv^{\mathfrak{r}}}{\mathbf{x}}_0, \overset{\diamond,\dashv^{r_k}}{\mathbf{x}}_k, \overset{\diamond,\dashv^{r_k}}{\mathbf{u}}_k, \hat{\mathcal{R}}_k, t_k, \overset{\diamond}{\Delta t}_k) = \sum_{\hat{r} \in \hat{\mathcal{R}}_k} \sum_{n=1}^{N} \mathfrak{p}_n^{\hat{r}}(\overset{\diamond,\dashv^{r_k}}{\mathbf{x}}_k) \cdot w(n) \tag{5.4.2}$$

$$\mathfrak{p}_n^{\hat{r}}(\overset{\diamond,\dashv^{r_k}}{\mathbf{x}}_k) = \begin{cases} \frac{p^+}{r(\overset{\ltimes^{\hat{r}}}{\mathbf{s}})^2} \cdot \left(r(\overset{\ltimes^{\hat{r}}}{\mathbf{s}}) - \| \overset{\diamond,\dashv^{r_k}}{\mathbf{x}}_k - \overset{\diamond,\dashv^{\hat{r}}}{\mathbf{x}}_n \| \right)^2 & \text{, falls } r(\overset{\ltimes^{\hat{r}}}{\mathbf{s}}) \geq \| \overset{\diamond,\dashv^{r_k}}{\mathbf{x}}_k - \overset{\diamond,\dashv^{\hat{r}}}{\mathbf{x}}_n \| \\ 0 & \text{, sonst} \end{cases} \tag{5.4.3}$$

Mittels des maximalen Potenzials p^+ kann die Funktion skaliert werden. Zur Realisierung einer konstanten und möglichst geringen Laufzeit sowie einer starken Parallelsierbarkeit, werden Ringpuffer mit konstanter Größe N erzeugt. Diese können ideal auf einer GPU oder CPU parallelisiert werden und arbeiten ohne Abhängigkeiten. Initialisiert werden sie mit konstanten Nullpotenzialen $\mathfrak{p}_n^{\hat{r}} = 0$. Die Umsetzung der Funktion wird in Algorithmus 5.1 aufgezeigt.

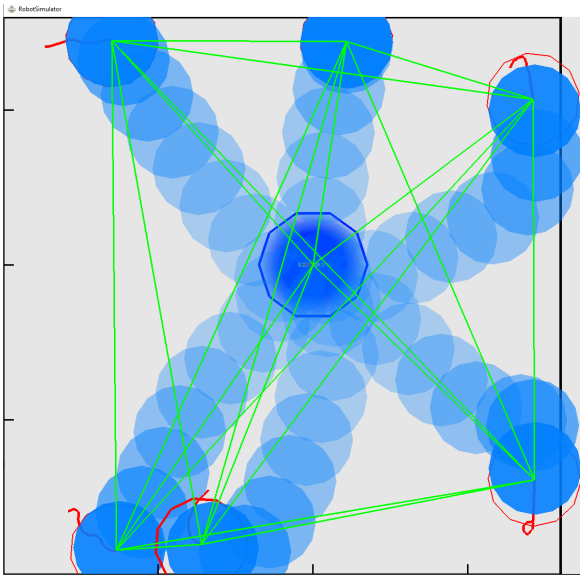
5.4.3. Erzeugte Schnittstellen und Daten

Diese Mission bietet keine weiteren Interaktionen oder Schnittstellen. Sie kann als Standard- oder Leerlaufmission angesehen werden. Als Daten wird eine Spur (Eng-

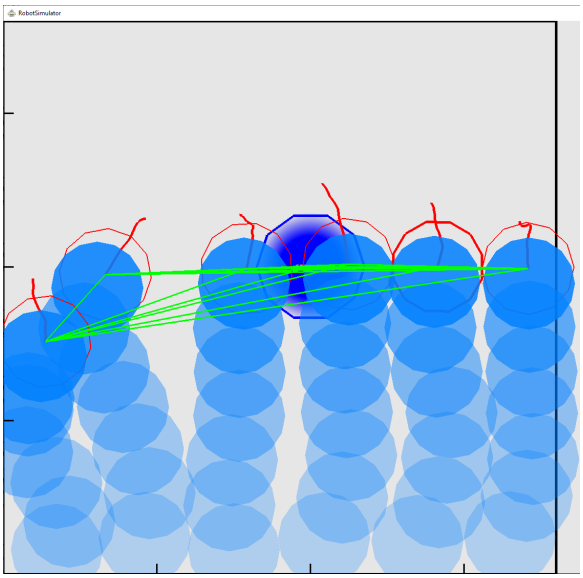
Algorithmus 5.1.: Mission *Exploration*

Voraussetzung: $\hat{\mathfrak{R}}$ Modelldaten, ${}^{\diamond,\dashv^{r_k}}\mathbf{x}_k$ aktuelle Position des Agenten r_k, Δ_t Zeit seit dem letzten Update, N Anzahl an Potenzialfunktionen bzw. Beobachtungspunkten, $w(n)$ Gewichtungsfunktion, t_k aktueller Zeitpunkt

1: **function** INIT(${}^{\diamond,\dashv^{r_k}}\mathbf{x}_k$)
2: **for all** $\hat{r} \in \hat{\mathcal{R}}_k$ **do**
3: CREATERINGBUFFER($\mathbf{q}^{\hat{r}}$) ▷ gut parallelisierbar
4: **for all** $n \in N$ **do** ▷ gut parallelisierbar
5: **for all** $\hat{r} \in \hat{\mathcal{R}}_k$ **do**
6: $\mathfrak{p}_n^{\hat{r}} \leftarrow 0$
7: $\mathbf{q}^{\hat{r}}(n) \leftarrow \mathfrak{p}_n^{\hat{r}}$
8: PRECALCULATE($w(n)$)
9: **function** GETPOTENTIAL(${}^{\diamond,\dashv^{r_k}}\mathbf{x}_k$)
10: $s \leftarrow 0$
11: **for all** $\hat{r} \in \hat{\mathcal{R}}_k$ **do** ▷ gut parallelisierbar
12: **for all** $n \in N$ **do**
13: $\mathfrak{p}_n^{\hat{r}} \leftarrow \mathbf{q}^{\hat{r}}(n)$
14: $s \leftarrow s + w(n) \cdot \mathfrak{p}_n^{\hat{r}}({}^{\diamond,\dashv^{r_k}}\mathbf{x}_k)$
return s
15: **function** UPDATECALLBACK(t_k, Δ_t)
16: **for all** $\hat{r} \in \hat{\mathcal{R}}_k$ **do** ▷ gut parallelisierbar
17: ${}^{\diamond,\dashv^{\hat{r}}}\mathbf{x}_N \leftarrow$GETSTATE($\hat{r}$)
18: ${}^{\ltimes^{\hat{r}}}\mathbf{s} \leftarrow$ GETSENSOR($\hat{r}$)
19: $r({}^{\ltimes^{\hat{r}}}\mathbf{s}) \leftarrow$ GETSENSORRADIUS(${}^{\ltimes^{\hat{r}}}\mathbf{s}$)
20: $\mathfrak{p}_N^{\hat{r}} \leftarrow$ CREATEPOTENTIAL(${}^{\diamond,\dashv^{\hat{r}}}\mathbf{x}_N, r({}^{\ltimes^{\hat{r}}}\mathbf{s})$)
21: POPBACK($\mathbf{q}^{\hat{r}}(1)$)
22: PUSHFRONT($\mathbf{q}^{\hat{r}}(N)$)$\leftarrow \mathfrak{p}_N^{\hat{r}}$



(a) Explorationsmission mit zentrierter Startposition der Agenten



(b) Explorationsmission mit optimierter Startposition der Agenten

Abbildung 5.3.: Vergleich der Explorationsmission mit verschiedenen Startkonfigurationen des Schwarms. Die blaue verblassende Spur kennzeichnet die erstellten Potenzialobjekte, die mit der Zeit ihren Einfluss verlieren, um die Alterung der Information widerzuspiegeln.

Tabelle 5.1.: Benötigte Zeit zur dezentralen, zufälligen Erkundung eines 360 m · 360 m großen Gebietes.

Stichprobengröße	Mittelwert	Std.	Min.	Max.
10	243,3648 s	34,2751 s	197,1124 s	283,8089 s

lisch: „Trace“), bestehend aus vergangenen Zuständen bzw. Positionen und deren Potenzialobjekten, gespeichert.

5.4.4. Experimente

Zur Untersuchung der Funktionsfähigkeit wird experimentell die Dauer zur Erkundung eines Gebietes mittels eines Agentenschwarms betrachtet.

Experiment 5.4.1: *Ein Agentenschwarm bestehend aus sechs Agenten soll ein quadratisches Gebiet von* 360 m · 360 m *erkunden.*

Der Sensorradius der Agenten wird mit $r(\overset{\ltimes^{\hat{r}}}{\mathbf{s}}) = 30\,\mathrm{m}$ gewählt. Die Maximalgeschwindigkeit der Agenten ist $3\,\mathrm{m\,s^{-1}}$. Ferner betragen die Ringpuffergröße $N = 15$ und die Skalierung $p^+ = 100$. $w(n)$ wird hyperbolisch gewählt mit: $w(n) = \frac{1}{1+0{,}35(N-n)}$. Die Aktualisierungsrate, mit der neue Potenzialobjekte erstellt werden, beträgt $t = 7\,\mathrm{s}$, sodass insgesamt 105 s abgedeckt werden. Die benötigte Laufzeit für die Absolvierung der Mission mit einem zentralen Startpunkt (siehe blaue Spur in Abbildung 5.3a) beträgt 243,36 s im Mittel (siehe Tabelle 5.1).

5.4.5. Analyse

Die Laufzeit wird nun mit der optimalen Zeit verglichen. Die optimale Zeit ergibt sich aus der maximalen Fahrzeuggeschwindigkeit, dem Sensorradius und der initialen Positionierung der Agenten. Ein Agent deckt eine Breite von 60 m ab, sodass sich das Gebiet in sechs Streifen zu je 360 m Länge unterteilen lässt (siehe Abbildung 5.3b). Diese können in 120 s zurückgelegt werden und bilden somit die minimale Zeit bei optimaler Verteilung der Fahrzeuge. Dem gegenüber startet das Experiment mit einer Häufung der Agenten am Mittelpunkt, dabei sind diese maximal 6 m vom Mittelpunkt entfernt (siehe Abbildung 5.3a). Es gilt, dass die Erkundung etwa doppelt so viel Zeit beansprucht wie die optimale Lösung. Wird eine optimale Startverteilung ebenfalls für die Mission gewählt, so wird die gegenüberliegende Seite tatsächlich im Mittel innerhalb von 131,512 009 s erreicht. Jedoch entstehen Lücken im observierten Gebiet (siehe Abbildung 5.3b), sodass eine vollständige Erkundung erst nach 222,789 719 s eintritt. Folglich gilt, dass trotz einer vollständigen Dezentralität ohne Koordination eine gute Laufzeit erzielt wird.

5.4.6. Zusammenfassung und Referenzen

Die Mission bildet eine laufzeiteffiziente, dezentrale Verhaltensstrategie, um mit einer Gruppe von Agenten ein unbekanntes, dynamisches Gebiet zu erkunden. Dabei ist das Verhalten der Agenten für die Teilnehmer des Schwarms prädizierbar, wohingegen es nach außen zufällig und unkoordiniert wirkt. Dies kann Vorteile für Überwachungs-, Spionage- und Militäroperationen bieten. Durch die Wahl der Datenstruktur und der Formalisierung wird eine Laufzeit von $O(N \cdot |\hat{\Re}|)$ erzielt. Dabei beschreibt $|\hat{\Re}|$ die Anzahl an Agenten im Schwarm bzw. die Menge der aktiv verwalteten Agentenmodelle. Ein lückenlose Exploration kann durch die Kombination mit einer Formationsmission (siehe Abschnitt 5.10) erzielt werden.

5.5. Mission: Mobiles ad hoc Netzwerk (MANET)

In einem Katastrophengebiet fällt oft die statisch installierte Kommunikationsinfrastruktur aus. Diese Infrastruktur ist allerdings für die Koordination der Such- und Rettungsaktionen zwingend erforderlich. Daher werden mobile Kommunikationsknoten aufgestellt, um diese wieder aufzubauen. Jedoch ist die Positionierung der Knoten ein Optimierungsproblem, das aufgrund der dynamischen Informationsverteilung ebenfalls dynamisch sein kann. Somit soll ein Agentenschwarm, ausgerüstet mit mobilen Kommunikationsmodulen, den Aufbau und die dynamische Verteilung der Knoten übernehmen. Aufgrund der Ausfallwahrscheinlichkeit in Katastrophengebieten wird erneut auf eine dezentrale Architektur geachtet.

5.5.1. Abhängigkeiten

Diese Mission benötigt einen Zugriff auf den Netzwerkdatencontainer, um den SNR und den RSSI jeder Verbindung zu verarbeiten. Dieser Zugriff ist ausschließlich lesend

und beeinflusst das Netzwerk nur durch die neu angestrebte Position des Netzwerkmoduls.

5.5.2. Funktionsweise und Formalisierung

Der Aufbau von stern- oder ringförmigen mobilen ad hoc Netzwerken (MANET) erfordert eine erweiterte Verwendung von Potenzialfunktionen. Einerseits gibt es eine diskrete Verbindungsqualität zwischen zwei Agenten r_k und $\hat{r}_k$ mit $\gamma^{\leftrightsquigarrow \hat{r}_k}(\overset{\diamond,\dashv^{r_k}}{\mathbf{x}}_k) \in [0,1]$, die auf dem Empfangsfeldstärkenindex (kurz RSSI) und dem Signalrauschabstand (kurz SNR) (siehe Tabelle 3.2) des drahtlosen Netzwerks basiert. Andererseits wird der Abstand $\left\| \overset{\diamond,\dashv^{r_k}}{\mathbf{x}}_k - \overset{\diamond,\dashv^{\hat{r}_k}}{\mathbf{x}}_k \right\| = d^{\hat{r}_k}(\overset{\diamond,\dashv^{r_k}}{\mathbf{x}}_k) \in \mathbb{R}_0^+$ zwischen r_k und $\hat{r}_k$ betrachtet. Ein großer Abstand ist erwünscht, wenn die Verbindungsqualität gut ist, andernfalls ist der Abstand ein Malus, der reduziert werden muss, z.B. wenn die Verbindung abbricht. Dieses Verhalten spiegelt sich in l_{MANET} wider und kann mit κ und ζ eingestellt werden. Dies muss für jeden geplanten Zeitschritt $k \in K$ und jeden Kommunikationspartner $\hat{r} \in \hat{\mathcal{R}}_k$ ausgewertet werden, um die MANET Mission zu bilden. Um stern- und ringförmige Strukturen anstelle von Dreiecken oder Clustern zu erzielen, ist der Logarithmus $\ln(d)$ in l_{MANET} notwendig, da er viele Verbindungen mit mittlerem Abstand gegenüber einigen großen und vielen kleinen Abständen bevorzugt (vgl. [141]).

$$\mathfrak{L}^{\mathfrak{r}}(T^{\mathfrak{r}}_{\overset{\diamond,\dashv^{\mathfrak{r}}}{\mathbf{x}}_0}, \hat{\mathfrak{R}}, \mathfrak{t}, \Delta\mathfrak{t}) = \sum_{k=1}^{K} L^{r_k}_{\text{MANET}}(\overset{\diamond,\dashv^{\mathfrak{r}}}{\mathbf{x}}_0, \overset{\diamond,\dashv^{r_k}}{\mathbf{x}}_k, \overset{\diamond,\dashv^{r_k}}{\mathbf{u}}_k, \hat{\mathcal{R}}_k, t_k, \overset{\diamond}{\Delta t}_k) \tag{5.5.1}$$

$$L^{r_k}_{\text{MANET}}(\overset{\diamond,\dashv^{\mathfrak{r}}}{\mathbf{x}}_0, \overset{\diamond,\dashv^{r_k}}{\mathbf{x}}_k, \overset{\diamond,\dashv^{r_k}}{\mathbf{u}}_k, \hat{\mathcal{R}}_k, t_k, \overset{\diamond}{\Delta t}_k) = \sum_{\hat{r}_k \in \hat{\mathcal{R}}_k} l_{\text{MANET}}(\gamma^{\leftrightsquigarrow \hat{r}_k}(\overset{\diamond,\dashv^{r_k}}{\mathbf{x}}_k), d^{\hat{r}_k}(\overset{\diamond,\dashv^{r_k}}{\mathbf{x}}_k)) \tag{5.5.2}$$

$$l_{\text{MANET}}(\gamma^{\leftrightsquigarrow \hat{r}_k}(\overset{\diamond,\dashv^{r_k}}{\mathbf{x}}_k), d^{\hat{r}_k}(\overset{\diamond,\dashv^{r_k}}{\mathbf{x}}_k)) = \tag{5.5.3}$$

$$\begin{cases} d^{\hat{r}_k}(\overset{\diamond,\dashv^{r_k}}{\mathbf{x}}_k) \cdot \kappa & , \text{ falls } \gamma^{\leftrightsquigarrow \hat{r}_k}(\overset{\diamond,\dashv^{r_k}}{\mathbf{x}}_k) = 0 \\ -(\gamma^{\leftrightsquigarrow \hat{r}_k}(\overset{\diamond,\dashv^{r_k}}{\mathbf{x}}_k) \cdot \zeta + \ln(d^{\hat{r}_k}(\overset{\diamond,\dashv^{r_k}}{\mathbf{x}}_k)) \cdot \kappa & , \text{ sonst} \end{cases}$$

$$\text{mit } \kappa, \zeta > 0,\ \gamma^{\leftrightsquigarrow \hat{r}_k}(\overset{\diamond,\dashv^{r_k}}{\mathbf{x}}_k) \in [0,1],\ d^{\hat{r}_k}(\overset{\diamond,\dashv^{r_k}}{\mathbf{x}}_k) \in \mathbb{R}^+$$

5.5.3. Erzeugte Schnittstellen und Daten

Diese Mission bietet ebenfalls keine weiteren Schnittstellen und Daten zur Weiterverarbeitung an.

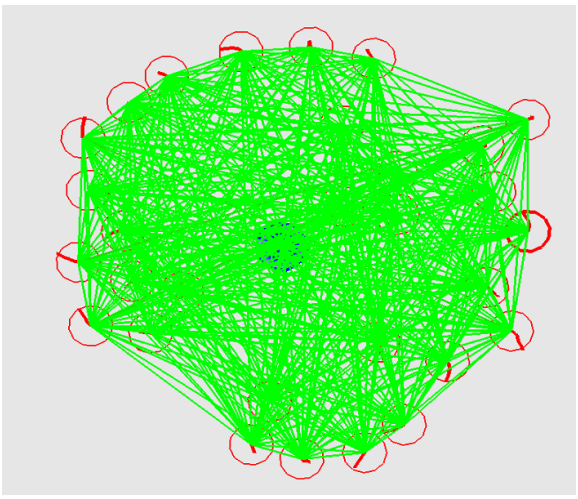
5.5.4. Experimente

Die Untersuchung der Funktionsfähigkeit erfolgt anhand der aufgebauten Strukturen, da eine optimale Verteilung eines Agentenschwarms mit 60 Agenten nicht gut berechenbar ist.

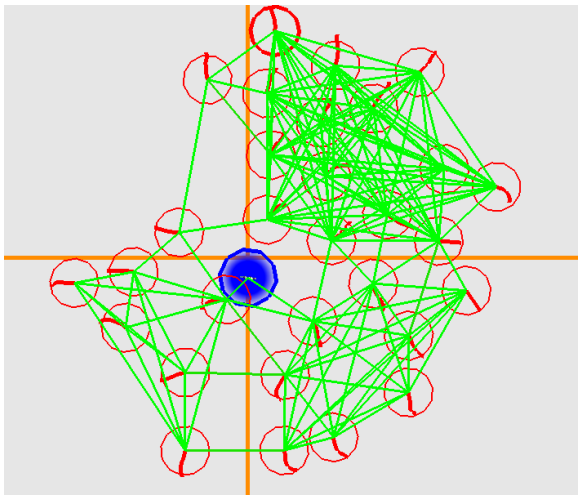
Algorithmus 5.2.: Mission *MANET*

Voraussetzung: $\hat{\mathfrak{R}}$ Modelldaten, ${}^{\diamond,\dashv^{r_k}}\mathbf{x}_k$ aktuelle Position des Agenten r_k, Δ_t Zeit seit dem letzten Update, $\mathbf{Q}_\gamma$ Tupelliste mit Qualitäts- und Positionsdaten, $\kappa, \zeta > 0$

1: **function** INIT()
2: $\quad \mathbf{Q}_\gamma \leftarrow \emptyset$
3: **function** GETPOTENTIAL(${}^{\diamond,\dashv^{r_k}}\mathbf{x}_k$)
4: $\quad s \leftarrow 0$
5: $\quad$ **for all** $< \gamma^{\overleftrightarrow{\hat{r}}}, {}^{\diamond,\dashv^{\hat{r}}}\mathbf{x}_k > \in \mathbf{Q}_\gamma$ **do** ▷ gut parallelisierbar
6: $\quad\quad d^{\hat{r}} \leftarrow \left\| {}^{\diamond,\dashv^{r}}\mathbf{x}_k - {}^{\diamond,\dashv^{\hat{r}}}\mathbf{x}_k \right\|$
7: $\quad\quad$ **if** $\gamma^{\overleftrightarrow{\hat{r}}} > 0$ **then**
8: $\quad\quad\quad s = s - (\gamma^{\overleftrightarrow{\hat{r}}} \cdot \zeta + \ln(d^{\hat{r}}) \cdot \kappa)$
9: $\quad\quad$ **else**
10: $\quad\quad\quad s = s + d^{\hat{r}} \cdot \kappa$
$\quad$ **return** s
11: **function** UPDATECALLBACK(t_k, Δ_t)
12: $\quad \mathbf{Q}_\gamma \leftarrow \emptyset$
13: $\quad$ **for all** $\hat{r} \in \hat{\mathcal{R}}_k$ **do** ▷ gut parallelisierbar
14: $\quad\quad {}^{\diamond,\dashv^{\hat{r}}}\mathbf{x}_N \leftarrow \text{getState}(\hat{r})$
15: $\quad\quad \gamma^{\overleftrightarrow{\hat{r}}} \leftarrow \text{getSignalQuality}(\hat{r})$
16: $\quad\quad \text{append}(\mathbf{Q}_\gamma) \leftarrow < \gamma^{\overleftrightarrow{\hat{r}}}, {}^{\diamond,\dashv^{\hat{r}}}\mathbf{x}_k >$



(a) MANET Mission für eine ungestörte Umgebung. Alle Verbindung mit weniger als 70 % Qualität sind ausgeblendet.



(b) MANET Mission in einer Umgebung mit Übertragungshindernissen (orange Linien).

Abbildung 5.4.: Vergleich der MANET Mission in verschieden Umgebungsszenarien.

Experiment 5.5.1: *Ein Agentenschwarm mit 30 Agenten soll ein dezentrales redundantes Netzwerk mit möglichst großem Ausmaß bei Wahrung einer guten Verbindungsqualität bilden. Das quadratische Gebiet umfasst dabei 25 km² und weist einmal keine und einmal zwei Netzwerkhindernisse auf. Die Hindernisse separieren das Gebiet in Viertel.*

Die Abbildungen 5.4a und 5.4b zeigen die Verteilung der Agenten sowie alle Verbindungen mit einer Qualität $\gamma^{\rightsquigarrow} \geq 70\,\%$. In Abbildung 5.4b vierteln zwei Dämpfungshindernisse die Umgebung. Jede Übertragung wird pro Schnitt mit den Hindernissen um 15 dB mW gedämpft. Somit reduziert sich der RSSI deutlich, wodurch sich die Verteilung anpasst. In einem zweiten Experiment soll die maximale Leistungsfähigkeit des Ansatzes untersucht werden, indem das gesamte Gebiet abgedeckt werden soll.

Experiment 5.5.2: *Ein Agentenschwarm mit 60 Agenten soll ein dezentrales redundantes Netzwerk aufspannen, welches das gesamte quadratische Gebiet von 25 km² umfasst. Es werden keine Netzwerkhindernisse platziert.*

Die Abbildung 5.5 präsentiert das vollständig abgedeckte Gebiet, bei dem alle Agenten mehr als eine Verbindung mit hoher Qualität besitzen.

5.5.5. Analyse

Aus Abbildung 5.5 und 5.4a geht eindeutig hervor, dass das gesamte Gebiet gleichmäßig abgedeckt wird und die Agenten stets über mehr als zwei Verbindungen mit einer Qualität von mindestens 70 % verfügen, sodass keine Separierung des Netzwerks auftritt und eine kreisförmige Netzstruktur entsteht. Weiterhin besitzt jeder Knoten mindestens einen alternativen Pfad und ist somit redundant verbunden. Ändert sich die Umgebung und damit die Verbindungsqualität, so ist in Abbildung 5.4b deutlich erkennbar, dass sich die Agenten in Bereichen schlechterer Verbindungsqualität $\gamma^{\rightsquigarrow}$

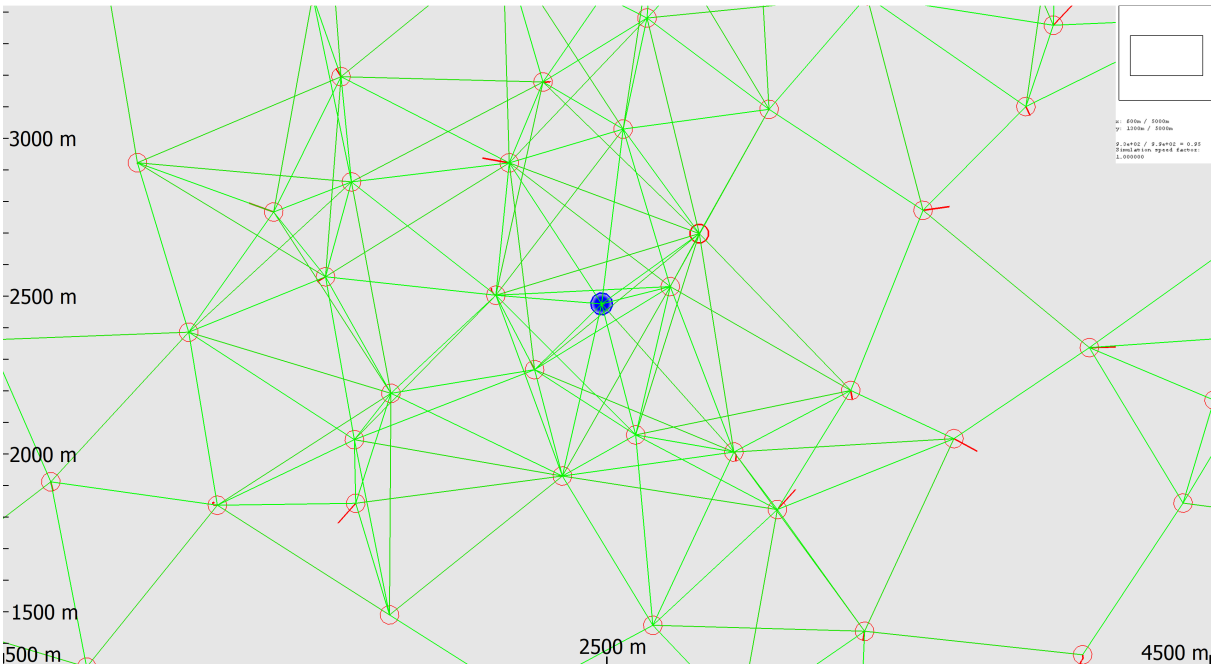


Abbildung 5.5.: MANET Mission für das gesamte Gebiet mit 60 Agenten. Alle Verbindung mit weniger als 70 % Qualität sind ausgeblendet. Die Entfernungen sind an der Skala angegeben und der Kartenausschnitt wird in der Übersichtskarte oben rechts dargestellt.

nähern, um die gewünschte Verbindungsqualität zu wahren.

5.5.6. Zusammenfassung und Referenzen

Aus der Analyse geht hervor, dass aufgrund des Logarithmus die gewünschte Verteilung im Gebiet unter Beachtung der Signalqualitäten erreicht wird. Weitere Analysen finden sich in der Arbeit Puzicha [141] und sollten noch im Hinblick auf die Hindernisse und Dämpfungshindernisse weiter betrachtet werden. Diese Mission ist in der Lage dynamisch Netzwerke in Katastrophengebieten aufzubauen, ohne die Positionen der Modems explizit vorzugeben. Somit können Rettungskräfte effizient unterstützt werden.

5.6. Mission: Eskortierung einzelner Objekte

Zur Sicherung von Rettungs- und Einsatzkräften sowie Fahrzeugkonvois ist es notwendig diese und die Umgebung um diese herum zu überwachen. Für Routine und Überwachungsaufgaben eignen sich autonome Roboterschwärme, die mit entsprechenden Sensoren ausgerüstet sind. So können Fahrbahnen nach Auffälligkeiten oder Schutthaufen auf Lebenssignaturen analysiert werden. Diese Mission beschreibt die Eskortierung eines einzelnen dynamischen Objekts.

5.6.1. Abhängigkeiten

Diese Mission benötigt einen lesenden Zugriff auf den Modelldatencontainer, indem für jeden Netzwerkteilnehmer ein erweitertes Zustandsmodell gespeichert ist. Anhand dieser Modelle wird für das zu eskortierende Objekt eine prädiktive Zustandsschätzung durchgeführt, um die Bewegungsrichtung zu ermitteln. Weiterhin ist der Zugriff auf den eigenen Positionsdatencontainer notwendig.

5.6.2. Funktionsweise und Formalisierung

Die Basis dieser Kostenfunktion bildet ein gewünschter Abstand $d_o \in \mathbb{R}_0^+$ zur dynamischen Position $\mathbf{x}_k^o \in \mathcal{X}$ des zu eskortierenden Objekts $o \in \mathcal{O}$. Dieser Abstand wird mit der gebrochen rationalen Robotersozialfunktion (vgl. [149]) kombiniert, sodass das Potenzial sowohl ein attraktives Minimum bei dem gewünschten Abstand $d > d_o$ als auch ein repulsives Maximum um das Objekt o, somit $d < d_o$, herum bildet. Dieses Maximum wird mit L_+ begrenzt und dient als Kollisionsschutz. Weiterhin kann das Minimum mit L_- definiert und zur Situation passend gewählt werden.

$$l^{r_k}_{\text{FOLLOW}}(\overset{\diamond,\dashv^{r_k}}{\mathbf{x}}_k) = \begin{cases} \underbrace{\dfrac{-2 \cdot L_- \cdot d_o^3}{\left\| \overset{\diamond,\dashv^{r_k}}{\mathbf{x}}_k - \mathbf{x}_k^o \right\|^3} - \dfrac{-3 \cdot L_- \cdot d_o^2}{\left\| \overset{\diamond,\dashv^{r_k}}{\mathbf{x}}_k - \mathbf{x}_k^o \right\|^2}}_{L}, \text{ if } L \leq L_+ \\ L_+, \text{ else} \end{cases} \tag{5.6.1}$$

Allerdings impliziert dieser Teil der Gesamtkostenfunktion nur, dass sich die Agenten auf einer beliebigen Position des Abstandskreises platzieren. Gewünscht ist hingegen eine gleichmäßige Verteilung der Agenten r_k auf Basis der Überschneidung ihrer Sensorradien $r^{r_k}_{\rightarrow}(\overset{\ltimes}{\mathbf{s}}, \hat{r}_k)$ der Sensoren $\overset{\ltimes}{\mathbf{s}}$:

$$\begin{aligned} l_{\text{INETERSECTION}}(d^{\hat{r}_k}(\overset{\diamond,\dashv^{r_k}}{\mathbf{x}}_k)) =& 2 \cdot (r^{r_k}_{\rightarrow}(\overset{\ltimes}{\mathbf{s}}, \hat{r}_k))^2 \cdot \arccos\left(\frac{d^{\hat{r}_k}(\overset{\diamond,\dashv^{r_k}}{\mathbf{x}}_k)}{2 \cdot r^{r_k}_{\rightarrow}(\overset{\ltimes}{\mathbf{s}}, \hat{r}_k)}\right) - \\ & \frac{d^{\hat{r}_k}(\overset{\diamond,\dashv^{r_k}}{\mathbf{x}}_k)}{2} \cdot \sqrt{(r^{r_k}_{\rightarrow}(\overset{\ltimes}{\mathbf{s}}, \hat{r}_k))^2 - \left(\frac{d^{\hat{r}_k}(\overset{\diamond,\dashv^{r_k}}{\mathbf{x}}_k)}{2}\right)^2} \end{aligned} \tag{5.6.2}$$

$$d^{\hat{r}_k}(\overset{\diamond,\dashv^{r_k}}{\mathbf{x}}_k) = \left\| \overset{\diamond,\dashv^{r_k}}{\mathbf{x}}_k - \overset{\diamond,\dashv^{\hat{r}_k}}{\mathbf{x}}_k \right\| \tag{5.6.3}$$

$$\mathfrak{L}^{\mathfrak{r}}(T^{\mathfrak{r}}_{\overset{\diamond,\dashv^{\mathfrak{r}}}{\mathbf{x}}_0}, \hat{\mathfrak{R}}, t, \Delta t) = \sum_{k=1}^{K} L^{r_k}_{\text{FOLLOW}}(\overset{\diamond,\dashv^{\mathfrak{r}}}{\mathbf{x}}_0, \overset{\diamond,\dashv^{r_k}}{\mathbf{x}}_k, \overset{\diamond,\dashv^{r_k}}{\mathbf{u}}_k, \hat{\mathcal{R}}_k, t_k, \overset{\diamond}{\Delta t}_k) \tag{5.6.4}$$

$$\begin{aligned} L^{r_k}_{\text{FOLLOW}}(\overset{\diamond,\dashv^{\mathfrak{r}}}{\mathbf{x}}_0, \overset{\diamond,\dashv^{r_k}}{\mathbf{x}}_k, \overset{\diamond,\dashv^{r_k}}{\mathbf{u}}_k, \hat{\mathcal{R}}_k, t_k, \overset{\diamond}{\Delta t}_k) =& l^{r_k}_{\text{FOLLOW}}(\overset{\diamond,\dashv^{r_k}}{\mathbf{x}}_k) + \\ & \omega \sum_{\hat{r}_k \in \hat{\mathcal{R}}_k} l_{\text{INETERSECTION}}(d^{\hat{r}_k}(\overset{\diamond,\dashv^{r_k}}{\mathbf{x}}_k)) \end{aligned} \tag{5.6.5}$$

Die Kosten der Überschneidung der Sensorradien können auch durch eine weniger

rechenintensive Formel approximiert werden:

$$l'_{\text{INETERSECTION}}(d^{\hat{r}_k}(\overset{\diamond}{\dot{\mathbf{x}}}{}^{r_k}_{k})) = -\omega' \cdot \min(d^{\hat{r}_k} - 2 \cdot r^{r_k}_{\rightarrow}(\overset{\ltimes}{\mathbf{s}}, \hat{r}_k), 0) \tag{5.6.6}$$

5.6.3. Erzeugte Schnittstellen und Daten

Die Mission bietet eine Schnittstelle, um die Bewegungsprädiktionen des zu eskortierenden Objektes zu teilen.

5.6.4. Experimente

Die Grundfassung dieser Mission ist bereits in der Arbeit Puzicha [141] experimentell untersucht worden.

5.6.5. Analyse

Die Aufteilung der ursprünglichen Mission in zwei separate Teilmissionen ermöglicht die Wiederverwendung der Mission zur Vermeidung der Überlagerung. Diese kann ebenfalls angewendet werden, um eine Linienformation zu erzeugen.

5.6.6. Zusammenfassung und Referenzen

Diese Mission bildet die Grundlage für die erweiterte Eskortierungsmission für Objektgruppen.

5.7. Mission: Eskortierung mehrerer Objekte

Die Mission zur Eskortierung von Objektgruppen erweitert die vorherige Mission um eine automatische Verteilung der Agenten auf die Objekte, um eine möglichst gute Gesamtabdeckung zu erzielen. Dazu werden die Objekte in Cluster separiert und die Verteilungen der Agenten auf diese optimiert. Dabei soll jedoch ein steter Wechsel der Agenten zwischen den Clustern vermieden werden. Mögliche Einsatzgebiete dieser Mission sind die Überwachung und der Schutz von Hilfslieferungen in Krisengebiete hinein.

5.7.1. Abhängigkeiten

Diese Mission benötigt dieselben Daten und Zugriffe wie die Basismission für ein Fahrzeug (siehe Abschnitt 5.6).

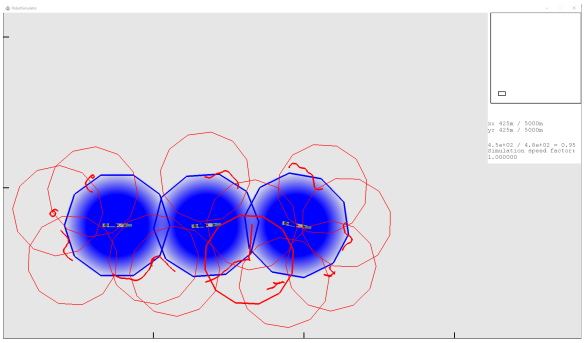
5.7.2. Funktionsweise und Formalisierung

Die eigentliche Potenzialfunktion verwendet intern jeweils eine $L^{r_k}_{\text{FOLLOW}}$-Funktion für das aktuelle Ziel $o \in \mathcal{O}$. Dieses wird zyklisch mit einer gegebenen Periode durch die „UpdateCallback“-Funktion neu berechnet (siehe Algorithmus 5.3).

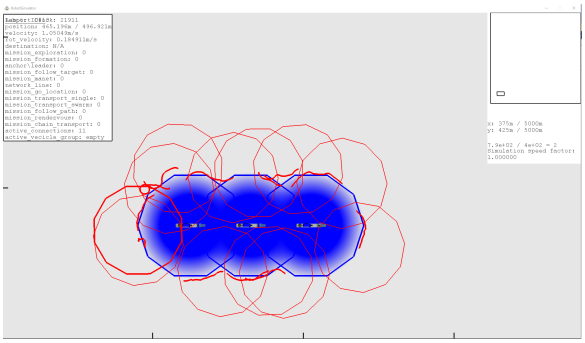
Algorithmus 5.3.: Mission *EscortConvoy*

Voraussetzung: $\hat{\mathfrak{R}}$ Modelldaten, ${}^{\diamond,\dashv^{r_k}}\mathbf{x}_k$ aktuelle Position des Agenten r_k, Δ_t Zeit seit dem letzten Update, $\mathcal{O}$ Zielobjektliste, d_{join} Vereinigungsdistanz, p_{diff} maximale Roboter-zu-Ziel-Differenz, d_{cluster} Zuordnungsdistanz

function CALCULATETARGETS(${}^{\diamond,\dashv^{r_k}}\mathbf{x}_k$, $\hat{\mathfrak{R}}$, O)
  $\mathcal{C} \leftarrow \emptyset$
  for all $o \in \mathcal{O}$ **do** ▷ Erzeuge Cluster für alle Ziele
    if $\mathcal{C} = \emptyset$ **then**
      $c \leftarrow < 0, \emptyset, \{o\} > = < p_c, \mathcal{R}_c, \mathcal{O}_c >$
      $\mathcal{C} \leftarrow \mathcal{C} \cup \{c\}$
    else
      $b_{\text{cluster}} \leftarrow 0$
      for all $c \in \mathcal{C}$ **do**
        for all $o_c \in \mathcal{O}_c^c$ **do**
          if $\|\mathbf{x}_{o_c} - \mathbf{x}_o\| < d_{\text{join}}$ **then**
            $c \leftarrow < p_c^c, \mathcal{R}_c^c, \mathcal{O}_c^c \cup o >$
            $b_{\text{cluster}} \leftarrow 1$
      if $b_{\text{cluster}} = 0$ **then**
        $c \leftarrow < 0, \emptyset, \{o\} >$
        $\mathcal{C} \leftarrow \mathcal{C} \cup \{c\}$
  $p_{\min} \leftarrow \infty, p_{\max} \leftarrow 0$
  $c_{\min}, c_{\max} \leftarrow < 0, \emptyset, \emptyset >$
  for all $\hat{\mathfrak{r}} \in \hat{\mathfrak{R}}$ **do**
    $c_r \leftarrow c \in \mathcal{C}$
    $o_r \leftarrow o \in \mathcal{O}_c^{c_r}$
    for all $c \in \mathcal{C}$ **do**
      for all $o_c \in \mathcal{O}_c^c$ **do**
        if $\|\mathbf{x}_{o_c} - {}^{\diamond,\dashv^{\hat{\mathfrak{r}}}}\mathbf{x}\| < \|\mathbf{x}_{o_r} - {}^{\diamond,\dashv^{\hat{\mathfrak{r}}}}\mathbf{x}\| \wedge \|\mathbf{x}_{o_c} - {}^{\diamond,\dashv^{\hat{\mathfrak{r}}}}\mathbf{x}\| < d_{\text{cluster}}$ **then**
          $c_r \leftarrow c$
          $o_r \leftarrow o_c$
    $c_r \leftarrow < p_c^{c_r}, \mathcal{R}_c^{c_r} \cup \{\hat{\mathfrak{r}}\}, \mathcal{O}_c^{c_r} >$
    $p_c^{c_r} \leftarrow \frac{|\mathcal{R}_c^{c_r}|}{|\mathcal{O}_c^{c_r}|}$
    if $p_c^{c_r} \leq p_{\min}$ **then**
      $p_{\min} \leftarrow p_c^{c_r}$
      $c_{\min} \leftarrow c_r$
    if $p_c^{c_r} \geq p_{\max}$ **then**
      $p_{\max} \leftarrow p_c^{c_r}$
      $c_{\max} \leftarrow c_r$
  if $p_{\max} - p_{\min} \geq p_{\text{diff}} \wedge r \in \mathcal{R}_c^{c_{\max}}$ **then**
    if r geringste Distanz zu einem o aus $c_{\min}$ **then**
      r wechsle Cluster zu $c_{\min}$
  return nahestes Objekt o aus eigenem Cluster c



(a) Konvoicluster mit Wunschdistanz d_o. Es ist zu erkennen, dass sich die Agenten auch zwischen den Fahrzeugen aufhalten. Die Agenten und ihre in rot geplanten Trajektorien befinden sich innerhalb ihrer roten Sensorradien und sollen die drei Fahrzeuge mit deren blauen Wunschdistanzen schützen.



(b) Konvoicluster mit Wunschdistanz $\frac{2}{3}d_o$. Es ist zu erkennen, dass sich die Agenten nicht zwischen den Fahrzeugen aufhalten. Die Agenten und ihre in rot geplanten Trajektorien befinden sich innerhalb ihrer roten Sensorradien und sollen die drei Fahrzeuge mit deren blauen Wunschdistanzen schützen.

Abbildung 5.6.: Vergleich der Objektabstände innerhalb eines Clusters.

5.7.3. Erzeugte Schnittstellen und Daten

Es werden keine weiteren Schnittstellen generiert, jedoch können die erzeugten Cluster weiterverwendet werden.

5.7.4. Experimente

Zur Parameteridentifikation der Mission wird der prozentuale Schwellwert für das Verlassen des maximalen Clusters hin zum minimalen Cluster experimentell ermittelt. Dieser Schwellwert, multipliziert mit der Normrate der Agenten pro Ziel $\frac{|\mathfrak{R}|}{|\mathcal{O}|}$, ergibt die Schwelle p_{diff}.

Experiment 5.7.1: *Die Schwarmgrößen 10, 15 und 60 sollen auf 2, 3, 10 Ziele aufgeteilt werden; dabei bildet jedes Ziel sein eigenes Cluster. Während der Simulation wird der prozentuale Schwellwert variiert und die maximale Roboterdifferenz der Cluster nach 3 min gemessen.*

Für kleine Schwarmgrößen und geringe prozentuale Schwellwerte ist zu beobachten , dass die Agenten oft ihre Cluster wechseln. Ein weiteres Experiment dient der Bestimmung der maximalen Objektabstände innerhalb eines Clusters.

Experiment 5.7.2: *Der maximale Objektabstand zwischen zwei Objekten eines Clusters, beginnend bei der zweifachen Wunschdistanz d_o der Objekte, ist soweit zu reduzieren, bis nur noch die Außenkontur des Clusters von den Agenten überwacht wird und sich kein Agent zwischen zwei benachbarten Objekten aufhält.*

Die Ergebnisse der Experimente sind in Tabelle 5.2 und 5.3 gelistet und in Abbildung 5.6 dargestellt.

Tabelle 5.2.: Analyse der Verteilung der Agenten auf die Ziele. Jedes Ziel bildet sein eigenes Cluster. Die maximale Differenz der Roboteranzahlen zwischen den Clustern sind über zehn Simulationsläufe gemittelt und stets zur selben Zeit gemessen.

Roboter	Ziele	Rate	Schwellwert	Mittlere maximale Roboterdifferenz
10	3	$\frac{1}{3}$	20 %	0,666
15	2	7,5	20 %	1,500
15	3	5	20 %	1,000
60	2	30	20 %	6,00
60	10	6	20 %	1,200
10	3	$\frac{1}{3}$	25 %	0,833
15	2	7,5	25 %	1,875
15	3	5	25 %	1,250
60	2	30	25 %	7,500
60	10	6	25 %	1,500
10	3	$\frac{1}{3}$	30 %	0,999
15	2	7,5	30 %	2,250
15	3	5	30 %	1,500
60	2	30	30 %	9,000
60	10	6	30 %	1,800
10	3	$\frac{1}{3}$	33 %	1,056
15	2	7,5	33 %	2,400
15	3	5	33 %	1,650
60	2	30	33 %	10,00
60	10	6	33 %	1,900
10	3	$\frac{1}{3}$	35 %	1,166
15	2	7,5	35 %	2,625
15	3	5	35 %	1,750
60	2	30	35 %	12,000
60	10	6	35 %	2,100

Tabelle 5.3.: Analyse der Objektabstände benachbarter Objekte innerhalb eines Clusters.

Distanz	60 m	55 m	50 m	45 m	40 m	35 m
Relativdistanz zu $2d_o$	1	$\frac{11}{12}$	$\frac{5}{6}$	$\frac{3}{4}$	$\frac{2}{3}$	$\frac{7}{12}$
Erfolg	×	×	×	×	✓	✓

5.7.5. Analyse

Die Simulationsdurchläufe mit 20 % und 25 % führen zu stetigen Umverteilungen der Agenten. Dies geschieht vor allem bei kleinen Schwarmgrößen, falls eine exakte gleichmäßige Verteilung nicht möglich ist. Wird der Schwellwert auf 30 % erhöht, so differiert die Roboteranzahl zwischen den Clustern deutlicher. Allerdings ist die Wechseldynamik bereits deutlich reduziert. Der Schwellwert 35 % erzeugt zu große Differenzen bei großen Schwarmgrößen und einer kleinen Clusteranzahl wie z. B. bei 2 Clustern und 60 Agenten. Die Auswahl von 33 % bewirkt schließlich einen guten Kompromiss aus geringer Wechseldynamik und geringen Differenzen zwischen den Clustern. Ferner wird das Problem der kleinen, nicht durch Drei teilbaren Schwarmgrößen für drei Cluster gelöst. Aus den Daten des Experimentes 5.7.2 geht hervor, dass ab einer Distanz von $\frac{2}{3} \cdot 2d_o$ die lokalen Minima aus dem resultierenden Gesamtpotenzialfeld entfernt sind, sodass sich die Agenten nicht mehr zwischen den Objekten aufhalten.

5.7.6. Zusammenfassung und Referenzen

Die Mission zur Eskortierung mehrerer Objekte ermöglicht eine effiziente Überwachung und Sicherung von Objekthäufungen. Dazu wird eine Gleichverteilung der Roboter mit einstellbarer Toleranz auf die Objekte erzielt. Die Zusammenfassung von Objekthäufungen zu Clustern ermöglicht eine effizientere Überwachung, da abhängig von der Distanz benachbarter Objekte nur die Außenkontur der Cluster abgedeckt werden muss. Weitere Untersuchungen zu dieser Mission sind in der Arbeit Leppersjohann [97] zu finden.

5.8. Mission: Zielansteuerung

Der Zweck der Zielansteuerungsmission (Englisch: „Go To Location“) ist es, einen Roboter vom aktuellen Zustand $\mathbf{x}$ in einen Referenzzustand $\mathbf{x}^{\text{ref}} = \mathbf{g}$ zu überführen. Aufgrund der Fähigkeit der betrachteten Kettenfahrzeuge auf der Stelle zu wenden, wird die Ausrichtung $\overset{\dagger}{\gamma}$ der Zustände vernachlässigt. Diese Mission bildet die Basis der meisten Missionen.

5.8.1. Abhängigkeiten

Die einzigen beiden Informationen, die zur Bestimmung des Missionspotenzialfeldes notwendig sind, sind der aktuelle Zustand $\mathbf{x}$ und der Zielzustand $\mathbf{x}^{\text{ref}}$. Ersterer ist im Positionsdatencontainer (siehe Abschnitt 4.2.8) gespeichert. Letzterer wird als Parameter mit der Missionsnachricht übertragen.

5.8.2. Funktionsweise und Formalisierung

Eine grundlegende Aufgabe eines jeden mobilen Roboters ist es, einen gewünschten Zielort zu erreichen. Um sich immer in Richtung des Ziels zu bewegen, wird eine

Potenzialfunktion $L_{\text{GOTO}} : \mathbb{R}_0^+ \to \mathbb{R}$ benötigt, die ein globales Minimum an der Zielkoordinate $\mathbf{g} \in \mathcal{X}$ aufweist. Das Potential wird als Funktion der Distanz zwischen der Zielposition $\mathbf{g}$ und dem Systemzustand ${}^{\diamond,\dashv^{r_k}}\mathbf{x}_k$ des Roboters r_k berechnet:

$$d_{\mathbf{g}}({}^{\diamond,\dashv^{r_k}}\mathbf{x}_k) = \| {}^{\diamond,\dashv^{r_k}}\mathbf{x}_k - \mathbf{g}_k \| \tag{5.8.1}$$

Die Funktion muss einseitig unbeschränkt und streng monoton steigend sein, sodass sich der Roboter von jedem Ort in Richtung des Ziels bewegt. Allerdings sollte das Potenzial keinen zu großen Wert annehmen, weil andernfalls abstoßende Potenziale von statischen und dynamischen Hindernissen relativ gesehen zu gering und somit ignoriert werden. Dadurch könnten Kollisionen auftreten. Folglich sind quadratische Funktionen als Basis nicht anwendbar. Auch hyperbolische Funktionen, also Potenzfunktionen mit negativen Exponenten, sind ungeeignet, da solche Funktionen für größere Distanzwerte kaum Potenzialunterschiede besitzen. Dadurch wird das Ziel für entfernte Roboter aufgrund der Umverteilungsstrategie des Optimierers praktisch unsichtbar (siehe Abschnitt 2.4.2). Es wird eine lineare Potentialfunktion als Basis gewählt. Da die Roboter bei Annäherung an das Ziel frühzeitig abbremsen sollen, wird im Zielbereich eine quadratische Funktion verwendet. Somit reduzieren die Roboter ihre Geschwindigkeit nicht abrupt. Die Funktion L_{GOTO} wird als eine Funktion der maximalen Attraktivität $a_{\mathbf{g}}$ eines Ziels $\mathbf{g}$, dem Bremsradius $r_{\mathbf{g}}$ und der Steigung der linearen Funktion $m_{\mathbf{g}}$ definiert. Diese Parameter können an die Mission und an die Potenziale anderer Entitäten der Simulation angepasst werden.

$$L^{r_k}_{\text{GOTO}}(d_{\mathbf{g}}(\cdot)) = \begin{cases} \frac{m_{\mathbf{g}}}{2r_{\mathbf{g}}} d_{\mathbf{g}}^2 - a_{\mathbf{g}} & \text{, falls } d_{\mathbf{g}} \leq r_{\mathbf{g}} \\ m_{\mathbf{g}}(d_{\mathbf{g}} - \frac{r_{\mathbf{g}}}{2}) - a_{\mathbf{g}} & \text{, sonst} \end{cases} \tag{5.8.2}$$

$$\mathfrak{L}^{\mathfrak{r}}(T^{\mathfrak{r}}_{{}^{\diamond,\dashv^{\mathfrak{r}}}\mathbf{x}_0}, \hat{\mathfrak{R}}, \mathfrak{t}, \Delta\mathfrak{t}) = \sum_{k=1}^{K} L^{r_k}_{\text{GOTO}}(d_{\mathbf{g}}({}^{\diamond,\dashv^{r_k}}\mathbf{x}_k)) \tag{5.8.3}$$

5.8.3. Erzeugte Schnittstellen und Daten

Diese Mission lässt sich stets in andere Missionen integrieren und zur Laufzeit neu parametrisieren. Dadurch bietet sie eine gute Basis für die Transport- und Rendezvousmissionen. Ebenso wird diese Mission als Grundlage für vollständige Pfadfolgemissionen und globale Pfadplanung mittels Zwischenpunkten genutzt.

5.8.4. Experimente

Die Parameteridentifikation der Zielansteuerungsmission wird mittels zweier Experimentalreihen mit je 50 Durchläufen je Parameterwert durchgeführt. Dabei werden die maximale Attraktivität, der Bremsradius und die Steigung untersucht. Die maximale Attraktivität $a_{\mathbf{g}}$ dient der relativen Anpassung der Zielfunktion an die Hinderniskosten und wird auf $a_{\mathbf{g}} = 100$ festgelegt.

Experiment 5.8.1: *Für jede Steigung $m_{\mathbf{g}}$ aus der Menge $\{0{,}1, 0{,}2, 1, 2, 5\}$ wird eine Zielansteuerungsmission mit je* 500 m *Länge durchgeführt und die benötigte Zeit gemessen.*

Tabelle 5.4.: Parametrisierung der Zielansteuerungsmission mit je 50 Durchläufen je Parameterwert.

(a) Steigungsauswertung $m_{\mathbf{g}}$.

$m_{\mathbf{g}}$	Mini -mum	Maxi -mum	Mittel -wert	unteres Quartil	oberes Quartil
5	662 s	324 s	477 s	431 s	538 s
2	571 s	311 s	389 s	363 s	475 s
1	509 s	291 s	386 s	352 s	448 s
0,2	692 s	283 s	379 s	348 s	440 s
0,1	977 s	358 s	500 s	433 s	567 s

(b) Bremsradiusauswertung $r_{\mathbf{g}}$.

$r_{\mathbf{g}}$	Mini -mum	Maxi -mum	Mittel -wert	unteres Quartil	oberes Quartil
25	304 s	602 s	401 s	363 s	461 s
15	293 s	557 s	383 s	342 s	419 s
10	281 s	515 s	379 s	351 s	407 s
7,5	289 s	508 s	390 s	354 s	416 s
5	296 s	506 s	392 s	354 s	422 s

Als nächstes wird der Bremsradius analog untersucht.

Experiment 5.8.2: *Für jeden Bremsradius $r_{\mathbf{g}}$ aus der Menge $\{5\,\mathrm{m}, 7{,}5\,\mathrm{m}, 10\,\mathrm{m}, 25\,\mathrm{m}, 25\,\mathrm{m}\}$ wird eine Zielansteuerungsmission mit je 500 m Länge durchgeführt und die benötigte Zeit gemessen.*

Die Ergebnisse sind in der Tabelle 5.4 gelistet und sind mit den Ergebnissen der Parametrisierung der Abschlussarbeit Schulz [165] nahezu identisch.

5.8.5. Analyse

Aus der Tabelle 5.4 geht hervor, dass die geringste durchschnittliche Laufzeit bei $m_{\mathbf{g}} = 0{,}2$ erzielt wird, jedoch sind das Maximum und die Varianz deutlich größer als bei $m_{\mathbf{g}} = 1$. Ferner unterscheidet sich der Mittelwert nur geringfügig. Aufgrund der vereinfachten und somit beschleunigten Berechnung, wird $m_{\mathbf{g}} = 1$ gewählt. Bei der Betrachtung der Bremsradien ist ersichtlich, dass zu große Radien aufgrund des frühzeitigen Abbremsens die Zielerreichungszeit negativ beeinflussen. Allerdings führen zu kleine Radien dazu, dass der Agent über den Zielpunkt hinausfährt und die zusätzliche Korrektur benötigt ebenfalls zusätzliche Zeit. Folglich wird $r_{\mathbf{g}} = 10\,\mathrm{m}$ gewählt. Die Attraktivität bleibt bei $a_{\mathbf{g}} = 100$, um keine zu kleinen Kosten zu erzeugen, woraufhin Hindernisse aufgrund ihrer relativ geringen absoluten Kostenwerte ignoriert werden.

5.8.6. Zusammenfassung und Referenzen

Diese Mission bildet die Basis der meisten komplexeren Transport-, Pfad- und Zielerreichungsmissionen und ist folglich ausgiebig getestet und parametrisiert.

5.9. Mission: Pfadfolge

Die Mission der Pfadfolge kann für den Objektschutz oder für das Erreichen weit entfernter Ziele mittels Wegpunkten verwendet werden. Ein Objektschutz ist möglich

indem die Wegpunkte zyklisch abgefahren werden. Dazu wird die vorherige Mission für die jeweiligen Teilstücke des Weges angewendet. Bei zu großen Entfernungen werden diese mit Hilfe automatisch generierter Zwischenpunkte aufgeteilt. Ferner behandelt diese Mission das Problem, dass Wegpunkte und Zwischenpunkte innerhalb von Hindernissen liegen können.

5.9.1. Abhängigkeiten

Ergänzend zu den Informationen der Basismission (siehe Abschnitt 5.8), müssen noch die Wegpunkte $\mathfrak{G} :=< \mathbf{g}_0, \mathbf{g}_1, \cdots >$ aus der zugehörigen Nachricht extrahiert werden.

5.9.2. Funktionsweise und Formalisierung

Die Pfadfolge erweitert die Zielansteuerung um mehrere Aspekte. Es kann sowohl eine ganze Punktfolge übergeben werden, die auch zyklisch durchlaufen werden kann, als auch automatisch Zwischenpunkte generiert werden, falls der Punktabstand über einem Schwellwert liegt. Für jeden Punkt wird ermittelt, wie nah er angefahren werden kann. Dazu wird ein dynamischer Akzeptanzradius $r_{\mathbf{g}}$ pro Ziel $\mathbf{g}$ der Wegpunktliste $\mathfrak{G}$ gespeichert. So lange der Knoten nicht erreicht wird, wächst dieser mit der Zeit. Sobald das Ziel mit Hilfe eines entsprechenden Radius erreicht worden ist, wird dieser gespeichert. Wird der Radius zu einem späteren Zeitpunkt unterschritten, so wird stets das Minimum gespeichert. Somit lassen sich die Erreichung garantieren und dennoch die Rundenzeiten für zyklische Pfade optimieren. Die Distanzfunktion $d_{\mathbf{g}}$ wird wie folgt erweitert:

$$d_{\mathbf{g}}(\mathbf{x}_k^{r_k}) = \max(\|\mathbf{x}_k^{r_k} - \mathbf{g}_k\| - r_{\mathbf{g}}, 0) \tag{5.9.1}$$

5.9.3. Erzeugte Schnittstellen und Daten

Durch die Annahme einer Wegpunktliste $\mathfrak{G}$ können sehr einfach stützstellenbasierte globale Planer und gitterbasierte Verfahren, wie z. B. A* (siehe Abschnitt 2.2.1.1 und 5.15), dieser Mission vorgeschaltet werden. Diese erzeugen dann ausschließlich die geforderte Wegpunktliste. Die modellprädiktive Regelung optimiert den lokalen Pfad, sodass dies einer Aufteilung in lokale und globale Pfadplanung entspricht (siehe Abschnitt 2.2.1.2). Jedoch können zeitgleich auch andere Missionen und Rahmenbedingungen eingehalten werden, wodurch sich dieser Ansatz von den Existierenden unterscheidet.

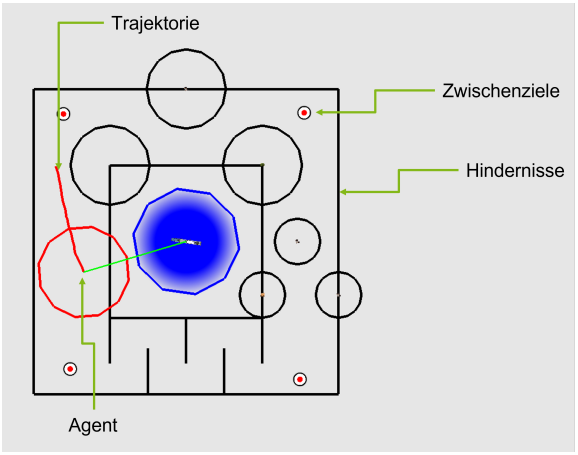
5.9.4. Experimente

Zur Bestimmung der Steigung $m_{\mathbf{g}}$ der Zielansteuerungsmission innerhalb der Pfadfolgemission durchfährt ein Agent ein Testlabyrinth (siehe Abbildung 5.7a) und die Rundenzeit wird gemessen.

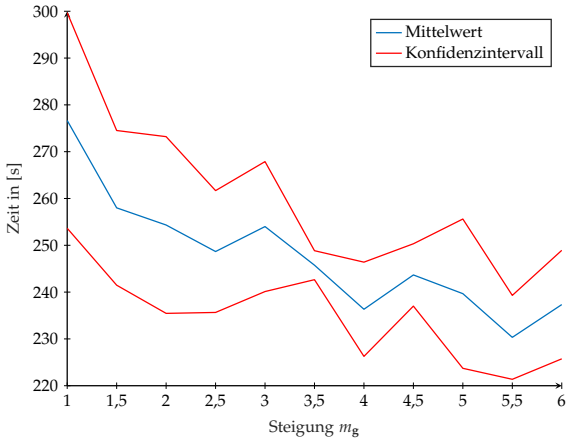
Algorithmus 5.4.: Mission *Follow Path*

Voraussetzung: $\mathfrak{G} :=< \mathbf{g}_0, \mathbf{g}_1, \cdots >$ geordnete Liste, $\mathbf{g}$ aktuelles Ziel, $r_\mathbf{g}$ Zielakzeptanzradius, $\bar{r}$ Startradius, r^+ Radiusschrittweite, d^+ Distanzinkrement, t^+ Zeitinkrement, $\tilde{d}_\mathbf{g}$ minimale Distanz zum Ziel, $t^{\dashrightarrow}$ Zeit bis zum nächsten Inkrement

1: **function** INIT
2: $\quad\mathfrak{G}' \leftarrow \emptyset$
3: $\quad$**for all** $\mathbf{g} \in \mathfrak{G}$ **do**
4: $\quad\quad r_\mathbf{g} \leftarrow \infty$
5: $\quad\quad\mathfrak{G}' \leftarrow \mathfrak{G}' \cup \{< \mathbf{g}, r_\mathbf{g} >\}$
6: **function** UPDATECALLBACK(t_k, Δ_t)
7: $\quad$**if** $\|\mathbf{x}_k^r - \mathbf{g}\| < r_\mathbf{g}$ **then**
8: $\quad\quad$**if** $\|\mathbf{x}_k^r - \mathbf{g}\| < \bar{r} \wedge t^{\dashrightarrow} < t_k$ **then**
9: $\quad\quad\quad$NEXTNODE( )
10: $\quad\quad$**else if** $\|\mathbf{x}_k^r - \mathbf{g}\| < \tilde{d}_\mathbf{g}$ **then**
11: $\quad\quad\tilde{d}_\mathbf{g} \leftarrow \|\mathbf{x}_k^r - \mathbf{g}\| - d^+$
12: $\quad\quad t^{\dashrightarrow} \leftarrow t_k + t^+$
13: $\quad$**else**
14: $\quad\quad$**if** $\|\mathbf{x}_k^r - \mathbf{g}\| < \tilde{d}_\mathbf{g}$ **then**
15: $\quad\quad\quad\tilde{d}_\mathbf{g} \leftarrow \|\mathbf{x}_k^r - \mathbf{g}\| - d^+$
16: $\quad\quad\quad t^{\dashrightarrow} \leftarrow t_k + t^+$
17: $\quad\quad$**else if** $t^{\dashrightarrow} < t_k$ **then**
18: $\quad\quad\quad r_\mathbf{g} \leftarrow r_\mathbf{g} + r^+$
19: $\quad\quad\quad t^{\dashrightarrow} \leftarrow t_k + t^+$
20: **function** NEXTNODE
21: $\quad$**if** ISCYCLIC($\mathfrak{G}$) **then**
22: $\quad\quad r_\mathbf{g} \leftarrow \tilde{d}_\mathbf{g} + d^+$
23: $\quad\quad\mathfrak{G}' \leftarrow \mathfrak{G}' \cup \{< \mathbf{g}, r_\mathbf{g} >\}$
24: $\quad$**else if** $\mathfrak{G}' = \emptyset$ **then**
25: $\quad\quad$end mission
26: $\quad$**else**
27: $\quad\quad< \mathbf{g}, r_\mathbf{g} > \leftarrow < \mathbf{g}, r_\mathbf{g} > \in \mathfrak{G}'$
28: $\quad\quad\mathfrak{G}' \leftarrow \mathfrak{G}' \setminus \{< \mathbf{g}, r_\mathbf{g} >\}$
29: $\quad\quad\tilde{d}_\mathbf{g} \leftarrow \infty$



(a) Testlabyrinth für die Bestimmung des Gradienten. Der Agent befindet sich im Zentrum des roten Sensorkreises. Die rote kurvige Strecke ist die geplante Trajektorie und die rot-weißen Kreise markieren die Pfadpunkte.



(b) Laufzeitgraph der Rundenzeiten abhängig vom Wert der Steigung m_g.

Abbildung 5.7.: Bestimmung der Steigung m_g der internen Zielansteuerungsmission.

Experiment 5.9.1: *Ein Agent durchläuft das in Abbildung 5.7a gezeigte Testlabyrinth für* $m_g = 1, 1{,}5, 2, \ldots, 6$ *und wiederholt dies zehn Mal.*

Die grundlegende Parameteridentifikation ist der Abschlussarbeit Grabenschröer [47] entlehnt und verwendet $\bar{r} = 7\,\mathrm{m}$, $r^+ = 0{,}4\,\mathrm{m}$, $d^+ = 0{,}2\,\mathrm{m}$, $t^+ = 2\,\mathrm{s}$, $t^{-\rightarrow} = 25\,\mathrm{s}$. Sie unterscheidet sich jedoch darin, dass alle Zielpunkte erreichbar sind. Die Abschlussarbeit untersucht den Fall, dass der obere rechte Zielpunkt außerhalb der rechteckigen Begrenzung liegt, sodass länger gewartet werden muss, bis der Zielakzeptansradius r_g angewachsen ist. Da der Verlauf der Ergebnisdaten analog zu den Daten aus Abbildung 5.7b ist, werden diese Werte für die Mission übernommen.

5.9.5. Analyse

Aus Abbildung 5.7b geht hervor, dass sich die Rundenzeiten für das Labyrinth mit zunehmender Steigung m_g einer asymptotischen Minimalzeit, bedingt durch die Stellgrößenbeschränkungen und die Hindernisse, annähern. Die Ziele sind je 150 m von einander entfernt, sodass ohne Hindernisse und ohne Lenkvorgänge eine rechnerische Minimalzeit von $\frac{4 \cdot 150\,\mathrm{m}}{3\,\mathrm{m\,s^{-1}}} = 200\,\mathrm{s}$ möglich ist. Ab einer Steigung von $m_g = 5{,}5$ wird das asymptotische Minimum mit geringer Varianz erreicht. Allerdings wird diese Steigung nicht empfohlen, weil zum Teil Kanten geschnitten werden und Kollisionen nicht ausgeschlossen sind. Eine Steigung von $m_g = 2{,}5$ bietet eine robuste Lösung ohne Kollision bei gleichzeitiger minimaler Zeit und wird daher gewählt. Eine größere Steigung wird nicht empfohlen, da ansonsten die Kosten für weit entfernte Ziele zu stark steigen und die Robustheit abnimmt.

5.9.6. Zusammenfassung und Referenzen

Die Pfadfolgemission ist ein weiterer Baustein für weitere und komplexe Missionen, bei denen Wegpunkte nur Zwischenstationen sind. Zudem wird durch sie die globale Pfadplanung ermöglicht. Weiterführende Analysen für den Fall, dass die Zielpunkte nicht immer erreichbar sind, sind in der Bachelorarbeit Grabenschröer [47] aufgeführt. Ein Verhaltensunterschied ist bis auf die verlängerte Wartezeit für das Anwachsen des Akzeptanzradius nicht erkennbar.

5.10. Mission: Linien- und Dreiecksformation

Such- und Rettungsaktionen erfordern in der Regel ein organisiertes Vorgehen und ein lückenloses Abschreiten des Suchgebietes durch die Rettungskräfte. Eine chaotische Suche, wie sie z. B. in der Explorationsmission vorgestellt wird, erzeugt nur einen geringen Mehrwert. Daher wird nun ein Ansatz aufgezeigt, bei dem die Agenten eine feste Formation bilden. Diese Mission kann mit der Pfadfolgemission für definierte Pfade oder mit der Explorationsmission für unbekannte Gebiete kombiniert werden. Dadurch lassen sich die Lücken der Exploration schließen und ein gutes Lagebild erzeugen.

5.10.1. Abhängigkeiten

Diese Mission benötigt für den Zugriff auf den eigenen Zustand $\overset{\diamond,\dashv}{\mathbf{x}}$ den eigenen Positionsdatencontainer. Zusätzlich wird noch der Modelldatencontainer verwendet, um die Identifikationsnummern und Positionen der anderen Agenten zur Zuordnung der Abstände zur eigenen Position zu bestimmen.

5.10.2. Funktionsweise und Formalisierung

Die Linienformation wird definiert durch die Positionen der Agenten mit den geringsten Identifikationsnummern. Die niedrigste Nummer bezeichnet den Führer ${r_k}^0$ und die zweite den Anker ${r_k}^1$. Alle anderen Agenten bilden das Gefolge. Die konstruierende Kostenfunktion besteht aus drei Elementen und wird in Gleichung 5.10.1 dargelegt. Erstens werden Kosten für die Distanz zur definierten Solllinie berechnet. Zweitens werden erneut Kosten für die Überschneidung der Sensorflächen bestimmt, sodass sich die Agenten auf der Linie verteilen. Als letztes werden Kosten erhoben, falls sich die Agenten zu weit voneinander entfernen. Dies kann mit der Funktion l_{CENTER} bestimmt werden, indem die Entfernungen zu den beiden nächsten Robotern bestimmt werden oder es wird der Mittelpunkt der Agenten als Schwerpunkt angenommen und der Abstand zu diesen mit Kosten belegt.
Die Dreiecksformation erweitert die Linienformation um einen weiteren Anker ${r_k}^2$. Sie besteht somit aus zwei Linien jeweils vom Führer aus zu einem Anker. Der Führer bestimmt die Richtung bzw. die Bewegung und die Anker halten die Formation, indem eine weitere Kostenkomponente für die Abweichung vom Wunschwinkel hinzukommt.

Das restliche Gefolge hat sich gleichmäßig auf beide Schenkel des Dreiecks zu verteilen, sodass die Basis frei bleibt. Diese Verteilung wird in Algorithmus 5.5 beschrieben.

$$\mathfrak{L}^{\mathfrak{r}}(T^{\mathfrak{r}}_{\overset{\diamond,\dashv^{\mathfrak{r}}}{\mathbf{x}}_0}, \hat{\mathfrak{R}}, \mathfrak{t}, \Delta\mathfrak{t}) = \sum_{k=1}^{K} L^{r_k}_{\text{FORMATION}}(\overset{\diamond,\dashv^{\mathfrak{r}}}{\mathbf{x}}_0, \overset{\diamond,\dashv^{r_k}}{\mathbf{x}}_k, \overset{\diamond,\dashv^{r_k}}{\mathbf{u}}_k, \hat{\mathcal{R}}_k, t_k, \overset{\diamond}{\Delta t}_k) \tag{5.10.1}$$

$$l_{\text{CENTER}}(\overset{\diamond,\dashv^{r_k}}{\mathbf{x}}_k, \hat{\mathcal{R}}_k) = \| \overset{\diamond,\dashv^{r_k}}{\mathbf{x}}_k - \sum_{\hat{r}_k \in \hat{\mathcal{R}}_k} \overset{\diamond,\dashv^{\hat{r}_k}}{\mathbf{x}}_k \| \tag{5.10.2}$$

$$l^{r_k}_{\text{FORMATION LINE}}(\overset{\diamond,\dashv^{r_k}}{\mathbf{x}}_k, \hat{r}^0_k, \hat{r}^1_k) = \begin{cases} (d^{r_k}_{\text{LINE}})^2, \text{ falls } d^{r_k}_{\text{LINE}} \leq 1 \\ a_F \cdot (d^{r_k}_{\text{LINE}}), \text{ falls } a_F \cdot (d^{r_k}_{\text{LINE}}) \leq b_F \\ b_F + \log(a_F \cdot (d^{r_k}_{\text{LINE}}) - b_F + 1) \cdot a_F, \text{ sonst} \end{cases} \tag{5.10.3}$$

$$d^{r_k}_{\text{LINE}}(\overset{\diamond,\dashv^{r_k}}{\mathbf{x}}_k, \hat{r}^0_k, \hat{r}^1_k) = \frac{|(\overset{\diamond,\dashv^{r_k}}{\mathbf{x}}_k - \overset{\diamond,\dashv^{\hat{r}^0_k}}{\mathbf{x}}) \times (\overset{\diamond,\dashv^{\hat{r}^1_k}}{\mathbf{x}} - \overset{\diamond,\dashv^{\hat{r}^0_k}}{\mathbf{x}})|}{|\overset{\diamond,\dashv^{\hat{r}^1_k}}{\mathbf{x}} - \overset{\diamond,\dashv^{\hat{r}^0_k}}{\mathbf{x}}|} \tag{5.10.4}$$

$$\begin{aligned} L^{r_k}_{\text{FORMATION}}(\overset{\diamond,\dashv^{\mathfrak{r}}}{\mathbf{x}}_0, \overset{\diamond,\dashv^{r_k}}{\mathbf{x}}_k, \overset{\diamond,\dashv^{r_k}}{\mathbf{u}}_k, \hat{\mathcal{R}}_k, t_k, \overset{\diamond}{\Delta t}_k) = {} & \omega_1 \cdot l^{r_k}_{\text{FORMATION LINE}}(\overset{\diamond,\dashv^{r_k}}{\mathbf{x}}_k, \hat{r}^0_k, \hat{r}^1_k) + \\ & \omega_2 \cdot \sum_{\hat{r}_k \in \hat{\mathcal{R}}_k} l'_{\text{INETERSECTION}}(d^{\hat{r}_k}(\overset{\diamond,\dashv^{r_k}}{\mathbf{x}}_k)) + \\ & \omega_3 \cdot l_{\text{CENTER}}(\overset{\diamond,\dashv^{r_k}}{\mathbf{x}}, \hat{\mathcal{R}}_k) \end{aligned} \tag{5.10.5}$$

Dabei definiert a_F einen Gewichtungsfaktor des Abstandes zur Linie und b_F einen Schwellwert, ab dem die lineare Kostenfunktion zu einer logarithmischen wechselt. Dadurch werden die Abstandskosten für weite Entfernungen nicht zu groß.

5.10.3. Erzeugte Schnittstellen und Daten

Die Formationen können mit den meisten Fahrtmissionen kombiniert werden. Vor allem eignet sich die Kombination mit der Exploration, der Pfadfolge oder der Triangulierung der Flächen für lückenloses Abfahren der Pfade. Die Kombination mit individuellen Missionen, wie dem Transport von Objekten, ist nicht zielführend, weil sich die Kostenfunktionen gegenteilig verhalten und somit keine Mission zufriedenstellend erledigt wird. Daten zur Weiterverarbeitung beinhalten die Zugehörigkeit zu einer Linie und ob aktuell eine besondere Rolle wie Führer oder Anker aktiv ist.

5.10.4. Experimente

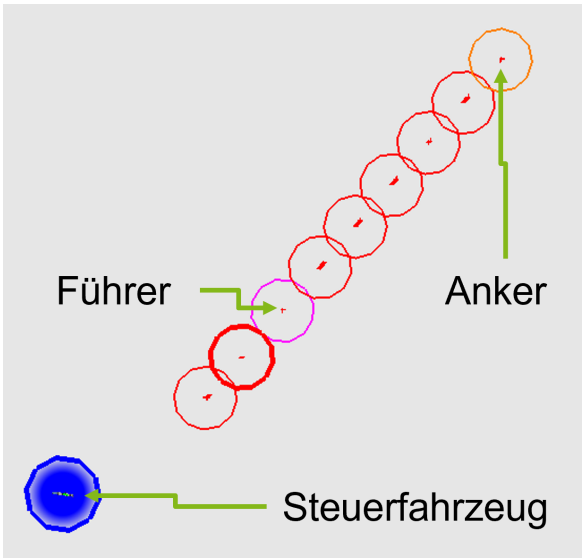
Damit die Agenten brauchbare Formationen bilden, müssen die Kostenverhältnisse zwischen den Formationskosten, den Abstandskosten und den Überlappungskosten ermittelt werden. Dazu wird eine kurze Experimentreihe genutzt.

Experiment 5.10.1: *Der Formationsfaktor a_F wird von 1 bis inklusive 5 mit Schrittweite 1 variiert und die Laufzeit über zehn Experimente hinweg gemittelt. Dabei beträgt $b_F = 350$, damit die Gesamtkosten nicht zu groß werden. Ferner gilt $\omega_1 = \omega_2 = \omega_3 = 1$.*

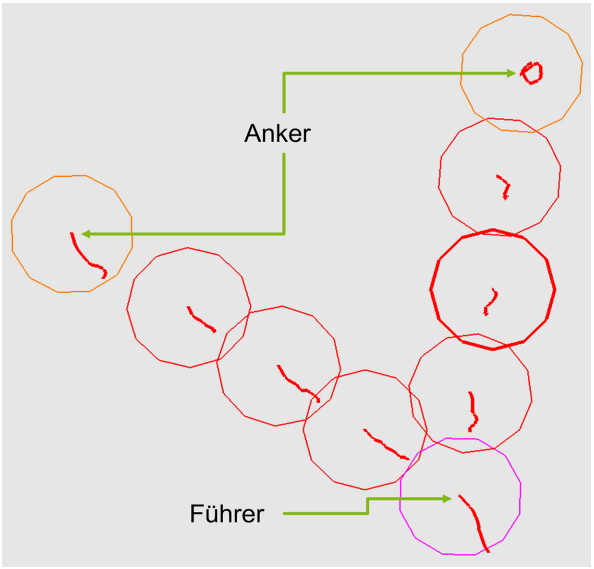
Algorithmus 5.5.: Mission *FormationTriangle*

Voraussetzung: r_k Informationen des eigenen Agenten, $\hat{\mathcal{R}}_k^{\text{left}}$ Menge der Agentenmodelle die dem linken Schenkel zu geordnet sind, $\hat{\mathcal{R}}_k^{\text{right}}$ Menge der Agentenmodelle die dem rechten Schenkel zugeordnet sind, s^{r_k} Schenkel, zu dem r_k zugeordnet ist, $\hat{r}_k^0$ Führer, $\hat{r}_k^1$ Anker 1, $\hat{r}_k^2$ Anker 2

1: **function** SETTRIANGLESIDE($r_k, \hat{\mathcal{R}}_k^{\text{left}}, \hat{\mathcal{R}}_k^{\text{right}}, s^{r_k}$)
2: $s_{\text{old}}^{r_k} \leftarrow s^{r_k}$
3: **if** $r_k = \hat{r}_k^0 \vee r_k = \hat{r}_k^1 \vee r_k = \hat{r}_k^2$ **then** ▷ r_k ist Führer oder Anker
4: $s^{r_k} = 0$ ▷ Keine Zuteilung zu einer Seite
5: **else** ▷ r_k ist Gefolge
6: $r_{\text{far}} \leftarrow \emptyset$ ▷ Initiale Belegung
7: $d_{\text{far}} \leftarrow \infty$
8: **if** $|\hat{\mathcal{R}}_k^{\text{left}}| > |\hat{\mathcal{R}}_k^{\text{right}}| + 1$ **then** ▷ Zu viele Agenten auf der linken Seite
9: **for all** $\hat{r}_k^l \in \hat{\mathcal{R}}_k^{\text{left}}$ **do**
10: $d \leftarrow$GETDISTANCETOLINE($\hat{r}_k^l$,$\hat{r}_k^0$,$\hat{r}_k^1$)
11: **if** $d < d_{\text{far}}$ **then**
12: $r_{\text{far}} \leftarrow \{\hat{r}_k^l\}$
13: $d_{\text{far}} \leftarrow d$
14: **if** $r_{\text{far}} = \{r_k\}$ **then**
15: $s^{r_k} \leftarrow 1$ ▷ rechter Schenkel
16: $\hat{\mathcal{R}}_k^{\text{left}} \leftarrow \hat{\mathcal{R}}_k^{\text{left}} \setminus r_{\text{far}}$
17: $\hat{\mathcal{R}}_k^{\text{right}} \leftarrow \hat{\mathcal{R}}_k^{\text{right}} \cup r_{\text{far}}$
18: **else if** $|\hat{\mathcal{R}}_k^{\text{right}}| > |\hat{\mathcal{R}}_k^{\text{left}}| + 1$ **then** ▷ Zu viele Agenten auf der rechten Seite
19: **for all** $\hat{r}_k^r \in \hat{\mathcal{R}}_k^{\text{right}}$ **do**
20: $d \leftarrow$GETDISTANCETOLINE($\hat{r}_k^r$,$\hat{r}_k^0$,$\hat{r}_k^2$)
21: **if** $d < d_{\text{far}}$ **then**
22: $r_{\text{far}} \leftarrow \{\hat{r}_k^r\}$
23: $d_{\text{far}} \leftarrow d$
24: **if** $r_{\text{far}} = \{r_k\}$ **then**
25: $s^{r_k} \leftarrow 2$ ▷ linker Schenkel
26: $\hat{\mathcal{R}}_k^{\text{right}} \leftarrow \hat{\mathcal{R}}_k^{\text{right}} \setminus r_{\text{far}}$
27: $\hat{\mathcal{R}}_k^{\text{left}} \leftarrow \hat{\mathcal{R}}_k^{\text{left}} \cup r_{\text{far}}$
28: **if** $s_{\text{old}}^{r_k} \neq s^{r_k}$ **then**
29: $\mathbf{m}^{\hookrightarrow} \leftarrow$CREATEUPDATEMSG($r_k$,$s^{r_k}$)
30: SENDMSG(r_k, $\mathbf{m}^{\hookrightarrow}$)



(a) Darstellung der erfolgreichen Linienformation mit neun Agenten.



(b) Darstellung der Dreiecksformation mit neun Agenten

Abbildung 5.8.: Visualisierung der funktionsfähigen Formationsmission mittels eins Führers und Ankern.

Tabelle 5.5.: Bestimmung des Formationsfaktors a_F.

Faktor	Mittlere Zeit des Linienaufbaus	Mittlere Zeit des Dreiecksaufbaus	Kosten der Formation $l^{r_k}_{\text{FORMATION LINE}}$ für $d = 70\,\text{m}$	a_F
1	313 s	∞s	70	1
2	161 s	445 s	140	2
3	140 s	360 s	210	3
4	138 s	321 s	280	4
5	123 s	265 s	350	5

Das Ergebnis wird in Tabelle 5.5 dargestellt. Die grundlegende Parameteridentifikation ist in der Abschlussarbeit Leppersjohann [97] weiter ausgeführt worden.

5.10.5. Analyse

Es ist deutlich ein exponentieller bzw. asymptotischer Verlauf der mittleren Zeit in Abhängigkeit zum Faktor der Formationskosten zu erkennen. Es muss stets beachtet werden, dass die Kosten für Missionen nicht deutlich die Kosten für Hindernisse, welche in der Regel zwischen 300 und 900 liegen, überschreiten. Daher wird der empirische Wert 3 als Faktor für die Formationskosten gewählt. Dieser bildet einen guten Kompromiss aus geringer Laufzeit und angemessenen Kosten. Alternativ kann zur Bestimmung der Gerade durch die Positionen der Agenten mit extremen Identifikationsnummern eine Ausgleichsgerade durch alle Positionen der Formation bestimmt werden. Allerdings wird die Ausgleichsgerade stark durch weiter entfernte Agenten, die Ausreißer bilden, beeinflusst. Ferner ist es schwieriger diese Mission mit einer

anderen Mission zu kombinieren, da nicht ein oder ein paar wenige Agenten die Hauptbewegungsrichtung vorgeben können. Daher wird diese Möglichkeit nicht weiter experimentell untersucht.

5.10.6. Zusammenfassung und Referenzen

Durch die Einhaltung einer Formation lassen sich einige Missionen, wie z. B. die Exploration, verbessern oder sogar erweitern, wie z. B. die Pfadfolgemission. Im Allgemeinen bieten diese Missionen die Möglichkeit zur koordinierten Suche und Überwachung. Mit Hilfe der vorgestellten Kostenfunktion können weitere Formationen einfach ergänzt werden, indem diese in Segmente unterteilt werden. Anschließend muss nur ein Algorithmus zur Zuordnung der Agenten zu den Segmenten, wie für die Dreiecke, erstellt werden. Somit lassen sich perspektivisch auch komplexe Formationen bilden und halten.

5.11. Mission: Transport

Einer der wesentlichen Anwendungsfälle autonomer Roboter ist der Transport von Waren zu gegebenen Ablageorten. Dies gilt sowohl für die Intra- als auch für die Interlogistik und ebenso für die Verteilung von Hilfsmitteln in Krisengebieten. Es existieren drei Kombinationsmöglichkeiten für die Zuordnung von Waren zu Agenten. Erstens kann ein Agent alle Waren transportieren, zweitens können die Waren von mehreren Agenten transportiert werden und drittens kann eine bestimmte Ware von einer Gruppe von Agenten transportiert werden. Die erste Möglichkeit wird bereits durch die Zielansteuerungsmission abgedeckt. Ebenso lässt sich der Transport eines komplexen Objektes mit Hilfe der gezielten Fahrt und einer passenden Formation erreichen. Die zweite Kombination ist ein schwieriges Problem in der Logistik gerade für dezentral arbeitende Agenten. In der Regel gibt es eine zentrale Verteilungsinstanz oder die Verteilung der Waren wird mittels eines Konsensalgorithmus global geregelt (vgl. [91] und [30]), da lokale Ansätze eine geringe Güte erzielen. Jedoch liegt der Fokus dieser Arbeit auf der Dezentralität mit hohen Ausfallwahrscheinlichkeiten innerhalb der Katastrophengebiete.

5.11.1. Abhängigkeiten

Da auch eine möglichst dezentrale Lösung einen Überblick über die Positionsverteilung der Agenten benötigt, wird der Zugriff auf den Modelldatencontainer benötigt. Um das Kommunikationsnetzwerk nicht deutlich mehr zu belasten, werden ausschließlich optionale Nachrichten gesendet, falls Waren aufgenommen oder abgelegt werden.

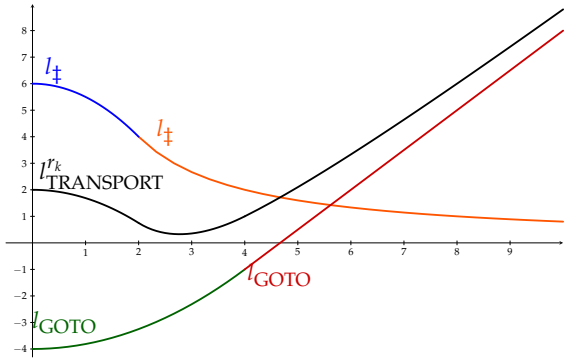
Weiterhin wird der eigene Zustand $\hat{\mathbf{x}}^{\diamond,\dashv}$ des Positionsdatencontainers benötigt und ein Lagermodul, das die Aufnahme und Abgabe von Objekten ermöglicht. Hier kann ein intelligenter Behälter mit automatischer Benachrichtigung verwendet werden (siehe Abschnitt 4.2.12).

5.11.2. Funktionsweise und Formalisierung

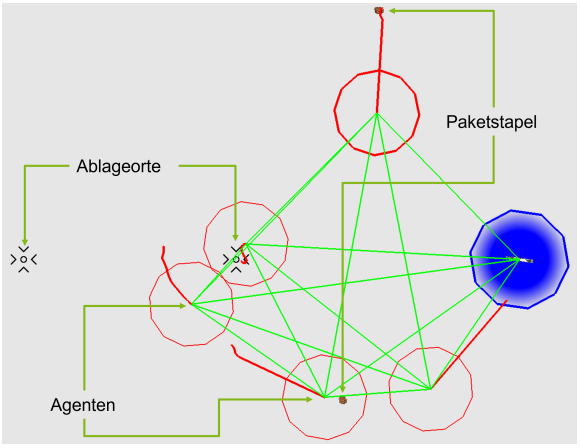
Die Basis der komplexeren Transportmission, fortan Schwarmtransport genannt, bildet erneut die Zielansteuerungsmission. Es seien die aktuellen Positionen der Waren sowie die Ablageorte mit $\mathbf{g} \in \mathcal{G} \subseteq \mathcal{X}$ beschrieben.

Durch die eigenständigen Entscheidungen muss beachtet werden, dass nicht zwei Roboter dasselbe Paket oder Pakete, die dicht beieinander sind, zeitgleich anfahren und aufnehmen. Dazu werden Sicherheitsabstände und Wartezeiten benötigt (siehe Algorithmus 5.6). Nur kurze Nachrichten der intelligenten Behälter informieren die anderen Agenten über die Position, an der ein Paket abgelegt oder abgeholt wird. Obgleich diese Nachrichten optional sind oder während der Übertragung verloren gehen können, verkürzen sie die Zeit für die Erfüllung der Mission, weil unnötige Wegstrecken und Überprüfungen entfallen. Die Schutzbereiche sind notwendig, da ein Agent seine Abhol- oder Absetzvorgänge nur ohne zusätzliche Störungen durch die anderen Agenten in kürzester Zeit erledigen kann. Der Schutz wird durch einen Blockierungsradius $r_{\ddagger}$ und eine Blockierungsfunktion $s_{\ddagger}^{r_k}(\hat{\mathcal{R}}_k)$ bestimmt und in Gleichung 5.11.2 definiert. Die Blockierungsfunktion sind eine komplexe Aggregation distanzabhängiger Kosten der Agenten im selben Transportzustand. Sei $b_{\text{TRANSPORT}}(r_k, \hat{r}_k)$ eine Wahrheitsfunktion, die 1 bzw. wahr zurückgibt, falls die Agenten denselben Transportzustand aufweisen. Dieser kann entweder „beladen“ oder „entladen“ sein. So können die Kosten in Gleichung 5.11.3 angegeben werden.

Dabei sei $d_{\ddagger}$ der maximale Blockierungsabstand eines Agenten $\hat{r}_k$, der einen Einfluss auf den Agenten r ausübt und auch nur, dann wenn sie denselbe Transportabsicht besitzen. $\omega_{\ddagger}$ ist ein Gewichtungsfaktor. Neben der Blockierung durch andere Agenten reduziert auch die Wartezeit an einem Pakethäufungspunkt dessen Attraktivität, sodass die Agenten ab einer situationsabhängigen Wartezeit das Ziel wechseln und so die Zielverteilung lokal optimieren und damit blockierende Situationen an Engstellen indirekt mit lösen. Um dies zu erreichen, wird das anziehende Potenzial $a_{\mathbf{g}}$ (siehe Abschnitt 5.8) zur Laufzeit auf der Grundlage einer akzeptablen Wartezeitschwelle $t_W \in \mathbb{R}_0^+$ am Standort und der Anzahl der Pakete an einem Häufungspunkt von Paketen $\mathbf{g}$ angepasst (siehe Algorithmus 5.6). Die Aktualisierungszeit $t^{\dashrightarrow} \in \mathbb{R}_0^+$ von Algorithmus 5.6 ist unabhängig von Δt_k, kann aber auf $t^{\dashrightarrow} = \Delta t_k$ gesetzt werden. Folglich versuchen die Agenten zunächst, das nächstgelegene Paket einzusammeln und es zum nächstgelegenen Ablageort zu bringen. Kommt es jedoch zu einem Wettlauf, wird der schnellste Agent auf der Grundlage seiner übermittelten Trajektorie priorisiert, und die übrigen Agenten warten eine bestimmte Schwellwertzeit t_W ab, um nacheinander Pakete von dem Häufungspunkt zu holen. So verschwenden sie keine Zeit, indem sie die Ziele zu oft wechseln. Falls einer der intelligenten Behälter eine Aufnahmenachricht sendet und es das einzige verfügbare Paket ist oder die Wartezeit $t_{\mathbf{g}}$ am Ziel $\mathbf{g}$ steigt und den Standort im Vergleich zu anderen weniger attraktiv macht, bestimmen die Argumente der Minimierung für die Agenten ein neues Ziel. Mit zunehmender Anzahl von Agenten werden die Pakete ebenfalls unattraktiver. Die sich daraus ergebende Potenzialfunktion, die die Agenten dazu veranlasst, in einer bestimmten Entfernung zu warten, während sie versuchen, ein Ziel zu erreichen, ist



(a) Darstellung der Kostenfunktion der Schwarmtransportmission.



(b) Darstellung des Szenarios des Experimentes.

Abbildung 5.9.: Visualisierung der Schwarmtransportmission.

in Abbildung 5.9a dargestellt. Die resultierende Gesamtfunktion lautet:

$$\mathfrak{L}^{\mathfrak{r}}(T^{\mathfrak{r}}_{\overset{\diamond,\dashv^{\mathfrak{r}}}{\mathbf{x}}_0}, \hat{\mathfrak{R}}, \mathfrak{t}, \Delta\mathfrak{t}) = \sum_{k=1}^{K} L^{r_k}_{\text{TRANSPORT}}(\overset{\diamond,\dashv^{\mathfrak{r}}}{\mathbf{x}}_0, \overset{\diamond,\dashv^{r_k}}{\mathbf{x}}_k, \overset{\diamond,\dashv^{r_k}}{\mathbf{u}}_k, \hat{\mathcal{R}}_k, t_k, \overset{\diamond}{\Delta t}_k) \tag{5.11.1}$$

$$l^{r_k}_{\ddagger}(d_{\mathbf{g}}(\cdot)), \hat{\mathcal{R}}_k) = \begin{cases} -\frac{s^{r_k}_{\ddagger}(\hat{\mathcal{R}}_k)}{3r^2_{\ddagger}} d^2_{\mathbf{g}} + s^{r_k}_{\ddagger}(\hat{\mathcal{R}}_k) & \text{, falls } d \leq r_{\ddagger} \\ \frac{2s^{r_k}_{\ddagger}(\hat{\mathcal{R}}_k) \cdot r_{\ddagger}}{3d_{\mathbf{g}}} & \text{, sonst} \end{cases} \tag{5.11.2}$$

$$s^{r_k}_{\ddagger}(\hat{\mathcal{R}}_k) = \omega_{\ddagger} \cdot \sum_{\hat{r}_k \in \hat{\mathcal{R}}_k} b_{\text{TRANSPORT}}(r_k, \hat{r}_k) \cdot \left(d_{\ddagger} - \min(d_{\ddagger}, \| \overset{\diamond,\dashv^{r_k}}{\mathbf{x}}_k - \overset{\diamond,\dashv^{\hat{r}_k}}{\mathbf{x}}_k \| \right) \tag{5.11.3}$$

$$L^{r_k}_{\text{TRANSPORT}}(\overset{\diamond,\dashv^{\mathfrak{r}}}{\mathbf{x}}_0, \overset{\diamond,\dashv^{r_k}}{\mathbf{x}}_k, \overset{\diamond,\dashv^{r_k}}{\mathbf{u}}_k, \hat{\mathcal{R}}_k, t_k, \overset{\diamond}{\Delta t}_k) = \min_{\mathbf{g} \in \mathcal{G}}(l_{\text{GOTO}}(d_{\mathbf{g}}(\overset{\diamond,\dashv^{r}}{\mathbf{x}}_k))) + \sum_{\mathbf{g} \in \mathcal{G}} l_{\ddagger}(d_{\mathbf{g}}(\overset{\diamond,\dashv^{r}}{\mathbf{x}}_k), \hat{\mathcal{R}}_k) \tag{5.11.4}$$

5.11.3. Erzeugte Schnittstellen und Daten

Diese Mission bietet viele Schnittstellen. So können die Paketlisten und Zielorte jeder Zeit extern, z. B. durch andere Mission, angepasst werden. Die Behälter können ebenfalls von außen durch die Be- oder Entladung verwaltet werden. Der Behältertyp kann gewechselt werden. Dabei hat der intelligente Behälter durch die automatischen Nachrichten den Vorteil, dass er das Schwarmverhalten verbessert.

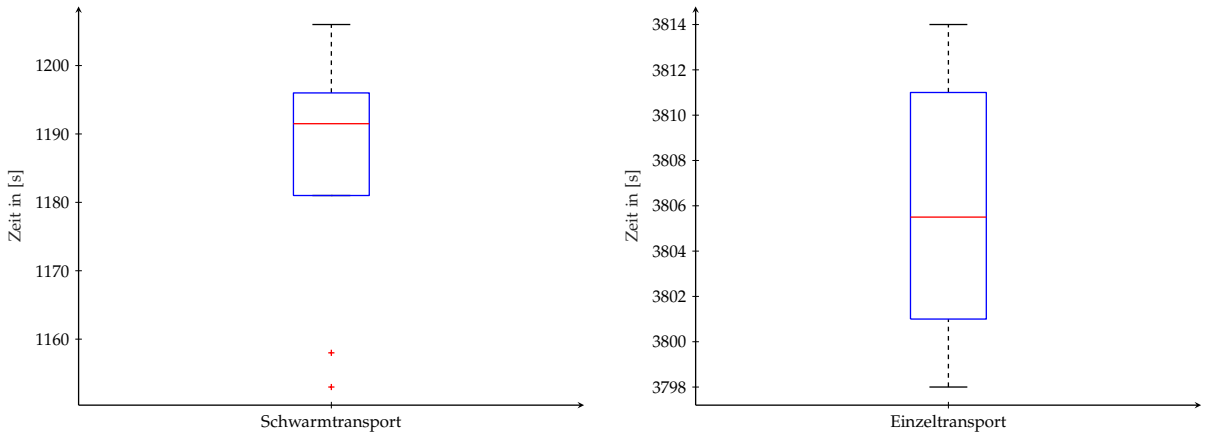
5.11.4. Experimente

Zur Qualitätsmessung der Mission wird der Schwarmtransport mit dem individuellen Transport verglichen. Dabei wird als ideelle Lösung die Zeit des individuellen Transports geteilt durch die Schwarmgröße angenommen.

Algorithmus 5.6.: Mission *Schwarmtransport*

Voraussetzung: $d_\ddagger$ Einflussbereich eines blockierenden Agenten, d_W Wartedistanz, Δ_t Zeit seit der letzten Aktualisierung, $\omega_\ddagger, \gamma_\mathbf{g}, \alpha_\mathbf{g}$ Parameter, t_W Wartezeitschwelle, $|\mathbf{g}_p|$ Anzahl der Pakete am Häufungspunkt $\mathbf{g}$, $|\mathcal{G}_p|$ absolute Anzahl aller zu transportierenden Pakete, $t_\mathbf{g}$ Wartezeit am Ziel $\mathbf{g}$

function CALCULATEPROTECTION($\mathbf{g}, \hat{\mathcal{R}}_k$)
  $s \leftarrow 0$
  for all $\hat{r}_k \in \hat{\mathcal{R}}_k$ **do**
    if $b_{\text{TRANSPORT}}(r_k, \hat{r}_k)$ **then**
      $d_\mathbf{g}^{\hat{r}} \leftarrow \| {}^{\diamond,\dashv^{\hat{r}_k}}\mathbf{x}_k - \mathbf{g}\|$
      if $d_\mathbf{g}^{\hat{r}_k} < d_\ddagger$ **then**
        $s \leftarrow s + d_\ddagger - d_\mathbf{g}^{\hat{r}_k}$
  return $s_\ddagger \leftarrow s \cdot \omega_\ddagger$
function UPDATEWAITINGTIME($\mathbf{g}, \Delta_t, \hat{\mathfrak{R}}$)
  $d_\mathbf{g} \leftarrow \| {}^{\diamond,\dashv^{r}}\mathbf{x} - \mathbf{g}\|$
  if $d_\mathbf{g} < d_\ddagger \vee d_\mathbf{g} > d_W$ **then**
    $t_\mathbf{g} \leftarrow t_\mathbf{g} - \Delta_t$
  else
    $i \leftarrow 0, j \leftarrow 0$
    for all $\hat{r} \in \hat{\mathfrak{R}}$ **do**
      if $b_{\text{TRANSPORT}}(r, \hat{r})$ **then**
        $d_\mathbf{g}^{\hat{r}} \leftarrow \| {}^{\diamond,\dashv^{\hat{r}}}\mathbf{x}_k - \mathbf{g}\|$
        if $d_\mathbf{g}^{\hat{r}} > d_\ddagger \wedge d_\mathbf{g}^{\hat{r}} < d_W$ **then**
          $i \leftarrow i + 1$
        $j \leftarrow j + 1$
    if $i > 0$ **then**
      if $t_\mathbf{g} < t_W$ **then**
        $t_\mathbf{g} \leftarrow t_\mathbf{g} + \Delta_t$
      else
        $t_\mathbf{g} \leftarrow t_\mathbf{g} + \frac{i}{j} \cdot \Delta_t$
    else
      $t_\mathbf{g} \leftarrow t_\mathbf{g} - \Delta_t$
  return $t_\mathbf{g} \leftarrow \max(t_\mathbf{g}, 0)$
function CALCULATEATTRACTIVENESS($\mathbf{g}$,$|\mathcal{G}_p|$)
  if isAccumulationPoint($\mathbf{g}$) **then**
    return $a_\mathbf{g} \cdot \frac{|\mathbf{g}_p|}{|\mathcal{G}_p|} - (\max(0, t_\mathbf{g} - t_W) \cdot \gamma_\mathbf{g})$
  return $a_\mathbf{g} - (\max(0, t_\mathbf{g} - t_W) \cdot \alpha_\mathbf{g})$



(a) Schwarmtransport.

(b) Einzeltransport.

Abbildung 5.10.: Laufzeit der Transportmissionen für einen Schwarm- und Einzeltransport.

Experiment 5.11.1: *In einem Szenario mit 24 Paketen, aufgeteilt auf zwei Häufungspunkte mit 8 und 16 Paketen, und zwei Ablageorten, von denen einer näher zu beiden Häufungspunkten liegt, sodass dieser von den Agenten zunächst favorisiert wird, werden fünf Roboter den Schwarmtransport ausführen und die Zeit wird gemessen.*

Experiment 5.11.2: *Im gleichen Szenario wie in Experiment 5.11.1 wird ein Agent den individuellen Transport ausführen, um eine Vergleichszeit zu erhalten.*

Die Experimente 5.11.1 und 5.11.2 werden jeweils zehnmal wiederholt und die Ergebnisse werden in Abbildung 5.10 veranschaulicht. Die Parameteridentifikation ist in der Abschlussarbeit Schulz [165] durchgeführt worden.

5.11.5. Analyse

Aus den in Abbildung 5.10a und 5.10b visualisierten Missionslaufzeiten ist ersichtlich, dass der Schwarmtransport im Mittel 1186 s und der Einzeltransport 3806 s benötigt. Somit reduziert die Mission die Laufzeit auf 31,16 % der ursprünglichen Zeit. Jedoch ist ebenfalls deutlich das die theoretische Minimalzeit von $\frac{3806\,\mathrm{s}}{5} = 761\,\mathrm{s}$ deutlich überschritten wird. Diese Abweichung ist auf die Dezentralität des Ansatzes zurückzuführen, denn zunächst steuern alle Agenten denselben Paketstapel an und werden erst später aufgrund der Wartezeit den Paketstapel wechseln. Somit bedarf es einer Anfangsphase, bis ein emergentes Verhalten erfolgt und die Agenten selbstständig die Abhol- und Ablageorte automatisch nach aktueller Auslastung auswählen. Weiterhin ist dies ein rein theoretischer Wert, da ein einzelner Agent nie bei der Aufnahme oder Ablage eines Paketes gestört wird und stets zum nächsten Ablageort fahren kann.

5.11.6. Zusammenfassung und Referenzen

Es ist deutlich erkennbar, dass die Lösung nicht optimal ist, allerdings auch nicht deutlich davon abweicht. Dies in Kombination mit der nahezu vollständigen Dezentralität

und der daraus resultierenden Robustheit ist ein gutes Ergebnis. Weitere Analysen sind in der Abschlussarbeit Schulz [165] aufgeführt.

5.12. Mission: Rendezvous

Neben den Einzelmissionen für die Agenten und dem Schwarmtransport, bei dem neben der Kollisionsvermeidung eine gewisse Rücksichtnahme implementiert ist, erzielt ein Rendezvous eine geplante Interaktion zwischen ausgewählten Agenten oder anderen Elementen in der Umwelt. Es wird dabei im Allgemeinen zwischen drei möglichen Rendezvoustypen unterschieden. Erstens gibt es örtliche Rendezvous, bei dem einem oder mehreren Agenten analog zur Zielansteuerungsmission (siehe Abschnitt 5.8) ein Ort des Treffens vorgegeben wird. Zweitens existieren zeitliche Rendezvous. Dabei wird ausschließlich die Zeit eines Treffens festgelegt und die Agenten wählen auf Basis der Entfernung zueinander und der verfügbaren Restzeit bis zum Treffen einen geeigneten Ort. Schließlich bleibt noch die Kombination der vorangegangenen Missionen, die sowohl zeitlichen als auch örtlichen Rendezvous. Die Einsatzgebiete dieser Mission können ebenfalls in der Logistik sein, bei der beispielsweise ein Medikament oder eine Ausrüstung zu einem Zeitpunkt bei sich dynamisch verhaltenden Rettungskräften ankommen muss, ohne einen Übergabeort zu spezifizieren. Außerdem lassen sich mit den Rendezvous Transportketten z. B. in Überschwemmungsgebieten realisieren (siehe Abschnitt 5.13). Die zeitliche Komponente ermöglicht es zuvor anderen Missionen nachzugehen.

5.12.1. Abhängigkeiten

Für die Bestimmung des Abstandes zu dem jeweiligen Rendezvouspartner sind die Modelldaten des anderen unabdingbar, woraus konsequentermaßen der Zugriff auf den Modelldatencontainer notwendig wird. Ferner wird die eigene Position in Form des Positionsdatencontainers benötigt. Zusätzliche Kommunikation ist ausschließlich zur Aktivierung der Mission notwendig.

5.12.2. Funktionsweise und Formalisierung

Rendezvous können als Verabredungen zu einer bestimmten Zeit $t_{\odot}$ und an einem bestimmten Ort $\mathbf{g}$ definiert werden. Wird kein Ort spezifiziert, wird $\mathbf{g} = \mathbf{x}^{\hat{r}_k}$ auf die Position des Rendezvouspartners gesetzt. Sind mehr als zwei Agenten an einem Rendezvous beteiligt, dann wird der Mittelpunkt zum Ziel. Dieses Ziel wird anschließend mit Hilfe der Zielansteuerungsmission (siehe Abschnitt 5.8) angefahren. Das Hauptmerkmal der Mission ist die Definition einer Zeitfunktion, die die Termintreue erzeugt. Dies kann durch eine binäre Sprungfunktion $\mathfrak{z}(t) \in \{0, 1\}$ zum passenden Zeitpunkt oder durch ein kontinuierliches Kostenwachstum $z(t) \in [0, 1]$ erreicht werden, um den Agenten frühzeitig in Richtung des Treffpunktes zu führen. Der frühzeitige Einfluss des Rendezvous kann die gewählte Sekundärmission negativ beeinflussen, falls komplementäre Ziele verfolgt werden. Die Kostenfunktion basiert auf der Entfernung zum

Ziel $d_{\mathbf{g}}$, der Durchschnittsgeschwindigkeit des Roboters $\bar{v}$, einer Sicherheitsspanne $\epsilon_{\bar{v}} \in \mathbb{R}_0^+$ zur Berücksichtigung von Hindernissen auf dem Weg und einem Anpassungsparameter $\gamma_{\bar{v}}$.

$$z(d_{\mathbf{g}}(\cdot), t_k) = \left(\min(\frac{d_{\mathbf{g}} \cdot (1+\epsilon_{\bar{v}})}{\bar{v} \cdot (t_{\odot} - t_k)}, 1)\right)^{\gamma_{\bar{v}}} \tag{5.12.1}$$

$$\mathfrak{z}(d_{\mathbf{g}}(\overset{\diamond,\dashv^r}{\mathbf{x}}_k), t_k) = \begin{cases} 1, \text{ falls } z(d_{\mathbf{g}}(\mathbf{x}_k^r), t_k) = 1 \\ 0, \text{ sonst} \end{cases} \tag{5.12.2}$$

$$L^{r_k}_{\text{RENDEZVOUS}}(\overset{\diamond,\dashv^{\mathfrak{r}}}{\mathbf{x}}_0, \overset{\diamond,\dashv^{r_k}}{\mathbf{x}}, \overset{\diamond,\dashv^{r_k}}{\mathbf{u}}_k, \hat{\mathcal{R}}_k, t_k, \overset{\diamond}{\Delta t}_k) = l_{\text{GOTO}}(d_{\mathbf{g}}(\overset{\diamond,\dashv^r}{\mathbf{x}}_k)) \cdot z(d_{\mathbf{g}}(\overset{\diamond,\dashv^r}{\mathbf{x}}_k), t_k) \tag{5.12.3}$$

$$\mathfrak{L}^{\mathfrak{r}}(T^{\mathfrak{r}}_{\overset{\diamond,\dashv^{\mathfrak{r}}}{\mathbf{x}}_0}, \hat{\mathfrak{R}}, \mathfrak{t}, \Delta\mathfrak{t}) = \sum_{k=1}^{K} L^{r_k}_{\text{RENDEZVOUS}}(\overset{\diamond,\dashv^{\mathfrak{r}}}{\mathbf{x}}_0, \overset{\diamond,\dashv^{r_k}}{\mathbf{x}}, \overset{\diamond,\dashv^{r_k}}{\mathbf{u}}_k, \hat{\mathcal{R}}_k, t_k, \overset{\diamond}{\Delta t}_k) \tag{5.12.4}$$

Das Produkt der Zeitfunktion z oder $\mathfrak{z}$ und der Zielansteuerung l_{GOTO} definiert die resultierende Funktion $L^{r_k}_{\text{RENDEZVOUS}}$. Diese Funktion wird in Abbildung 5.11a veranschaulicht.

5.12.3. Erzeugte Schnittstellen und Daten

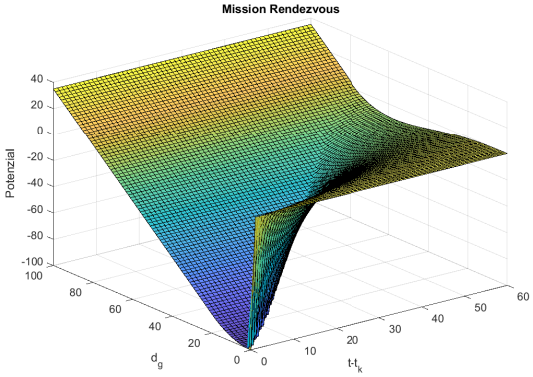
Die Sekundärmission, die zunächst verfolgt wird, kann frei gewählt und verändert werden. Ebenso lassen sich die Ziele und Rendezvouspartner extern anpassen. Ansonsten wird nur eine Indikatorvariable bereitgestellt, die das Erreichen der Zielposition anzeigt und bei der diskreten Variante, ob die Mission aktiv ist. Weitere Interaktionsmöglichkeiten sind momentan nicht implementiert, jedoch bildet diese Mission erneut die Basis einer weiteren Mission. Dadurch lässt sich das Konzept der modularen, rekombinierbaren und geschachtelten Missionen deutlich erkennen.

5.12.4. Experimente

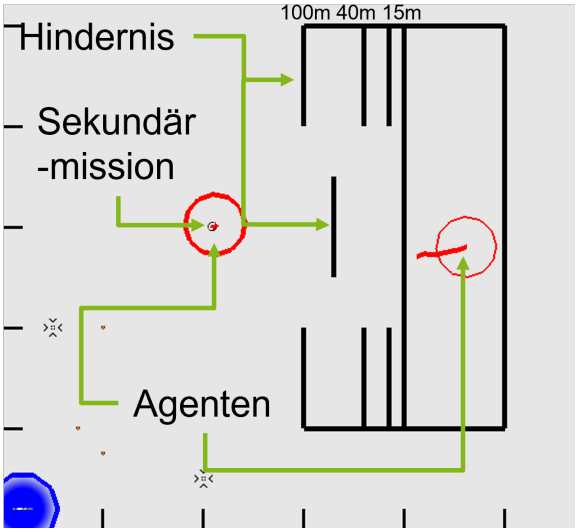
Ziel der durchgeführten Experimentreihe ist die Bestimmung eines geeigneten $\epsilon_{\bar{v}}$ bei gegebenem $\gamma_{\bar{v}} = 6$ für Umgebungen mit Hindernissen. Dabei wird eine mittlere Geschwindigkeit von $\bar{v} = \frac{2\overset{\dagger,\dashv}{v}_+}{3} = 3\,\mathrm{m\,s^{-1}}$ angenommen (siehe Kapitel 6).

Experiment 5.12.1: *Es soll ein Rendezvous in 150 s zwischen zwei Agenten stattfinden, die 250 m voneinander entfernt sind. Jedoch kann sich nur ein Agent bewegen, da der andere in einer Hindernisstruktur gefangen ist (siehe Abbildung 5.11b). Zunächst erhält der freie Agent ein Zielansteuerungsmission auf der Startposition als Sekundärmission. Die Zeiten, ab der sich der Agent von seiner Sekundärmission löst und sich auf 100 m, 40 m und 15 m nähert, sollen gemessen werden. Aufgrund des trennenden Hindernisses und dessen Einflussradius wird ein Abstand von unter 15 m als erreicht betrachtet. Diese Zeit wird mit der Vorgabe von 150 s als Zielzeit gemessen.*

Das Besondere an diesem Experiment ist, dass es die Robustheit des Ansatzes untersucht, weil der erste Agent einer konstanten Fehleinschätzung unterliegt. Zum einen



(a) Darstellung der Gesamtkostenfunktion $L^{r_k}_{\text{RENDEZVOUS}}$ über den Abstand zum Ziel $d_{\mathbf{g}}$ in Metern und der verbleibenden Zeit $t - t_k$ in Sekunden.



(b) Darstellung der Testwelt für das Experiment 5.12.1. Sie besteht aus zwei Agenten, die durch Hindernisse getrennt sind und sich nicht erreichen können. Der erste Agent, dessen Zeit gemessen wird, befindet sich 200 m von dem Trennstreifen entfernt und absolviert eine Zielansteuerung als Sekundärmission. Es liegt ein unbekanntes 100 m langes Hindernis auf seinem direkten Weg zum anderen Agenten.

Abbildung 5.11.: Visualisierung der Rendezvouskostenfunktion und ihre Anwendung in unbekannten Umgebungen.

Tabelle 5.6.: Bestimmung des Reservefaktors $\epsilon_{\bar{v}}$ für unbekannte Umgebungen.

$\epsilon_{\bar{v}}$	Startzeit	100 m	40 m	15 m
0	87 s	120 s	158 s	191 s
0,1	78 s	112 s	173 s	188 s
0,2	64 s	111 s	137 s	154 s
0,3	56 s	100 s	129 s	147 s
0,4	51 s	100 s	126 s	142 s

nimmt er an, dass er sich auf der halben Wegstrecke mit dem anderen Agenten trifft und zum anderen kennt er das in der Mitte liegende Hindernis nicht, welches deutlich seinen Prädiktionshorizont überschreitet. Somit wird ein Wert für $\epsilon_{\bar{v}}$ erforscht, der zwei oft auftretende Fehleinschätzungen berücksichtigt. Die gemessenen Werte sind in Tabelle 5.6 aufgeführt. Weitere Experimente werden in der Abschlussarbeit Grabenschröer [47] erläutert, dort wird auch die Bestimmung des Parameters $\gamma_{\bar{v}} = 6$ beschrieben.

5.12.5. Analyse

Aus der Tabelle 5.6 ist ersichtlich, dass der Agent für $\epsilon_{\bar{v}} = 0$ zu lange an seiner Startposition verweilt und die Sekundärmission wahrnimmt. In den verbleibenden 63 s kann der Agent ohne Berücksichtigung des Hindernisses nur maximal 189 m der 200 m zurücklegen. Auch mit der Schwelle von 15 m ist dies zu knapp, denn er erreicht das Ziel verspätet nach 191 s. Die Zeit zum pünktlichen Erreichen des Schwellwerts erfolgt erst ab $\epsilon_{\bar{v}} = 0{,}3$, welches nun als Standardwert gewählt wird. Der Puffer beträgt zwar nur 3 s, jedoch unterliegt der Agent einer stetigen Fehlannahme. Daher wird dies als gutes Mittel zwischen Zeitreserve durch unbekannte Hindernisse und Vorkommnisse sowie einer hinreichenden Effizienz angesehen, sodass der Agent seine Sekundärmission eine angemessene Zeit lang verfolgt.

5.12.6. Zusammenfassung und Referenzen

Die Rendezvous bilden eine weitere Kategorie an Basismissionen, die zur Verwendung in komplexeren Missionen gedacht sind. Zudem führen sie erstmalig tatsächliche Interaktionen der Agenten ein. Weiterhin ist die Mission von praktischer Relevanz um die zeitgenaue Versorgung der Rettungskräfte in unbekannten Gebieten zu bewerkstelligen, ohne dabei stets jeden Agenten einzeln zu überwachen oder die Transportmissionen stets zur richtigen Zeit zu starten.

5.13. Mission: Kettentransport

Als Alternative zum Schwarmtransport (siehe Abschnitt 5.11) können, analog zu Menschenketten für Sandsäcke bei Überschwemmungen, Transportketten mit Agenten realisiert werden. Dazu werden die Rendezvous (siehe Abschnitt 5.12) verwendet, sodass

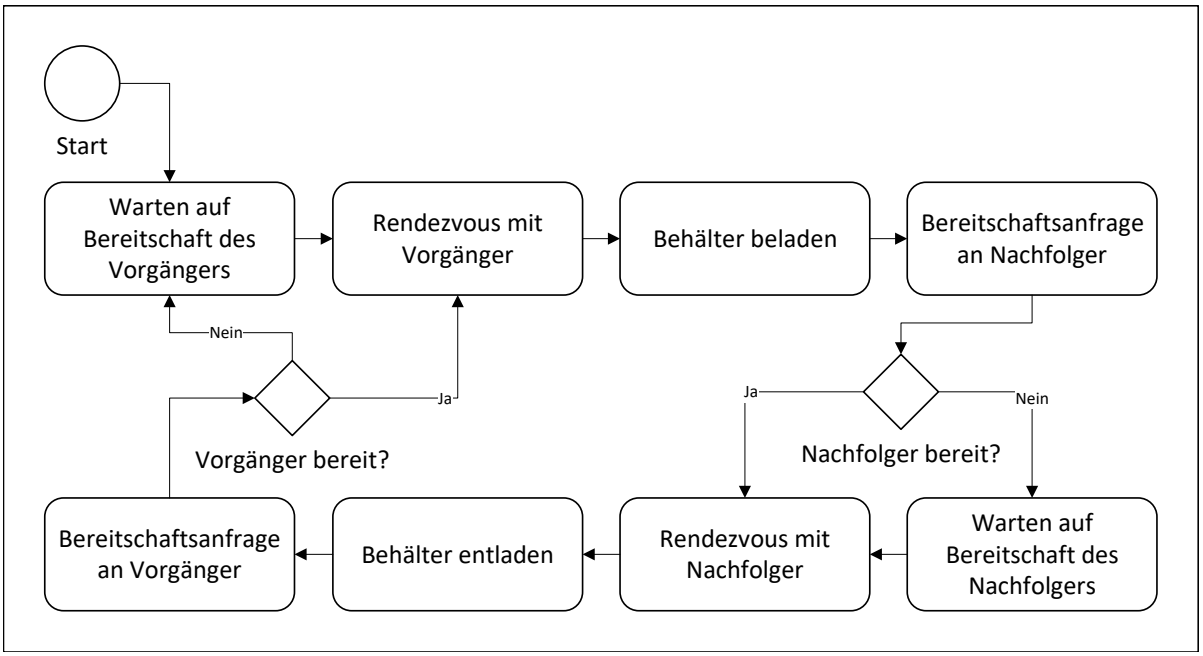


Abbildung 5.12.: Darstellung des Flussdiagramms der Rendezvousmission.

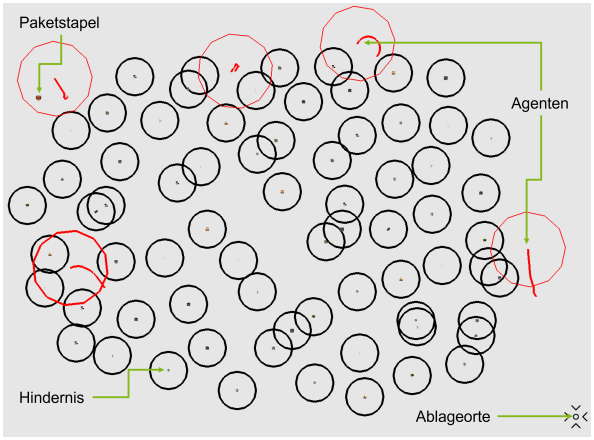
sich die Agenten abwechselnd mit ihrem Vorgänger und Nachfolger in der Transportkette treffen und die Pakete bzw. Waren austauschen. Das Ziel dieser Transportvariante ist die implizite Optimierung der Transportroute, da die Agenten aufgrund ihrer internen modellprädiktiven Regelung die zu fahrende Strecke lokal optimieren. Dadurch kann ausgehend von einer beliebigen Startverteilung ein lokales Pfadoptimum gefunden werden. Ferner können auch Transporte durch komplexe Hindernisformen wie Labyrinthe oder U-Formen effizient gelöst werden ohne stets für jeden Agenten einen globalen Pfad zu berechnen. Die Idee stammt von dem Transportverhalten der Ameisen oder dem Sandsacktransport in Überflutungsgebieten.

5.13.1. Abhängigkeiten

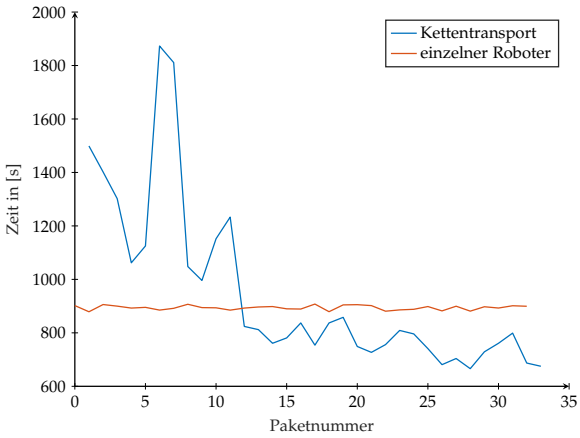
Die Anforderungen an den Datenzugriff unterscheiden sich nicht von denen der Rendezvous. Allerdings muss nun aktiv mit dem Vorgänger und Nachfolger kommuniziert werden, um anzufragen, ob diese für ein Rendezvous bereit sind oder noch in einem Rendezvous mit dem für sie anderen Partner sind. Dazu wird der Zugriff auf das Kommunikationsmodul notwendig.

5.13.2. Funktionsweise und Formalisierung

Das Verhalten des zugrunde liegenden Algorithmus kann mittels des folgenden Flussdiagramms 5.12 beschrieben werden. Zu Beginn der Mission wird auf die Bereitschaft des Vorgängers gewartet. Ist diese gegeben, wird eine Rendezvous mit diesem gestartet. Bei erfolgreicher Annäherung entlädt der Vorgänger seinen Behälter und dessen Nachfolger startet die Beladung bzw. die Transportmission zur Beladung, weil die Annäherung natürlich unter Wahrung der Schutzabstände geschieht. Anschließend



(a) Darstellung eines Kettentransportszenarios mit sehr vielen kreisförmigen Hindernissen, die den Weg versperren, sodass ein einzelner Roboter stets einen neuen Weg hindurch finden muss.



(b) Paketauslieferungszeiten.

Abbildung 5.13.: Visualisierung der Kettentransportmission.

wird der Nachfolger des nun beladenen Agenten auf Bereitschaft für das folgende Rendezvous angefragt. Ist diese gegeben, wird eine neue Rendezvousmission mit dem Nachfolger gestartet. Bei einem Treffen wird das Paket entladen und nun beginnt der Kreislauf erneut mit einer Bereitschaftsanfrage an den Vorgänger. Die Bestimmung des Vorgängers und Nachfolgers wird anhand der Identifikationsnummer der Kommunikationsmodule durchgeführt. Ausnahmen bilden der erste und der letzte Agent der Kette. Diese müssen sich entweder mit der ausgebenden und aufnehmenden Station absprechen oder jeweils zum nächsten Paket bzw. Ablageort fahren, ohne auf eine Nachricht zu warten.

5.13.3. Erzeugte Schnittstellen und Daten

Diese Mission ist an sich abgeschlossen und bietet in der aktuellen Implementierung keine weiteren Schnittstellen zur Wiederverwendung. Denkbar ist ein Paketzähler pro Agent. Damit kann der Abschluss der Mission leicht festgestellt werden. Ebenfalls wäre ein Zugriff auf die letzten Be- und Entladestellen nützlich, um eine Pfadoptimierung vorzunehmen und ein stationäres Verhalten detektieren zu können.

5.13.4. Experimente

Zur Beurteilung der Sinnhaftigkeit und der Funktionsfähigkeit eines Kettentransports, wird nun ein besonders komplexes Szenario für den Transport gewählt. Bei der Erprobung des Schwarmtransports wird gezeigt, dass dieser sogar unter idealen Bedingungen für den Einzeltransport gut abschneidet. Der Kettentransport wird im Allgemeinen aufgrund der Paketübergabe zwischen den Agenten deutlich mehr Zeit für die Absolvierung der Mission in Experiment 5.11.1 benötigen. Die Stärke des Kettentransports liegt darin, dass der Transportweg implizit durch das stetige Pendeln der Agenten

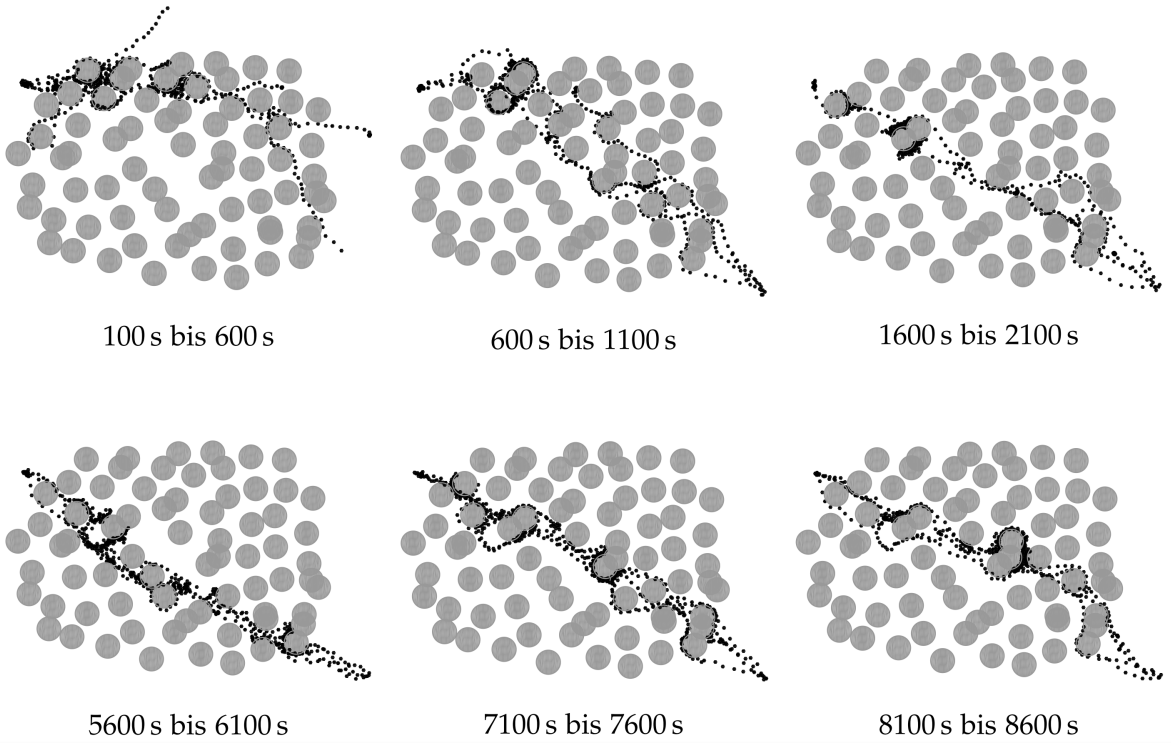


Abbildung 5.14.: Entwicklung der Trajektorien über die Zeit, entnommen aus der Abschlussarbeit Grabenschröer [47]. Die grauen Kreise bilden die Hindernisse aus Abbildung 5.13a ab. Die schwarzen Punkte sind die Zustände der fünf Agenten zu jeweiligen Abtastpunkten im angegebenen Intervall.

über einen längeren Zeitraum hinweg optimiert wird. Demgegenüber verbrauchen einzelne Agenten mehr Zeit mit dem Finden einer dynamischen optimalen Route. Folglich wird ein Gebiet mit einer komplexen Hindernisstruktur für das Experiment gewählt.

Experiment 5.13.1: *In einem Szenario mit 34 Paketen auf der einen Seite eines zufälligen Hindernisfeldes, bestehend aus kreisförmigen Hindernissen, und einem Ablageort auf der Anderen (siehe Abbildung 5.13a) sollen die Einzeltransportmission und der Kettentransport mit fünf Agenten durchgeführt werden und die Transportzeit pro Paket gemessen werden.*

Da jeder Paketdurchlauf unabhängig ist, muss das Experiment nicht mehrfach wiederholt werden. Die Ergebnisse werden in Abbildung 5.13b visualisiert und der Verlauf der Trajektorien des Kettentransports wird in Abbildung 5.14 dargestellt. Weitere Experimente sind in der Abschlussarbeit Grabenschröer [47] ausführlich beschrieben.

5.13.5. Analyse

Abbildung 5.13b hebt deutlich hervor, dass der Kettentransport, ebenso wie eine Menschenkette auch, erst einen passenden Transferrhythmus entwickeln muss, bevor er effizient arbeitet. Diese Konvergenz ist deutlich am Kurvenverlauf der Transportzeiten pro Paket zu erkennen. Im Vergleich dazu erzielt der Einzeltransport eine nahezu

konstante Transferzeit von im Mittel 878 s. Die Hindernisse zwingen den Agenten aufgrund des zufallsbasierten Reglers stets neue Pfade durch die Hindernisse zu wählen, wodurch die Wahrscheinlichkeit zur Wahl des optimalen Pfades gering ist. Im Kontrast dazu visualisiert Abbildung 5.14 die Entwicklung der Trajektorien der Agenten, die zunächst suboptimal ist. Das stete Wechseln zwischen den Transportpartnern erzielt eine lokale Optimierung der Pfadstücke, die im Gesamtem in dem optimalen Pfad konvergieren. Die zusätzlichen Zeiten durch das Aufnehmen und Ablegen lassen sich nicht vermeiden und verhindern das Erreichen der optimalen Lösung. Daraus ergibt sich, dass sich diese Mission vor allem für dynamische und unbekannte Gebiete mit einer komplexen Hindernisstruktur eignet. Zudem ist diese Mission bevorzugt für einzelne große Paketstapel zu verwenden, da bei mehreren Stapel ein erneutes Ausrichten der Pfade notwendig ist, bis diese erneut konvergieren. Für stark verstreute Pakete und Ablageorte eignet sich eher die Schwarmtransportmission.

5.13.6. Zusammenfassung und Referenzen

Analog zum Schwarmtransport ist diese dezentrale Lösung nicht mit einem global optimierten Routenplan vergleichbar. Das stetige Aufnehmen und Ablegen der Pakete sowie die häufigen Richtungswechsel und Rendezvousanfahrten kosten gegenüber einer mit maximaler Geschwindigkeit gefahrenen Strecke Zeit. Jedoch bietet die sich implizit optimierende Strategie den Vorteil, dass auch komplexe Hindernisse ohne explizite Pfadvorgabe überwunden werden. Einzelne Agenten können an den Hindernissen aufgrund der beschränkten Sichtweite scheitern oder benötigen eine weitere zeitintensive Routenplanungsmission (siehe Abschnitt 5.15). Nach einer Einlaufphase konvergiert der Kettentransport und ermöglicht effiziente Rundenzeiten.

5.14. Mission: Flächenanalyse

Nach der Zielansteuerung und der Pfadfolge wird nun die präzise Flächenanalyse die grundlegende Idee weiterführen. Mit Hilfe der Flächenanalyse lassen sich alle beliebigen einfachen Polygone abfahren. Diese Fähigkeit wird für die Bodenanalyse in Katastrophengebieten, für die Minenräumung oder auch zum autonomen Mähen und Mulchen großer Felder benötigt. Die Grundlage dieser Mission bildet im Sinne der Modularität die Pfadfolgemission, indem die einfachen Polygone zunächst in monotone Polygone zerlegt werden und schließlich trianguliert werden. Die resultierenden Dreiecke werden in Pfadsegmente mit gegebener Breite aufgeteilt. Die Pfade können anschließend mit der Pfadfolgemission abgefahren werden.

5.14.1. Abhängigkeiten

Diese Mission hat neben der Vorgabe des abzufahrenden Polygons dieselben Abhängigkeiten wie die Pfadfolgemission (siehe Abschnitt 5.9), weil sie auf dieser aufbaut und nur um die Pfadberechnung ergänzt wird.

5.14.2. Funktionsweise und Formalisierung

Der implementierte Algorithmus ist auf einfache Polygone beschränkt. Im Allgemeinen gilt, dass jedes einfache Polygon trianguliert werden kann. Es existieren sogar Ansätze zur Triangulierung einfacher Polygone in einer Laufzeitkomplexität von $\mathcal{O}(n)$ (vgl. [22] und [171]). Aufgrund der extrem hohen Komplexität der Implementierung und der hohen Laufzeitkonstante existiert bis heute noch keine öffentliche Implementierung. Die gewählte Implementierung (siehe Algorithmus 5.7) arbeitet wie folgt: Als erstes werden einfache Polygone $\mathbf{P}$ mittels eines „Sweep Line"-Verfahrens in monotone Polygone $\mathbf{P}_m$ unterteilt. Diese können in Dreiecke mittels eines weiteren „Sweep Line"-Verfahrens zerlegt werden. Die Algorithmen stammen von Berg u. a. [13], jedoch müssen sie angepasst werden, sodass sie nicht nur die Diagonalen, sondern die Polygone und Dreiecke zurückgeben. Von den Dreiecken Δ werden die Basen $\mathbf{b}$ und das Lot $\mathbf{l}$ zum Eckpunkt bestimmt. Schließlich wird die Basis $\mathbf{b}$ entlang des Lot $\mathbf{l}$ mit Schrittweite $d_\mathbf{l}$ verschoben und mit den Schenkeln geschnitten. Die resultierenden Strecken $\mathbf{s}$ werden mit der Schrittweite $d_\mathbf{g}$ zu Zwischenpunkten $\mathbf{g}$ abgetastet. Der Gesamtpfad $\mathfrak{G}$ wird an die Pfadfolgemission weitergegeben.

Algorithmus 5.7.: Mission *Flächenanalyse*

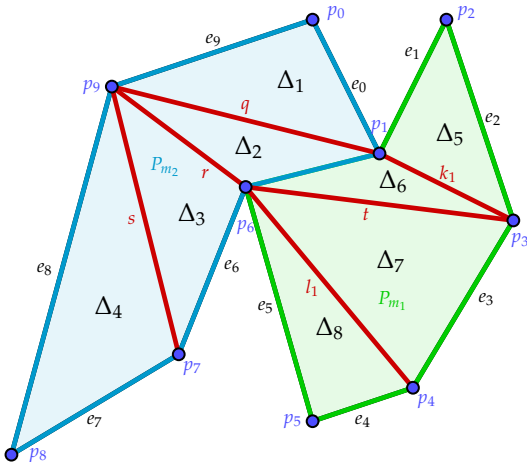
Voraussetzung: $\mathbf{P}$ einfaches Polygon, $d_\mathbf{l}$ Lotabstand zwischen den Bahnen der Pfade, $d_\mathbf{g}$ Zwischenzielabstand

```
1:  function calculatePathlist(P)
2:      G ← ∅
3:      P_m ←calculateMonotonousPolygonSet(P)
4:      for all P_m ∈ P_m do
5:          D ←triangulate(P_m)                          ▷ Dreiecksmenge
6:          for all Δ ∈ D do
7:              b ←calculateBase(Δ)                      ▷ Dreiecksbasis
8:              l ←calculatePerpendicular(Δ, b)          ▷ Dreieckslot
9:              S ←calculateSegments(Δ, b, l, d_l)       ▷ Streckenmenge
10:             for all s ∈ S do
11:                 G' ← sampleSegement(s, d_g)
12:             G ← G ∪ G'
        return G
```

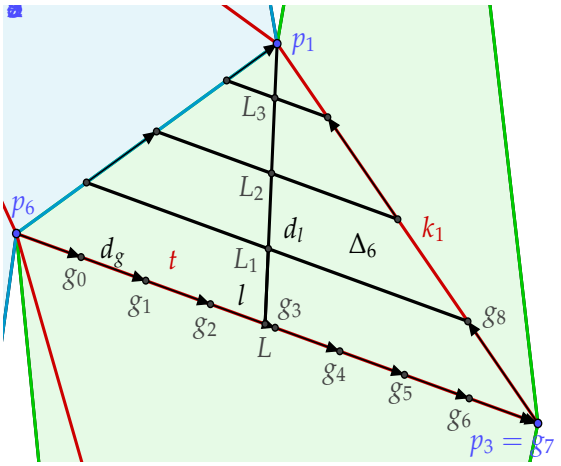
Weitere Einzelheiten zu den Algorithmen sind in der Abschlussarbeit Shaker [169] und in dem Buch Berg u. a. [13] ausgeführt.

5.14.3. Erzeugte Schnittstellen und Daten

Diese Mission bietet einen prozentualen Fortschritt und eine Indikatorvariable, die den Abschluss der Mission anzeigt. Weitere Schnittstellen sind nicht implementiert.



(a) Darstellung der Polygonzerlegung über monotone Polygone zu Dreiecken.



(b) Darstellung der Dreieckszerlegung hinzu einer Pfadliste.

Abbildung 5.15.: Visualisierung der Polygonzerlegung in einen Gesamtpfad.

Tabelle 5.7.: Ergebnisse der Parameteranalyse der Flächenanalysemission.

Durchführung	d_g	d_l	$\lvert\mathfrak{G}\rvert$	Minimum	Maximum	$\bar{t}$	σ_t
1	30 m	30 m	44	824,14 s	864,93 s	849,30 s	17,96 s
2	15 m	30 m	77	863,43 s	927,41 s	906,20 s	31,28 s
3	30 m	15 m	80	1375,25 s	1485,43 s	1426,21 s	45,35 s
4	15 m	15 m	141	1424,44 s	1463,12 s	1449,47 s	17,72 s
5	50 m	15 m	52	1426,01 s	1469,34 s	1454,44 s	20,11 s
6	100 m	15 m	34	1446,68 s	1534,22 s	1438,51 s	37,06 s
7	3 m	15 m	656	1466,62 s	1527,58 s	1489,20 s	27,27 s

5.14.4. Experimente

Mit Hilfe des folgenden Experimentes soll der Einfluss der Parameter d_g und d_l analysiert werden.

Experiment 5.14.1: *Das einfache Polygon* $\mathbf{P} =<(500\,\text{m}, 700\,\text{m}), (600\,\text{m}, 500\,\text{m}), (700\,\text{m}, 700\,\text{m}), (800\,\text{m}, 400\,\text{m}), (650\,\text{m}, 150\,\text{m}), (500\,\text{m}, 100\,\text{m}), (400\,\text{m}, 450\,\text{m}), (300\,\text{m}, 200\,\text{m}), (50\,\text{m}, 50\,\text{m}), (200\,\text{m}, 600\,\text{m})>$ *(siehe Abbildung 5.15a) wird mit* $d_l = \{30\,\text{m}, 15\,\text{m}\}$ *und* $d_g = \{3\,\text{m}, 15\,\text{m}, 30\,\text{m}, 50\,\text{m}, 100\,\text{m}\}$ *trianguliert. Die benötigte Zeit zum Abfahren wird gemessen und jeder Durchlauf dreimal wiederholt.*

Das Ergebnis des Experiments wird in Tabelle 5.7 zusammengefasst. Dabei beschreibt $\bar{t}$ den Mittelwert und σ_t die Standardabweichung der Zeiten. Weiterhin gibt $\lvert\mathfrak{G}\rvert$ die Gesamtanzahl an Zwischenpunkten in der Pfadliste an. d_g ist der Abstand zwischen zwei konsekutiven Zwischenpunkten und d_l der Streckenabstand, bzw. die Schrittweite in Lotrichtung.

5.14.5. Analyse

Aus den Ergebnissen in Tabelle 5.7 geht hervor, dass die Reduktion des Abstandes $d_\mathbf{l}$ einen deutlichen Einfluss auf die Steigerung der Laufzeit besitzt, wohingegen die Steigerung der Anzahl an Zwischenpunkten $\mathbf{g}$ durch die Reduktion des Abstandes $d_\mathbf{g}$ einen kaum messbaren Einfluss zeigt. Dies ist logisch, da die Agenten ab einer gewissen Dichte an Zwischenpunkten keinen Nutzen aus weiteren Zwischenpunkten für die Berechnung der Trajektorie ziehen, da bereits der optimale Pfad gewählt wird. Demgegenüber erhöht die Verringerung von $d_\mathbf{l}$ die Anzahl an Strecken in den Dreiecken, wodurch die Gesamtstrecke zunimmt. Folglich ist $d_\mathbf{l}$ anwendungsspezifisch zu wählen. Als Beispiel dient die Schnittbreite eines Rasenmähers oder der Sensorradius eines Bodenanalysesystems.

5.14.6. Zusammenfassung und Referenzen

Die Flächenanalysemission formuliert die Zerlegung einfacher Polygone und wie diese systematisch abgefahren werden können. Verbesserte Algorithmen und Verfahren für beliebige zweidimensionale Polygone können aus der additiven Fertigung (vgl. [79]) oder aus der Pfadabdeckung (vgl. [182]) übernommen werden.

5.15. Mission: Globale Pfadplanung

Die bisher vorgestellten Missionen fokussieren vor allem lokale Aktionen und Entscheidungen im Umfeld der Agenten. Die Simulation ist momentan für eine Fläche von $25\,\mathrm{km}^2$ ausgelegt und der Optimierer bietet eine maximale Sichtweite von 90 m aufgrund seines Prädiktionshorizontes von 30 s und einer maximalen Geschwindigkeit der Agenten von $3\,\mathrm{m\,s}^{-1}$. Diese Sichtweite eignet sich gut, um lokale Hindernisse zu beachten. Jedoch ist sie ungeeignet, um komplexes urbanes Gebiet mit weitreichenden Sackgassen zu durchqueren oder durch Katastrophen erzeugte Hindernisschneisen zu umgehen. Somit werden beispielhaft zwei Ansätze präsentiert, um auch globale Pfadplanung mit den Missionen abzubilden.

5.15.1. Abhängigkeiten

Zur globalen Pfadplanung wird der eigene Zustand aus dem Positionsdatencontainer gelesen. Weiterhin müssen alle bekannten statischen Hindernisse aus dem Kartendatencontainer sowie aus dem Hinderniscontainer der Sensoren verarbeitet werden. Weitere Datenzugriffe und Abhängigkeiten sind nicht gegeben.

5.15.2. Funktionsweise und Formalisierung

Als globale Pfadplaner für die Mission wird sowohl der gitterbasierte optimale Planer A* (vgl. [102]) als auch das kontinuierlich arbeitende Verfahren RRT* (siehe Abschnitt 2.2.1.1 und vgl. [65]) verwendet. Für die Verfahren existieren Referenzimplementierungen, die an den Simulator angepasst werden. So arbeitet A* in der Regel mit

Tabelle 5.8.: Laufzeiten der Algorithmen.

Algorithmus	Minimum	Maximum	$\bar{t}$	σ_t
RRT*	5 ms	25 024 ms	4301,8 ms	7129,7 ms
A*	517 239 ms	589 236 ms	526 259,4 ms	15 311 ms

einem Hindernisgitter, in dem für jede Zelle einer der drei Zustände „belegt“, „frei“ oder „unbekannt“ gilt. Die Simulation basiert auf einer kontinuierlichen Welt, sodass diese diskretisiert werden muss. Der Zellzustand wird mit Hilfe der Summe der Hindernispotentiale bestimmt. Ist diese größer Null, so liegt ein Hindernis vor und der Zustand gilt als „belegt“. Bei dem kontinuierlichen Verfahren werden die erzeugten Strecken auf Schnittpunkte mit den Hindernissen überprüft, um kollisionsfreie Pfade zu garantieren. Die konkrete Umsetzung der Mission sowie die Bestimmung der Diskretisierungsschrittweite für den A* Algorithmus sind in der Abschlussarbeit Gerwins [46] ausgeführt. Dabei wird 25 m als Schrittweite festgelegt. Beide Verfahren generieren einen Pfad bestehend aus Zustandsknoten in einer Liste $\mathfrak{G}$, welche anschließend an die Pfadfolgemission weitergeleitet wird. Im Unterschied zur vorangegangenen Pfadfolge wird das Bremsverhalten aus der Kostenfunktion für alle Zwischenknoten bis zum Ziel entfernt, um mit möglichst hoher Geschwindigkeit das Ziel zu erreichen, ohne vor jedem Zwischenpunkt leicht abzustoppen, bevor der nächste Punkt forciert wird.

5.15.3. Erzeugte Schnittstellen und Daten

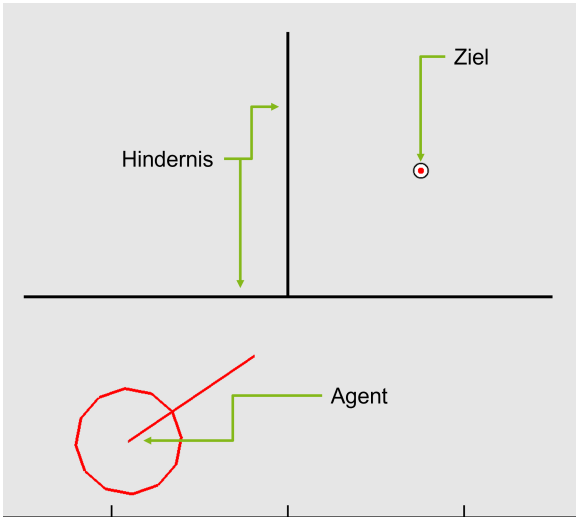
Diese Mission lässt sich bewusst mit Übergabe eines Zielpunktes in andere Missionen integrieren, um die Einsatzfähigkeit im gesamten Gebiet zu gewährleisten. In der künftigen Entwicklung ist es sinnvoll das bestimmte Gitter für A* über die Missionen hinweg zu behalten, um Rechenzeit zu sparen.

5.15.4. Experimente

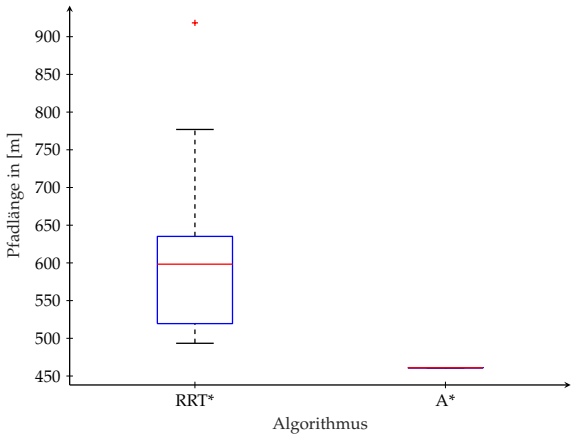
Mit Hilfe eines Experimentes soll untersucht werden, welcher globale Pfadplaner sich besser für den hier vorgestellten Kontext eignet. Dazu soll sowohl die Lösungsgüte als auch die Laufzeit betrachtet werden.

Experiment 5.15.1: *Es sei ein Szenario gegeben mit einem T-förmigen Hindernis, abgebildet in Abbildung 5.16a, in dem der Agent nicht mit lokaler Pfadplanung allein das Ziel erreichen kann. Dieser befindet sich unten links im Zentrum des roten Kreises und soll die Zielmarke oben rechts erreichen. Gemessen werden die Güte der Strecke in Metern und die benötigte Rechenzeit.*

Die Abbildung 5.16b stellt die Lösungsgüte dar. Dabei wird für jeden der beiden Algorithmen das Experiment zwanzigmal wiederholt. In Tabelle 5.8 beschreiben $\bar{t}$ den Mittelwert und σ_t die Standardabweichung der Zeiten.



(a) Darstellung des Szenarios für die experimentelle Auswertung der Algorithmen A* und RRT*.



(b) Lösungsgüte der berechneten Pfade.

Abbildung 5.16.: Visualisierung des Experimentes zur Auswertung der globalen Pfadplanung.

5.15.5. Analyse

Aus dem Experiment geht deutlich hervor, dass die deterministische Lösung von A* stets optimal für den diskreten Raum ist. Doch auch die RRT* Lösung generiert einen möglichen maximal doppelt so langen Pfad zum Ziel. Dieser ist deutlich suboptimaler, allerdings ist die benötigte Berechnungslaufzeit im Mittel um den Faktor 100 geringer. Aus dem Experiment und weiteren Analysen und Vergleichen in der Arbeit (vgl. [46]) lässt sich folgern, dass, im Sinne einer möglichst geringen Laufzeit, zunächst zufallsbasierte Ansätze ausgeführt werden sollten. Erzielen diese kein Ergebnis, so kann die Diskretisierung und der A* Algorithmus als Notfalllösung dienen. Dabei kann RRT* auch durch andere Verfahren ersetzt werden (vgl. [71], [196], [99] und [87]).

5.15.6. Zusammenfassung und Referenzen

Die vorgestellte globale Pfadplanungsmission rundet die Menge der verfügbaren Missionen ab und zeigt auf, wie trotz der gewählten Potenzialfunktionstechnik auch globale und optimale Missionen erzeugt werden können. Dabei ermöglicht der gewählte Ansatz die Vorteile einiger Methoden durch die Modularität zu kombinieren. Basierend auf dieser Menge lassen sich auch noch deutlich komplexere Missionen bilden. Weiterhin wird jedoch eine strikte Trennung zwischen lokaler Trajektorien- und globaler Pfadberechnung vermieden.

5.16. Weitere Missionen

Ergänzende Missionen, die mit Unterstützung einiger Abschlussarbeiten untersucht worden sind, sind:

1. Konsensmechanismen zur Entscheidungsfindung und Optimierung der Logistik (vgl. [119] und [127])

2. Energiegewahres Missionsmanagement und Ladestrategien der Agenten (vgl. [80])

3. Störsenderdetektion und -abwehr (vgl. [195])

4. Liniennetzwerke (vgl. [97])

Die Idee der Konsensmechanismen ist es dezentral eine gemeinsame Entscheidung zu finden, um z. B. einen einheitlichen Paketverteilungsplan für die Logistikmission zu generieren. Dazu werden verschiedene Algorithmen wie PAXOS (vgl. [91]) und RAFT (vgl. [30]) verwendet und mit der Laufzeit der dezentralen Lösung verglichen (vgl. [119] und [127]). Dabei ist ersichtlich, dass die globale Lösung nur für sehr statische Szenarien eine kürzere Laufzeit erzielt. Jedoch ist die fehlende Flexibilität bei sich ändernden Schwarmgrößen oder Paketorten derart nachteilig, dass der Ansatz nicht weiter verfolgt worden ist. Ein weiterer sehr relevanter Aspekt des Missionsmanagements ist die Betrachtung der verfügbaren Energie vor allem bei elektrischen Fahrzeugen. Dazu ist eine übergeordnete Mission entwickelt worden, die die Idee der Rendezvous in der Art wiederverwendet, dass zunächst eine übergebene andere Mission ausgeführt wird, sofern genügend Energie dafür vorhanden ist. Zeitgleich werden die Ladekapazitäten an verfügbaren Ladestationen sowie die benötigte Energie zur Erreichung dieser kontinuierlich überwacht. Bevor die Energie aufgebraucht ist oder falls nicht genügend Energie mehr für eine Mission vorhanden ist, wird analog zur Transportmission eine Ladesäule unter Betrachtung der Auslastung und Entfernung gewählt.
Einen vollständig anderen Bereich deckt die Entwicklung zur Detektion und Abwehr von Störsenderangriffen ab. Diese vor allem militärisch-relevante Mission kann ebenfalls in Katastrophengebieten mit radioaktiver Strahlung an Bedeutung gewinnen. Da die Strahlung die Kommunikation deutlich beeinträchtigt. Somit lassen sich anstatt der Störsender zentrale Emittierungsstellen erfassen bzw. schätzen und die Kommunikation aufrecht erhalten.

5.17. Missionsverteilung

Die zuvor vorgestellten Missionen werden entweder einzelnen Agenten explizit zugewiesen oder an alle ausgegeben. Jedoch werden vor allem bei großen Schwarmgrößen nicht alle Agenten dieselbe Mission ausführen, da dies zu ineffizient ist. Ebenso ist es unpraktikabel in solchen Schwärmen stets einzelne Agenten für Missionen zu selektieren. Stattdessen sollte der Schwarm anteilig nach frei wählbaren Vorgaben eine Menge an Missionen ausführen. Die Idee entstammt der autonomen Aufteilung verschiedener Insektenkolonien in Kämpfer, Späher und Sammler. So wechseln Ameisen selbstständig ihre Aufgaben zwischen Erkundung nach neuer Beute, Zerlegung bzw. Bekämpfung der Beute und dem Abtransport. Zur Umsetzung werden drei verschiedene Ansätze verfolgt. Zunächst wird eine manuelle Aufteilung vorgegeben (vgl. [185]).

Im zweiten Schritt wird die vorgegebene Aufteilung dynamisch in jeder Kommunikationsgruppe aufrecht erhalten. Schließlich können verschiedene Missionen ihren Bedarf an Agenten festlegen. So kann die Transportmission so viele Agenten wie vorhandene Pakete anfordern. Diese Erweiterung wird in das Missionsbehandlungsmodul (siehe Abschnitt 5.3) integriert. Folglich ist der Zugriff auf alle verfügbaren Komponenten eines Agenten gegeben. Die Verteilungsfunktion der Missionen auf die Agenten ist stets gleich und in Algorithmus 5.8 ausgeführt.

Algorithmus 5.8.: Missionsverteilung

Voraussetzung: $\mathcal{V}_\mathcal{M}$ Verteilungsliste bestehend aus Missions- und Anteilstupel, $\hat{\mathfrak{R}}$ angepasste Agentensvektornmodellmenge

function DISTRIBUTEMISSION($\mathcal{V}_\mathcal{M}, \tilde{\hat{\mathcal{R}}}$)

- $n_1 \leftarrow 0$ ▷ Bisher zugewiesene Agenten
- $n_2 \leftarrow 0$ ▷ Künftig zugewiesene Agenten
- SORTBYID($\hat{\mathfrak{R}}$)
- **for all** $(m, v_\%) \in \mathcal{V}_\mathcal{M}$ **do**
  - $n_r \leftarrow v_\% \cdot |\hat{\mathfrak{R}}|$ ▷ Anzahl der Agenten für die Mission m
  - $n_2 \leftarrow n_2 + n_r$
  - **for** $i \leftarrow n_1$ to $i = n_2$ **do**
    - ACTIVATEMISSION($\hat{\mathfrak{R}}_{[i]}, m$)
  - $n_1 \leftarrow n_1 + n_r$

Der Algorithmus sortiert zunächst die Agenten der Agentenliste $\hat{\mathfrak{R}}$ nach ihren Identifizierungsnummern. Eine direkte Zuweisung auf Basis der Nummer ist nicht möglich, weil diese nicht konsekutiv sein müssen. Anschließend wird abhängig von der prozentualen Verteilung $v_\%$ der Mission $m \in \mathcal{M}$ des Verteilungstupels $(m, v_\%) \in \mathcal{V}_\mathcal{M}$ die konkrete Anzahl der Agenten für die Mission m bestimmt. Daraufhin kann mit Hilfe der Hilfsvariablen n_1 und n_2 der Indexbereich für die ausgewählten Agenten bestimmt und die Mission für diese gestartet werden.
Die zweite und dritte Erweiterungsstufen modifizieren ausschließlich die Verteilungsliste $\mathcal{V}_\mathcal{M}$ und die Agentenlisten $\hat{\mathfrak{R}}$, wodurch ein Zugriff auf das Kommunikationsmodul und den Modelldatencontainer notwendig ist. Das Kommunikationsmodul ermöglicht die Berechnung einer Kommunikationsgruppe, welche aus allen transitiv erreichbaren Agenten besteht, und ist somit eine Partition der Agentenmodelle $\hat{\mathcal{R}}_k$. Das bedeutet, dass, falls eine Gruppe von Agenten, z. B. durch Hindernisse, getrennt oder zerstört wird, innerhalb jeder Gruppe die Verteilung gilt. Ansonsten kann es passieren, dass ein verschollener explorierender Agenten nicht automatisch ersetzt wird.
Für die dritte Stufe werden die verfügbaren Missionen in drei Kategorien unterteilt:

1. Missionen für beliebig viele Agenten
2. Missionen mit einer spezifischen Agentenanzahl
3. Missionen mit einer unspezifischen Agentenanzahl

Tabelle 5.9.: Zuordnung der Missionen zu den Kategorien.

Kategorie		
1	2	3
Exploration 5.4	Zielansteuerung 5.8	MANET 5.5
Pfadfolge (Patrouille) 5.9	Einzeltransport	Liniennetzwerk
Energiemanagement	Schwarmtransport 5.11	Linienformation 5.10
Störsenderdetektion	Kettentransport 5.13	Dreiecksformation 5.10
Triangulierung 5.14	Rendezvous 5.12	
Globale Pfadplanung 5.15	Eskortierung 5.6	
	Gruppeneskortierung 5.7	

Beispiele für Missionen der ersten Kategorie, die nahezu eine unbegrenzte Anzahl an Agenten verwenden können, sind Explorations- oder Patrouillenmissionen (siehe Abschnitte 5.4 und 5.9), denn bei diesen Missionen erhöht in der Regel die Anzahl der Agenten die Lösungsqualität oder reduziert die benötigte Zeit. Demgegenüber haben die meisten Transportmissionen sowie der Einzeltransport, Schwarmtransport und der Kettentransport eine sinnvolle Anzahl an Agenten, die sich durch die Paketanzahl oder den zu überbrückenden Abstand ergibt. Ebenso ist die Zielansteuerung stets für einen Agenten bestimmt. Bei der Eskortierungsmission ist die Anzahl der Agenten abhängig von der Anzahl an Fahrzeugen, sowie der gewünschten Eskortdichte der Agenten pro Fahrzeug. Die letzte Kategorie hat oftmals nur eine vom Anwender vorgegebene Spanne an Agenten. In diese Kategorie fallen die Formationsmissionen und die Netzwerkmissionen, wobei letztere auch der ersten Kategorie zugeordnet werden könnten, da oftmals nur eine untere Schranke existiert. Ein Übersicht über die Einteilung ist in Tabelle 5.9 gegeben.
Die Verteilung wird berechnet, indem zunächst alle Bedarfe der Kategorie Zwei abgefragt werden und anschließend die nominalen Bedarfe der Kategorie Drei, allerdings mindestens deren Minimum, bestimmt werden. Die Restkapazität fließt in die Missionen der Kategorie Eins. Stehen nicht mehr genügend Agenten für alle Missionen zur Verfügung, werden die Missionen in der Reihenfolge nur soweit aktiviert, solange die minimal benötigte Anzahl an Agenten noch vorhanden ist.

5.18. Zwischenfazit

Dieses Kapitel beschreibt einen mathematischen Potenzialfeldansatz zur Generierung verschiedenster Missionstypen und deren Behandlung für Roboterschwärme. Dabei wird bewusst eine Alternative zu bestehenden Verhaltensbaumansätzen oder Aufteilungen in globale und lokale Planungsschichten aufgezeigt, die in anderen Softwaresystemen wie dem „Robot Operating Systemen“ (kurz ROS) verwendet werden (vgl. [147] und [83]). Der Vorteil dieses Ansatzes besteht in der Modularität und der Möglichkeit zur Schachtelung bzw. der Faltung der Funktionen, um somit einen Missionsbaukasten zu erzeugen. Durch die Wahl des Optimierers (siehe Kapitel 2.4) können auch nichtlineare, unstetige und binäre Funktionen genutzt werden. Durch diese

Freiheitsgrade und die Funktionsvielfalt können auch andere Ansätze integriert oder übernommen werden, wie z. B. die globale Pfadplanungsmission. Das Ergebnis muss nur anschließend in ein Potenzialfeld umgewandelt werden. Dadurch lassen sich die Vorteile der verschiedenen Verfahren kombinieren. Als größter Nachteil bleibt hingegen die aufwändigere Parametrisierung der Funktionen und deren Gewichtung zueinander. Um ebenfalls komplexe Missionsabläufe zu bilden, werden die Konzepte der Implementierung zur automatischen Missionsverteilung (siehe Abschnitt 5.17) und zur zeit- oder prioritätsbasierten Behandlung (siehe Abschnitt 5.3) beschrieben. Die angeführten Experimente zu den jeweiligen Missionen und die weiterführenden Analysen in den Abschlussarbeiten zu jeder Mission untermauern die Funktionsfähigkeit dieses neuartigen Ansatzes.

6

Realer Roboterprototyp

Zur Analyse und Validierung der entwickelten Simulation sind reale Agenten oder physisch simulierte Agenten notwendig. Aus den ermittelten Anforderungen und den zur Verfügung stehenden Ressourcen wird ein Vorschlag für Prototypen der Agenten entwickelt.

6.1. Anforderungen

Als übergeordnetes Ziel sollen künftig Unterstützungsroboter für Einsatzkräfte in Katastrophengebieten hergestellt werden. Somit soll zur Überprüfung der Einsatzfähigkeit der Konzepte und ihrer Implementierung ein geländegängiges Kettenfahrzeug entwickelt werden, das mit einem Prozessor ausgestattet ist, der ein vollständiges Betriebssystem unterstützt, auf dem das „Robot Operating System“ (kurz ROS) [147] genutzt werden kann. ROS wird als führendes System für die Roboterforschung aufgrund der umfangreichen Unterstützung für viele Sensoren ausgewählt. Weiterhin muss sich das Fahrzeug eigenständig lokal und global lokalisieren und Hindernisse erkennen. Es sollen eigene Koppeldaten für eine besser Lokalisierung erzeugt werden. Ferner sollen die Aktionen des Roboters projiziert werden, um dessen Verhalten und Ziele Menschen verständlich zu machen. Der Preis pro Einheit soll aufgrund der verfügbaren Mittel 1500 € nicht überschreiten.

6.2. Aufbau

Zunächst werden der Aufbau des Roboters und die verwendeten Komponenten beschrieben. Dabei wird auf die unterschiedlichen Verbindungen und Schnittstellen eingegangen und die Einsatzdauer berechnet.

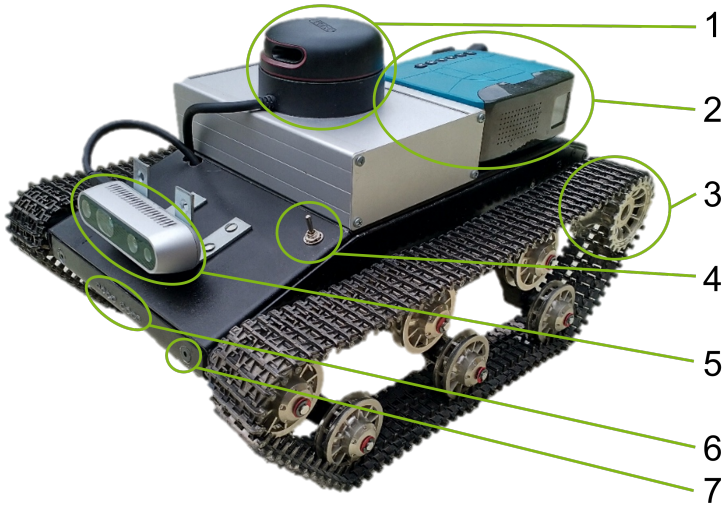


Abbildung 6.1.: Vollständig zusammengesetzter Prototyp bestehen aus einem LIDAR (1), einem mobilen Projektor (2), einem mit Gleichspannung betriebenen Kettenchassis (3) mit Einzelfederung, einem Schalter für den Hauptrechner (4), einer Stereotiefenkamera (5), einer LED-Beleuchtungseinheit (6) und zwei Lasermarkierern (7).

Motorspezifikation	
Nominalspannung	12 V
Strom ohne Last	1,3 A
Nominalstrom	4 A
Sperrstrom	11 A

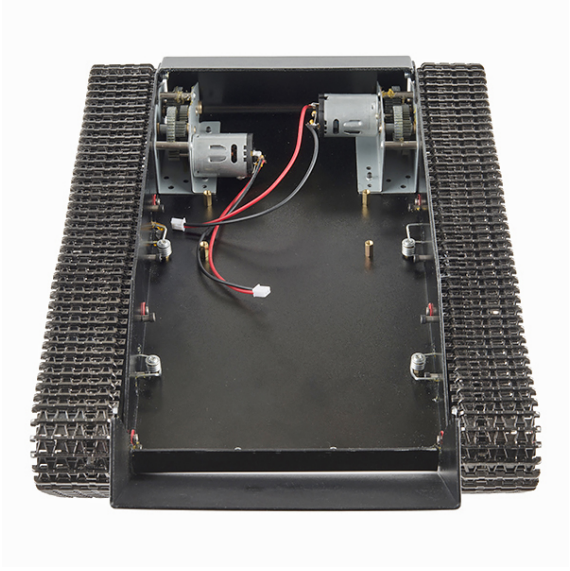
Tabelle 6.1.: Motorspezifikationen der Antriebe [174].

6.2.1. Chassis

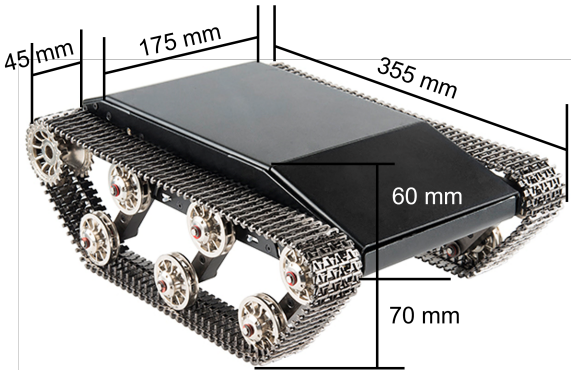
Als Chassis wird das „T'REX TANK" Chassis der Firma SparkFun Electronics (vgl. [174]) verwendet, da es das einzige auf dem Markt verfügbare Vollmetallchassis mit adäquaten Abmaßen und einer guten Verarbeitungsqualität ist. Die hohe Bodenfreiheit von 7 cm und die einzeln gefederten Laufrollen der Kettenantriebe bieten eine hohe Geländegängigkeit im Vergleich zu Radfahrzeugen. Die Auswahl anderer Chassis wird durch die Kosten stark limitiert. Angetrieben wird das Kettenfahrzeug durch zwei Gleichstrommotoren mit einem kontinuierlichen Strom von 4 A bei einer Spannung von 12 V, wodurch eine Leistung von 48 W pro Motor benötigt wird.

6.2.2. Leistungselektronik

Die Versorgungs- oder Leistungselektronik wird gespeist von zwei Batterien zu je 5 A h bei 14,8 V. Dies ergibt eine Energiemenge von 532,8 kJ. Die Spannung liegt zum Teil direkt am Zwei-Kanal-Motorregler zur Versorgung des Reglers und der Motoren, sowie an einem 12 V Gleichspannungswandler an, um die Hauptrecheneinheit mit einer konstanten Spannung zu versorgen. Der Motorregler kann bis zu 18 A dauerhafte Ströme und 50 A Spitzenlastströme schalten, sodass auch der Kurzschlussstrom des Motors mit 11 A kein Problem darstellt. Von der regulierten 12 V Versorgung aus wird die 5 V Spannungsversorgung für die Mikrocontroller und die Sensoren gespeist. Zuletzt bieten die über jeweils einen Metall-Oxid-Halbleiter-Feldeffekttransistor (Englisch: „Metal-Oxide-Semiconductor Field-Effect Transistor", kurz MOS-FET) gesteuerten externen Ausgänge den Zugriff auf die 12 V und 5 V Versorgung, um beispielsweise den Projektor zu versorgen. Die Gesamtleistung des Systems kann wie folgt bestimmt

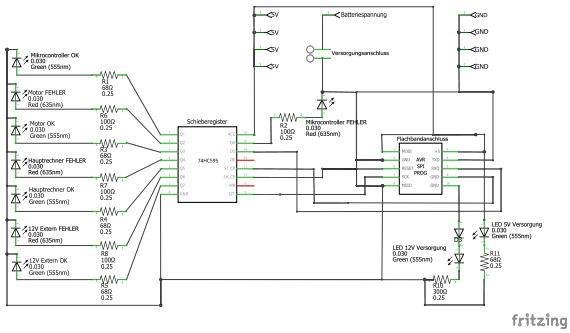


(a) Draufsicht auf das geöffnete Chassis mit Blick auf die Antriebsmotoren und die Getriebe [174].

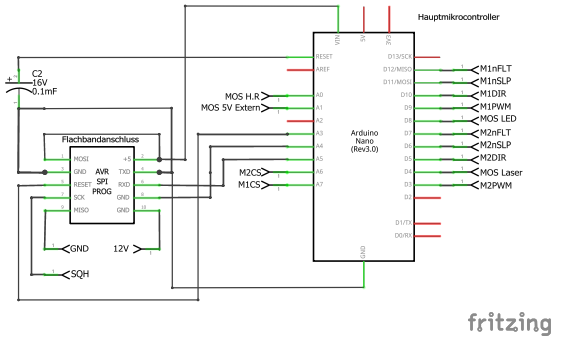


(b) Bemaßung des 3,7 kg schweren Chassis im geschlossenen Zustand [174].

Abbildung 6.2.: Ansicht des Chassis und seiner Maße.

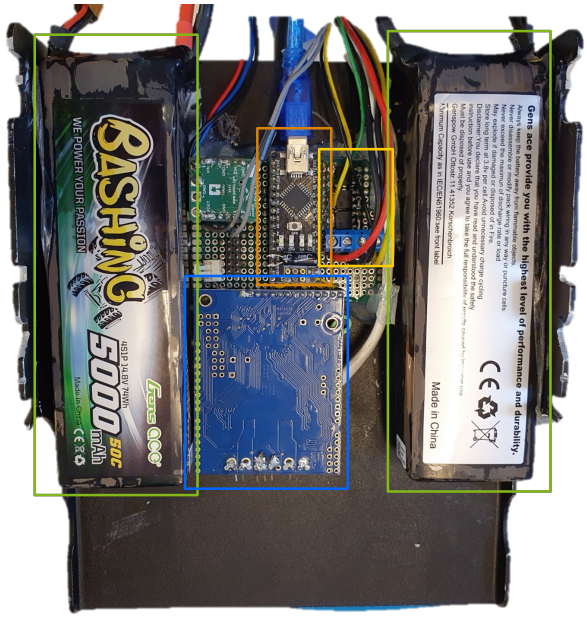


(a) Schaltplan der Stromversorgungs- und Statusanzeigeplatine mittels Schieberegister.

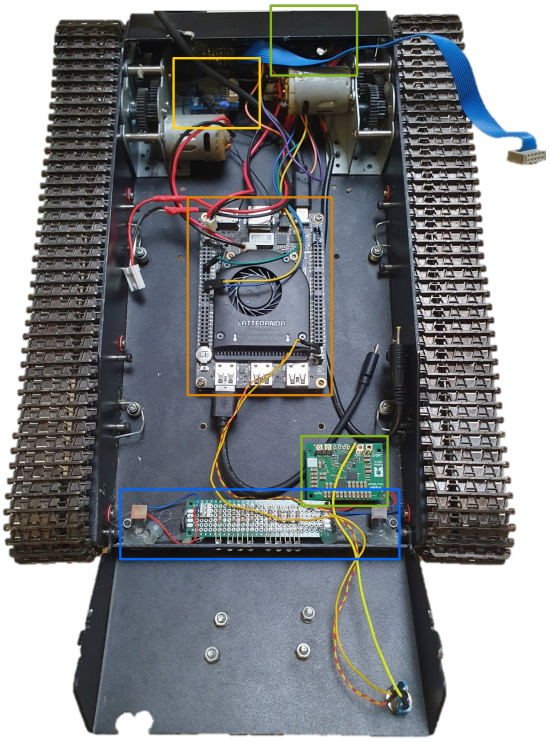


(b) Schaltplan des Hauptmikrocontrollers mit Flachbandbuchse für die Verbindung mit und Versorgung durch die Statusanzeigeplatine. Die mit „M1“ oder „M2“ bezeichneten Ausgänge steuern den Motortreiber und die mit „MOS“ gekennzeichneten Ausgänge steuern einen Metall-Oxid-Halbleiter-Feldeffekttransistor zur Schaltung der Versorgungspannungen.

Abbildung 6.3.: Visualisierung der wichtigsten Schaltpläne.



(a) Abbildung der unter dem Chassisdeckel montierten Komponenten. Grün eingerahmt sind die zwei Akkumulatoren, in blau der Motortreiber, dessen Kühlkörper mit dem Aluminiumgehäuse wärmeleitend verbunden ist, gelb eingerahmt sind die MOS-FETs und in orange eingerahmt ist der Hauptmikrocontroller mit seinem Zusatzkondensator, um ein Zurücksetzen zu verhindern.



(b) Abbildung der internen Komponenten. In grün eingerahmt sind die 5 V und 12 V Spannungswandler im Heck und in der Front, in blau ist die Beleuchtungseinheit bestehend aus LEDs und Lasern, in gelb eingerahmt ist das Satellitenpositionsmodul und in orange eingerahmt ist der Hauptrechner.

Abbildung 6.4.: Darstellung der Komponenten innerhalb des Chassis.

werden:

$$\underbrace{48\,\mathrm{W}\cdot 2}_{\text{Motoren}}+\underbrace{7{,}5\,\mathrm{W}}_{\text{LIDAR}}+\underbrace{24\,\mathrm{W}}_{\text{Projektor}}+\underbrace{3{,}5\,\mathrm{W}}_{\text{Kamera}}+\underbrace{45\,\mathrm{W}}_{\text{Hauptrechner}}+\underbrace{4\,\mathrm{W}}_{\text{Rest}}=180\,\mathrm{W} \tag{6.2.1}$$

$$\frac{532{,}8\,\mathrm{kJ}}{180\,\mathrm{W}}=2960\,\mathrm{s}\approx 49\,\mathrm{min} \tag{6.2.2}$$

$$\frac{532{,}8\,\mathrm{kJ}}{180\,\mathrm{W}-24\,\mathrm{W}-15\,\mathrm{W}}=3778{,}7\,\mathrm{s}\approx 63\,\mathrm{min} \tag{6.2.3}$$

Daraus ergibt sich eine Betriebszeit von 49 min bei Volllast bzw. ca. 1 h bei einem Betrieb ohne die konstante Nutzung des Projektors und einem nicht vollständig ausgelasteten Rechner.

6.2.3. Kommunikationselektronik

Die Hauptsteuereinheit bildet ein Lattepanda Alpha 864s. Er verfügt über einen Zweikern x86 Prozessor mit vier parallelen Prozessfäden und einen Arduino® Leonardo

Abbildung 6.5.: Darstellung der Statusleuchten (grün), der Antennenanschlüsse für Satellitenortung (blau) und IEEE 802.11 Wi-Fi (gelb) sowie des Einschalters (orange) am Heck des Roboters.

Koprozessor (vgl. [94]). Dieser Koprozessor bietet einige Kommunikationsschnittstellen sowie Vielzweckein- und Vielzweckausgaberegister (Englisch: „General Purpose Input Output“, kurz GPIO). An diesen Mikrocontroller ist ein L76X Satellitenpositionsbestimmungsmodul (vgl. [191]) angebunden, das mittels einer seriellen Schnittstelle via des „National Marine Electronics Association“ (kurz NMEA) 0183 Standards kommuniziert. Die interne Keramikantenne ist durch eine externe Antenne am Gehäuse ersetzt worden. An die drei zur Verfügung stehenden USB3.0-A Ports sind ein Laserabstandssensor (Englisch: „Light Detection And Ranging“, kurz LIDAR, vgl. [170]), eine Stereo-3D-Tiefenkamera Intel® RealSense™ D345i (vgl. [69]) und eine Arduino Nano (vgl. [6]) angeschlossen. Letzterer bildet die echtzeitfähige Basis des Roboters. Er steuert den Motorcontroller [137] und schaltet die Spannungsversorgung [136] für die Hauptrecheneinheit sowie alle weiteren Peripheriegeräte. Ferner zeigt er den Status des Roboters über einen rot-grünen LED-Farbcode an. Die LEDs werden über ein Schieberegister angesprochen. Die Beleuchtung und die Lasermarkierung werden über den Mikrocontroller verwaltet. Da dieser auch die Versorgung für die Hauptrecheneinheit frei gibt, entsteht ein zyklisches Problem. Denn mit dem Starten des Hauptrechners wird ein serieller Port zum Mikrocontroller geöffnet. Durch das Öffnen des Ports wird der Mikrocontroller in der Standardkonfiguration zurückgesetzt, sodass er programmiert werden kann. Durch das Zurücksetzten startet dieser neu und schaltet vorher als Fehlerschutz alle Ausgänge ab, die auch die Versorgung der Hauptrecheneinheit steuern. Folglich wird der Zurücksetzimpuls, der ein Register auf Masse herunter zieht mit einem Kondensator kompensiert, ohne jedoch die Funktionalität vollständig zu unterdrücken (siehe Abbildung 6.3b). Somit kann der Controller bei einem längeren gewollten Impuls immer noch neu gestartet werden.

6.2.4. Beleuchtung und Projektion

Neben den roten und grünen Statuslichtern, die anzeigen, welche Komponenten aktiv und funktionstüchtig sind, gibt es eine Frontbeleuchtung, zwei Laser und einen Projektor. Die Frontbeleuchtung mit acht sehr hellen, weißen LEDs verbessert die Kameraaufnahmen und die Tiefenerkennung in der Dunkelheit. Die Laser ermöglichen die Markierung und Ausrichtungsanzeige über weite Entfernungen. Abschließend dient der Projektor, der ebenfalls über den Mikrocontroller aktiviert werden kann, der

Projektion der Oberfläche des Hauptrechners mittels einer „High Definition Multimedia Interface“-Schnittstelle (kurz HDMI). Aufgrund der geringen Auflösung sind vor allem Bilder oder große Schriftgrößen zu verwenden.

6.2.5. Kühlung

Die Wärmeentwicklung aus der Umwandlung der elektrischen Energie, die für die Berechnung, die Spannungswandlung, den Antrieb und die Beleuchtung benötigt wird, muss abgeleitet werden. Hierbei wird der Vorteil genutzt, dass das gesamte Gehäuse aus gut wärmeleitfähigem Aluminium besteht. Daher werden alle Bauteile, die eine starke Wärmeentwicklung aufweisen, direkt oder indirekt mittels Aluminiumkühllamellen mit dem Chassis mit Wärmeleitflächen verbunden (siehe Abbildung 6.4). Die Hauptrecheneinheit besitzt darüber hinaus noch einen aktiven Lüfter, wodurch die Luftmasse im Inneren zusätzlich zirkuliert. Mit dieser Bauweise ist sogar eine spritzwassergeschützte oder wasserdichte Variante des Roboters möglich.

6.2.6. Hinderniserkennung und Lokalisation

Die eingebauten Gyrometer, Beschleunigungsmesser und Magnetometer (Englisch: „Inertia Measurement Unit“, kurz IMU) innerhalb der Kamera können die Positionierung der Hindernisse unterstützen und bieten gleichzeitig die Datengrundlage der Kopplung bzw. Odometrie (Englisch: „Odometry“) Diese werden mit den Daten des L76X Satellitenpositionsmoduls in einem erweiterten Kalman-Filter (kurz EKF) fusioniert, um die Präzision der Messdaten zu erhöhen. Im Allgemein sind die Zustände zum Teil messbar und werden zur Verbesserung und Rauschminimierung mittels Sensordatenfusion kombiniert. Dieses Verfahren entspricht in der Regelungstechnik einem erweiterten Beobachter auf Modellbasis. Jedoch sind die Modelle stark nicht linear und schwierig linearisierbar, wodurch die Möglichkeit der Verwendung eines Lungenberger Beobachters entfällt. Zur Hinderniserkennung werden der Laserscanner RPlidar A2 der Firma SLAMTEC (vgl. [170]) und die RealSense™ Kamera D345i der Firma Intel® (vgl. [69]) verwendet. Die vom LIDAR erzeugte Tupelliste bestehend aus Entfernung und Richtung wird an den in Abschnitt 4.2.15 beschriebenen Algorithmus zur Objektdetektion übergeben.

6.2.7. Kommunikation

Der Roboter besitzt mehrere Datenkommunikationswege. Zum einen bietet die Hauptrecheneinheit die Kommunikation via Bluetooth sowie über IEEE 802.11 im 2,4 GHz und 5 GHz Band über die gleichen Antennenausgänge an. Diese werden mit Externen Antennen am Gehäuse verbunden. Zum Anderen wird das in Abschnitt 3.5 beschriebene LoRa-Modul an einem externen USB-C Port angebunden. Somit sind sowohl Datenübertragungen im Nahbereich mit hoher Rate als auch eine geringere Übertragungsrate für weite Strecken möglich.

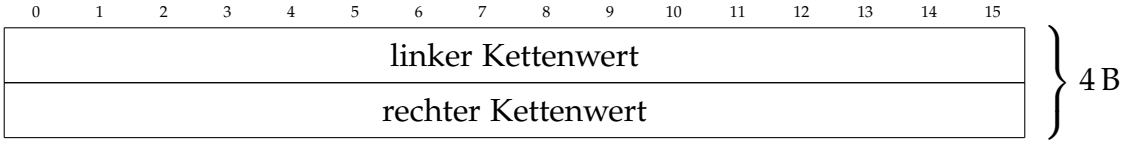


Abbildung 6.6.: Nachrichtenpaket für die Kettengeschwindigkeiten.

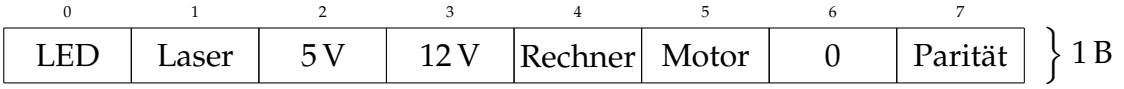


Abbildung 6.7.: Nachrichtenpaket für die Stromversorgung.

6.2.8. Software

Die Plattform ermöglicht die Verwendung der Betriebssysteme Microsoft® Windows™ 10 und Ubuntu 18 bis 22. Aufgrund der besseren Unterstützung des ROS-Mittelbaus wird Ubuntu verwendet. Um genügend Speicherplatz für die Software zu haben, wird der 64 GB große interne Speicher mit einer 256 GB großen Festplatte (Englisch: „Solid State Drive“, kurz SSD) erweitert. Aufgrund der Verfügbarkeit der Treiber und Softwarepakete für die Sensoren und zur Lokalisation wird die Version ROS der Version ROS 2 vorgezogen. Die Kommunikation mit dem Hauptmikrocontroller erfolgt mit dem „ROSSerial“ Grundgerüst, das über eine serielle Schnittstelle mit der Hauptrecheneinheit ROS-Nachrichten austauscht. Um die Auslastung der Schnittstelle und die Verarbeitungszeit auf dem Mikrocontroller gering zuhalten, wird die zu übertragene Information mit minimaler Bitanzahl kodiert. Zur Steuerung der Motorströme wird eine 16 bit-Zahl pro Kanal gefordert, die intern mittels Pulsweitenmodulation umgesetzt wird. Jedoch fehlt jegliche Dokumentation zur Umrechnung, sodass diese experimentell angenähert wird (siehe Experiment 6.4.2). Folglich kann die Vortriebssteuerung mit einer einzigen 32 bit-Zahl kodiert werden, die im Mikrocontroller durch eine Verschiebungsoperation (Englisch: „Shift Operation“) in die zwei Komponenten zerlegt wird. Das Vorzeichen entscheidet über die Laufrichtung der Ketten. Weiterhin gibt es sechs Komponenten, die ein- und ausgeschaltet werden können und somit als Bitvektor in einem Byte kodiert werden, weil dies die kleinste zu übertragende Einheit ist. Mittels einer Bitmaskierung wird direkt der entsprechende GPIO-Ausgang, der einen Transistor schaltet, gesetzt.
Neben den zuvor beschriebenen Nachrichten, die mittels des ROS-Nachrichtendienstes verschickt werden, gibt es noch weitere Softwaresystemknoten (Englisch: „ROS Node“) für den Betrieb des Roboters:

1. Abschalter
2. Erhalter
3. WiFi
4. Odometrie
5. Steuerdatenkonverter

6. EKF-Lokalisierung

7. GPS-Seriell

Dabei bilden die beiden Komponenten „Abschalter“ und „Erhalter“ eine wichtige Schutzkomponente. Der „Abschalter“ wird alle Versorgungssysteme abschalten, falls er nicht zuvor eine Nachricht vom „Erhalter“ registriert hat. Dies wird zyklisch geprüft. Der „Erhalter“ ist die erste Komponente, die im Fehlerfall beendet wird. So stoppt der Roboter automatisch bei Signalabbrüchen, wenn der „Erhalter“ auf einem anderen Rechner läuft.
Der „WiFi“-Knoten innerhalb des IEEE 802.11 Netzwerks misst den RSSI und den SNR, um wie in Kapitel 3 beschrieben als Qualitätsmaß des Netzwerks zu dienen. Die Kopplungsdaten können aus drei Quellen bezogen werden, in dem „EKF-Lokalisierung“-Knoten bereinigt und mit den globalen Satellitendaten („GPS-Seriell“) fusioniert und rauschminimiert werden. Die Daten kommen von der IMU oder können durch Integration der Steuerdaten im „Odometrie“-Knoten bestimmt werden. Die dritte Quelle ergibt sich aus Drehimpulsgebern an den Antriebsrädern (siehe Abschnitt 6.2.9.1).
Der Steuerdatenkonverter berechnet aus der gegebenen Vortriebs- und der Rotationsgeschwindigkeit die notwendigen ganzzahligen Werte für den Motorcontroller.

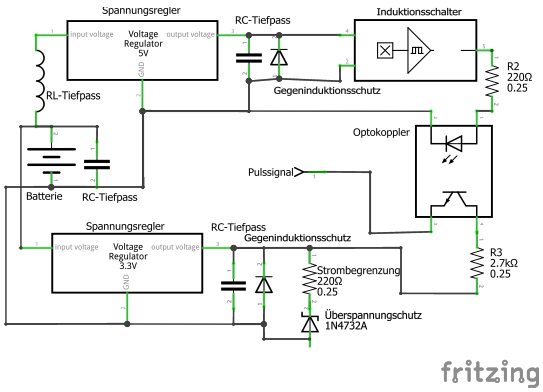
6.2.9. Erweiterung

Einsatzspezifische Erweiterungen wie z. B. Sensoren können in das Aluminiunkästchen unterhalb des LIDARs integriert werden (siehe Abbildung 6.1). Dieses bietet einen geschützten Bauraum von 16,5 cm · 4 cm · 10 cm und kann mit 12 V und 5 V sowie einer USB-C Schnittstelle versorgt werden. Die Versorgungsspannung lässt sich ebenfalls durch den Hauptmikrocontroller schalten.

6.2.9.1. Drehimpulsgeber

Der große Nachteil einer IMU-basierten Kopplung ist der große Rauschanteil und die interne Drift eines solchen Sensors. Dies führt zu einer großen Besteckversetzung und somit zu Positionsungenauigkeiten. Die Kopplung auf Grundlage der Steuerdaten ist weniger rauschbehaftet, jedoch sind die aus den Motorströmen resultierenden Geschwindigkeiten lastabhängig. Somit fehlt die Rückkopplung oder die Einbeziehung der Geländeeinflüsse. Daher wird versucht, die reale Drehzahl der Kettenantriebsräder zu messen. Die Gleichstrommotoren und der Platz im Gehäuse bieten keine Möglichkeit Drehimpulsgeber anzuschließen und zudem verfälscht das Spiel des Getriebes die tatsächliche Bewegung. Aufgrund der Einsatzbedingungen und der Verschmutzung im Gelände kann weder ein optisches noch ein mechanisches Verfahren angewandt werden. Kapazitive Sensoren entfallen wegen der möglichen Feuchtigkeit. Folglich wird die Rotation mit Hilfe induktiver Näherungsschalter gemessen, die mittels starker Neodymmagneten am Antriebsrad geschaltet werden (siehe Abbildung 6.8a). Die zugehörige Schaltung ist aufwändig, da die induzierte Spannung sowie die Gegeninduktion auf die Signalspannung und die Versorgungsspannung durchschlägt und

(a) Abbildung der regelmäßig an das Antriebsrad angeklebten Neodymmagnete, die grün eingerahmt sind, zur Erzeugung eines induktiven Impulses im blau eingerahmten Näherungsschalter.



(b) Schaltplan zur Filterung der durch die Induktion erzeugten Spannungsspitzen mit Hilfe von RL- und RC-Tiefpassfiltern sowie Optokopplern, Dioden und Z-Dioden.

Abbildung 6.8.: Darstellung des mechanischen und elektrischen Aufbaus für die Drehimpulsgebererweiterung.

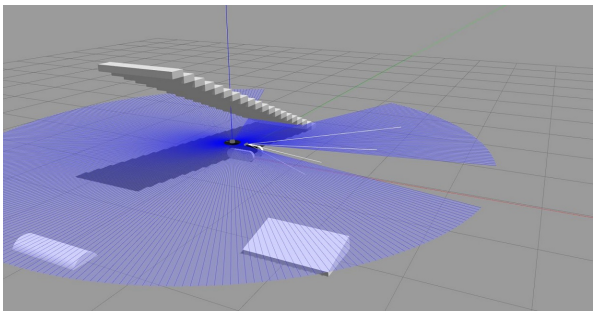
für den Mikrocontroller unzulässige Spannungen erzeugt. Daher wird mit einer Kombination aus Tiefpassschaltungen - RL- und RC-Tiefpass -, Dioden und Z-Dioden zur Spannungsbegrenzung sowie durch Optokoppler der Mikrocontrollereingang und dessen Versorgung abgeschirmt (siehe Abbildung 6.8b).

6.3. Hardwaresimulation

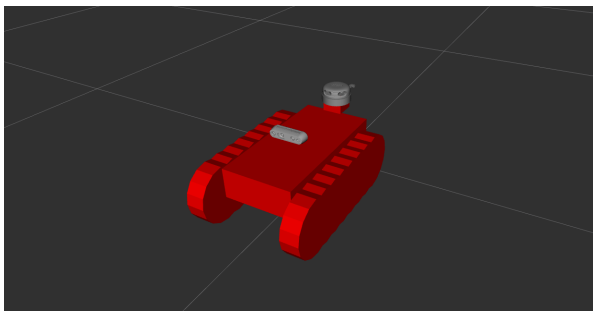
Um die für ROS entwickelten Softwarekomponenten zu testen und zu verbessern sowie um den Agentenschwarm mit mehr emulierten Agenten zu betreiben, werden die Kinematik, das Fahrverhalten und der Bodenkontakt in einer Simulation für einzelne Roboter simuliert. Dieser Multilevelsimulationsansatz wird mit der für ROS optimierten Simulationssoftware Gazebo (vgl. [82]) durchgeführt. Alternative Simulationsumgebungen für die Interaktion mit ROS sind Webots (vgl. [116]) und ARGos (vgl. [134]). Die Maße und Gewichte werden übernommen, jedoch kann das Kettenfahrwerk in Gazebo nur angenähert werden (vgl. [173] und [172]). Die Sensoren werden vollständig repliziert, sodass dieselben Nachrichtenpakete erzeugt werden. Weiterhin verarbeiten die virtuellen Aktoren dieselben Steuernachrichten, die an den Mikrocontroller geschickt werden.

6.3.1. Gazebo

Gazebo (vgl. [82]) ist eine Simulationssoftware für physikalische Bewegungen und Kontaktkräfte sowie für komplexe Sensoren. Weiterhin unterstützt die optionale dreidimensionale Visualisierung das Verständnis und die Analyse der Prozesse. In Abbildung 6.9a werden ebenfalls Sensorfunktionsweisen visualisiert. Somit lassen sich die



(a) Darstellung des modellierten Agenten in einer Testumgebung in Gazebo. Dabei visualisieren die blauen Strahlen die Strahlen des LIDAR. Die weißen Strecken verdeutlichen das Sehfeld der Kamera.



(b) Nahansicht des Modells in der Visualisierungssoftware rviz von ROS.

Abbildung 6.9.: Visualisierung des simulierten Robotermodells.

Strahlen des LIDARs erkennen und mit den in der übergeordneten Simulation entstanden Hindernissen vergleichen. Weiterhin können auch Messungenauigkeiten und Störeinflüsse miteingebracht werden, um die Software möglichst realitätsnah zu testen. Die Interaktion mit dem Agenten und die Aktivierung der verschiedenen Missionen erfolgt ausschließlich über die übergeordnete Simulation. Allerdings wird ein Versetzen des Roboters in der Gazebo Simulation auch bidirektional übernommen. Nur die Gazebo Simulation unterstützt nicht die Aktivierung der Missionen. Kamerabilder und Informationen der Tiefenkamera sind ebenfalls implementiert. Die Modellierung der Roboter in Gazebo erfolgt in einer Baumstruktur, die in Abbildung 6.10 zu sehen ist und in Abbildung 6.9b visualisiert wird. Der Vorteil dieser Verarbeitung ist, dass durch Matrizenmultiplikation und inverse Kinematik die Verschiebungs- und Rotationsbewegungen für alle Komponenten effizient berechenbar sind.

6.4. Integration und Kommunikation

Zur Kommunikation mit Gazebo mittels ROS wird der aus diversen Bibliotheken bestehende Simulationskern um je eine Bibliothek für ROS und ROS2 erweitert. Diese implementieren die Sensorschnittstellen und die Aktorschnittstellen, die in Abschnitten 4.2.15 und 4.2.10 beschrieben werden. Dabei wird das aktuelle Steuersignal in eine „Geometry Twist“-Nachricht des ROS-Systems umgewandelt und als Rundfunknachricht dem Nachrichtenverteiler übermittelt (Englisch: „Publish to topic“). Diese enthält die Geschwindigkeiten in alle lokalen Koordinatenrichtungen sowie die Rotationsgeschwindigkeiten um jede Achse. Ebenso abonnieren die Sensoren die Kanäle (Englisch: „Topics“), auf denen die Sensornachrichten von Gazebo veröffentlicht werden. Damit die Roboter sich jedoch in derselben Welt befinden, müssen die per YAML-Datei definierten Szenarien zuvor in dreidimensionale Welten für Gazebo konvertiert werden. Dazu wird jedes statisches Hindernis in der Datei in ein Objekt übersetzt. Dabei werden Linienzüge durch rechteckige Wände mit einer Breite von 1 m, runde Hindernisse durch Zylinder und rechteckige Hindernisse durch Quader ersetzt.

Abbildung 6.10.: Visualisierung der baumartigen Modellstruktur, die eine effiziente Berechnung der Transformationen ermöglicht.

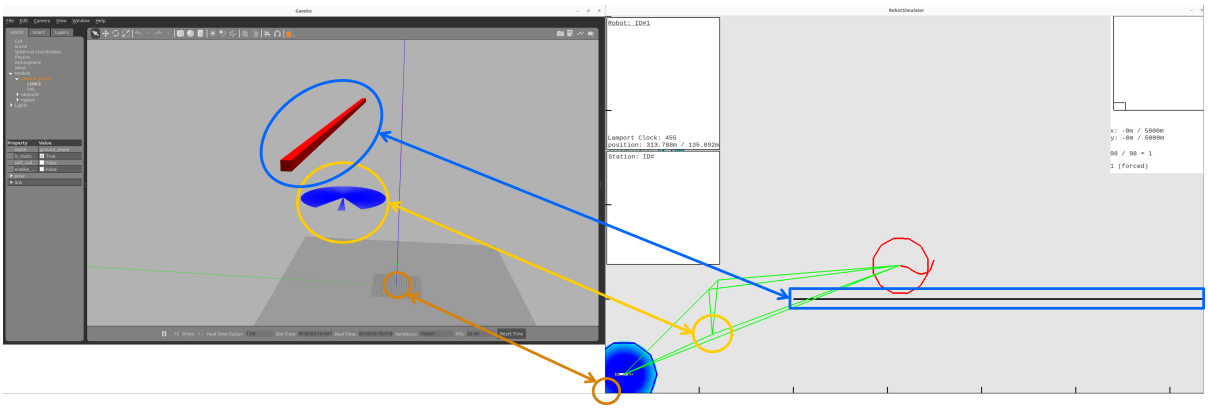


Abbildung 6.11.: Visualisierung der Datengleichheit beider Simulationen. In orange werden die Koordinatenursprünge, in blau das streckenförmige Hindernis und in gelb wird der Roboter sowie die Position seiner Kommunikationsbrücke in der Schwarmsimulation markiert. Die grünen Strecken verdeutlichen die Netzwerkverbindungen zwischen den Knoten.

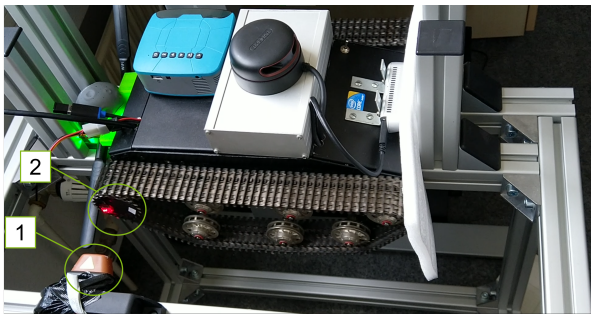
6.4.1. Experimente

Ziel des ersten Experiments ist es zu untersuchen, ob alle Sensordaten und Steuerdaten vollständig übermittelt werden.

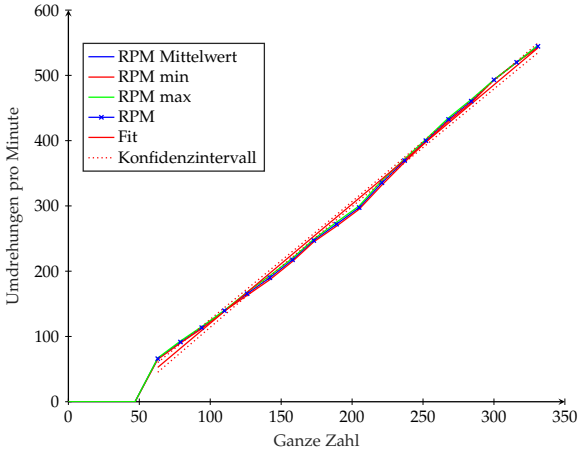
Experiment 6.4.1: *Erstelle einen Agenten in Gazebo an der Position* $\overset{\dagger,\dashv}{x} = 60\,\mathrm{m}$, $\overset{\dagger,\dashv}{y} = 40\,\mathrm{m}$, $\overset{\dagger,\dashv}{\gamma} = 0$ *und verknüpfe diesen mittels ROS2 mit einer Instanz eines simulierten Roboters. Dieser soll mit Hilfe einer Netzwerkbrücke auf Position* $\overset{\dagger,\dashv}{x} = 100\,\mathrm{m}$, $\overset{\dagger,\dashv}{y} = 100\,\mathrm{m}$, $\overset{\dagger,\dashv}{\gamma} = 0$ *in die zweidimensionale Schwarmsimulation eingebunden werden. Anschließend soll die Erkundungsmission (siehe Kapitel 5.4) gestartet werden. Der Datenabgleich wird auf Korrektheit untersucht.*

Bei der Durchführung ist zunächst zu erkennen, dass beim Starten der Schwarmsimulation die zum Agenten gehörende Nachrichtenbrücken und der Roboter in Gazebo auf verschiedenen Ausgangspositionen stehen. Wird nun ein virtueller Roboter mit ROS-Sensoren und Aktoren gestartet, der sich mit der Nachrichtenbrücke verbindet, so wird diese direkt mit Eintreffen der ersten Zustandsnachricht auf die Startposition $\overset{\dagger,\dashv}{x} = 60\,\mathrm{m}$, $\overset{\dagger,\dashv}{y} = 40\,\mathrm{m}$, $\overset{\dagger,\dashv}{\gamma} = 0$ versetzt und repliziert stets den Zustand. Da der Agent noch keine aktive Mission verfolgt, bleibt er aufgrund der zufallsbasierten Stellgrößen beinahe auf der Stelle. Mit Beginn der Erkundungsmission bewegen sich beide Repräsentationen synchron. Im dritten Experiment wird nun untersucht, ob, anstatt des mit Gazebo simulierten Roboters, der reale Roboter die gleichen Bewegungen ausführt. Dazu muss zunächst die Abbildung der „Geometry Twist“-Nachricht in ganzzahlige Werte für die Motorcontroller bestimmt und anschließend die Relation der Werte hin zu Kettenlaufgeschwindigkeiten ermittelt werden.

Experiment 6.4.2: *Iteriere über die 16 bit ganzzahligen Werten von 0 bis −32 768 unter der Annahme, dass der Motor ein identisches Verhalten für Vorwärts- und Rückwärtslaufen besitzt. Messe dabei die Kettenlaufgeschwindigkeiten.*



(a) Darstellung des Messstandes mit Lasermessgerät (1) und Reflexionsstreifen auf dem abgedunkelten Antriebsrad (2).



(b) Abbildung der ganzzahligen Werte auf Drehzahl pro Minute (Englisch: „Rounds Per Minute“, kurz RPM).

Abbildung 6.12.: Darstellung zur experimentellen Bestimmung des linearen Zusammenhangs zwischen dem durch eine ganze Zahl erzeugten Motorstroms und der Drehzahl des Antriebsrads.

Die Laufgeschwindigkeiten werden mit Hilfe eines Laserdrehzahlmessers und eines Reflektionsbandes am Antriebsrad aufgezeichnet. Auf Grundlage dieser Werteabbildung (siehe Abbildung 6.12b) kann der Mikrocontroller die Steuerdaten umsetzen. Dazu wird die Vortriebs- und die Rotationsgeschwindigkeit wie folgt in ganzzahlige Kettenwerte umgewandelt:

$$v_{\text{RPM}} = 1{,}8263\,\text{min}^{-1} \cdot v_I - -62{,}4481\,\text{min}^{-1} \tag{6.4.1}$$

$$v_{\text{Kette}} = \frac{\pi \cdot r_{\text{Antrieb}}}{60} \cdot v_{\text{RPM}} \tag{6.4.2}$$

$$v_{\text{KetteR}} = \overset{\dagger,\dashv}{v} + \pi \cdot \overset{\dagger,\dashv}{\gamma} \cdot b_{\text{Kettenabstand}} \tag{6.4.3}$$

$$v_{\text{KetteL}} = \overset{\dagger,\dashv}{v} - \pi \cdot \overset{\dagger,\dashv}{\gamma} \cdot b_{\text{Kettenabstand}} \tag{6.4.4}$$

$$v_I < 50 \implies v_I = 0 \tag{6.4.5}$$

Dabei gilt, dass der mittlere Kettenabstand $b_{\text{Kettenabstand}} = 0{,}22\,\text{m}$ und der Durchmesser des Antriebsrades $r_{\text{Antrieb}} = 0{,}055\,\text{m}$ beträgt. Auf Basis dieser Gleichungen wird für jede Kette der entsprechende ganzzahlige Wert bestimmt. Werte unter $v_I < 50$ werden auf Null gesetzt, da der Motorstrom zu gering wäre, um die Reibung der Bauteile zu überwinden, sodass die Motorspulen einem Kurzschlussstrom ausgesetzt wären. Aus der maximalen Geschwindigkeit der Agenten von $3\,\text{m}\,\text{s}^{-1}$ und dem zugestandenen Schutzbereich von 1 m um die Agenten herum, wird die Systemzeitschranke von 300 ms abgeleitet. Somit kann ein Notstopp innerhalb des Schutzbereichs garantiert werden.
Da alle Komponenten erfolgreich getestet sind, folgt ein reines Demonstrationsexperiment für die Zusammenführung der einzelnen Experimente zu einem realen Funktionstest.

Experiment 6.4.3: *Verbinde den realen Roboter mittels ROS mit der Schwarmsimulation und überprüfe, ob die durch den Simulationskern bestimmten Stellgrößen korrekt an die Simulation weitergegeben werden und ob die Nenngeschwindigkeit auf dem Messstand erreicht wird.*

Das Experiment 6.4.3 wird erfolgreich durchgeführt, jedoch gilt dies nur für den Messstand. Für einen realen Einsatz muss ein Geschwindigkeitsregler pro Kette auf Basis der Drehimpulssensoren oder der IMU entwickelt werden, sodass die Nenngeschwindigkeiten auch lastunabhängig gehalten werden. Ferner erhöht dies die Kursstabilität, sodass der Roboter deutlich präziser geradeaus fährt, indem Fertigungsunterschiede ausgeregelt werden. Dazu kann ein einfacher Proportional-Integral-Differential-Regler entweder durch das Ziegler und Nichols-Verfahren oder durch die Bestimmung der Pol- und Nullstellen exakt bestimmt werden (vgl. [105]).

6.5. Zwischenfazit

Dieses Kapitel beschreibt die prototypische Realisierung des Simulationskerns und der Regelungstechnik für kleine Kettenfahrzeuge mit sehr beschränkten Mitteln. Damit wird aufgezeigt, dass die Simulation im Stande ist, auf verschiedenen Geräten mit unterschiedlicher Hardware, unterschiedlichen Betriebssystemen und verschiedenen Mittelbausoftwaresystemen, in diesem Fall ROS und ROS2, zu kommunizieren und zu interagieren. Weiterhin wird dargelegt, wie die Integration und Kosimulation mit anderen Softwaresystemen durch die Implementierung weniger Schnittstellen erfolgen kann. Somit ist die entwickelte Software in der Lage, bei der Erstellung neuer Missionen nicht nur als Test- und Planungswerkzeug zu dienen, sondern sie kann ebenfalls in allen Zwischenstufen bis zur Integration in reale Systeme verwendet werden. Dabei unterstützt sie sowohl X86, X64 als auch ARM Prozessoren. Die Möglichkeit zur Verwendung in Kombinationen mit weit verbreiteten Softwaresystemen wie aber auch die Unabhängigkeit von diesen hat den Vorteil, dass bei neueren Versionen oder neu erscheinenden Softwaresystemen für reale Roboter ein Fortbestehen der Software gegeben ist.

7

Zusammenfassung und Ausblick

Dieses Kapitel fasst jedes vorangegangene und die Zusammenhänge zwischen diesen zusammen. Die daraus resultierenden Erkenntnisse und Implikationen werden aufgezeigt sowie der weitere Forschungsbedarf herausgestellt.

7.1. Zusammenfassung

Die Einleitung in Kapitel 1 führt aus, dass die aktuellen Softwaresysteme und Missionsplanungssysteme Robotergruppen, die gemeinsam im offenen unstrukturierten Gelände agieren, nur unzureichend abdecken. Einerseits existieren Simulatoren und Verhaltensalgorithmen für solitäre Systeme oder kleine Fahrzeuggruppen, die in Katastrophengebieten operieren. Andererseits existieren für spezielle Aufgabenfelder, wie die der Intralogistik, zentralisierte Schwarmalgorithmen, die jedoch auf eine Umgebung, eine zuverlässige Infrastruktur und eine dedizierte Aufgabe abgestimmt sind. Allerdings fehlt die kombinierte Betrachtung von großen Roboterschwärmen, die dezentral ohne zuverlässige Infrastruktur und mit einer hohen eigenen Systemausfallwahrscheinlichkeit gemeinsam ein emergentes Missionsverhalten erbringen.
Aufbauend auf dieser Erkenntnis wird ein echtzeitfähiger modellprädiktiver Regler im Kapitel 2 entwickelt . Dieser ist in der Lage, Trajektorienlösungen für verschiedene kinematische Systeme zu berechnen, die nahe an das Optimum herankommen. Dabei wird trotz einer großen Funktionsfreiheit der Kostenfunktion eine Zeitschranke von 300 ms gewahrt, innerhalb dieser stets ein mögliche, aber nicht zwingend optimale, Trajektorie generiert wird. Hingegen wird garantiert, dass diese Trajektorie niemals die schlechteste wählbare Trajektorie ist. Durch eine geschickte exponentielle Zerlegung des Prädiktionshorizontes lässt sich sowohl ein hochaufgelöstes Fahrverhalten für den Nahbereich als auch eine weite Sichtweite für die Prädiktion erzielen. Jedoch bleibt die Lösung stets eine lokale Lösung. Dieses Problem wird in Kapitel 5 mit Hilfe globaler Missionen gelöst.
Um die Kommunikationsmöglichkeit zwischen den Agenten des Roboterschwarms zu wahren, ist die Überwachung, Schätzung und Prädiktion der Netzwerkqualität notwendig. Folglich wird in Kapitel 3 eine mathematisch-physikalische Methode zur Schätzung und somit zur distanzbasierten Prädiktion der Übertragungsqualität für IEEE 802.11 Netzwerke im 2,4 GHz und 5 GHz Band vorgestellt. Da sich diese Netzwer-

ke jedoch nicht für Katastrophengebiete oder weiträumige infrastrukturlose Gelände eigenen, wird ein dezentrales Knotenpunktnetzwerk auf Basis der LoRa-Technologie entwickelt. Dieses muss jedoch als beschränkt einsatzfähig eingestuft werden, weil die erzielten Übertragungsraten trotz starker Datenkompression sehr gering sind und somit das Netzwerk stets im Sättigungsbereich betrieben wird.
Anschließend folgt die Vorstellung des vollständig eigenständig entwickelten Softwaresystems für mobile autonome Roboterschwärme in Kapitel 4. Dieses als Autonomiekern bezeichnete Softwaresystem zeichnet sich durch eine hohe Modularität, Wiederverwendbarkeit und Laufzeiteffizienz aus. Es umfasst alle Komponenten, die für einen vollständigen autonomen Betrieb notwendig sind und bietet darüber hinaus die Möglichkeit von Simulationen großer Schwärme mit bis zu 70 Agenten, die ihren eignen vollständigen Kern verwenden. Zudem ermöglicht das gewählte Konzept verschiedene Visualisierer für unterschiedliche Hardware und Präsentationsformen zu nutzen. Auf eingebetteten Systemen kann eine Konsolenanwendung, Standardoberfläche oder eine zweidimensionale Darstellung verwendet werden. Auf leistungsstärkeren Rechnern kann die immersive Darstellung mittels der virtuellen Realität genutzt werden bei gleichzeitiger Interaktion mit Realsystemen. Diese können durch die einfache Interaktion mit anderen Softwarebibliotheken auch extern eingebunden werden.
Aufbauend auf dem Softwaresystem werden in Kapitel 5 Missionen anhand verschiedener Anwendungsfälle abgeleitet und in mathematische Funktionen umgewandelt. Neben der mathematischen Formulierung der Kostenfunktionen, die ein gewünschtes Missionsverhalten erzeugen, werden auch algorithmische Details präsentiert, die die Dezentralität der Ausführung ermöglichen und effiziente speichertechnische Kniffe aufzeigen. Weiterhin ist es aufgrund der gestuften Berechnungsebenen und Ausführungszeiten möglich sowohl globale als auch lokale Trajektorienplanung zu kombinieren, um ein global konvergentes Verhalten des lokalen Optimierers zu erzeugen. Ein Beispiel dafür ist die globale Pfadplanung mittels des A*-Algorithmus (siehe Abschnitt 5.15). Ebenfalls wird die Modularität des Softwaresystems bei den Missionen weitergedacht, sodass die stets komplexer werdenden Missionen in der Regel aus einer Komposition einfacher Missionen zusammengesetzt werden. Durch die Ausarbeitung der verschiedenen Missionen und die experimentellen Analysen der Funktionsfähigkeit wird eine Alternative zum verbreiteten Ansatz der Verhaltensbäume oder der Trennung von globaler und lokaler Pfadplanung geboten.
Abschließend folgt zur Validierung der Simulation, der Regelungstechnik und des gesamten plattformübergreifenden Softwaresystems dessen Integration in einen realen selbst entwickelten Prototypen unter Achtung starker Mittelbeschränkungen. Weiterhin wird durch die Nutzung einer weiteren Robotersimulation zur Hardwareemulation die Kosimulationsfähigkeit veranschaulicht. So ist es möglich das entwickelte Softwaresystem in verschiedene Simulationsabstraktionsebenen einzubinden, sodass reale Roboter, Roboter mit emulierter Hardware und rein simulierte Agenten in einer übergeordneten Missionssimulation miteinander interagieren, ohne einen Unterschied feststellen zu können. Folglich ist es möglich, das Softwaresystem auf verschiedenen realen Plattformen mit unterschiedlichen Betriebssystemen, Prozessorarchitekturen und Datenvermittlungsschichten wie z. B. ROS und ROS2 einzusetzen und durch die

einfache Integration die bereits gewohnten Simulatoren weiterhin zu verwenden.

7.2. Implikationen der Arbeit

Eine der wichtigsten Implikationen dieser Arbeit ist die Erkenntnis, dass angepasste, stützstellenbasierte, modellprädiktive Regler ein nahezu optimales Verhalten generieren können, obwohl die Stabilität und die Optimalität kaum oder gar nicht beweisbar sind. Die Funktionsfreiheit ermöglicht die Konstruktion von Impulsfunktionen mit einer optimalen reellwertigen Lösung, deren Wahrscheinlichkeit, sie zu finden, Null ist. Jedoch wird anhand von ausgiebigen empirischen Versuchen auch für verschiedene Systemmodelle, Kostenfunktionen und Regelungszeile eine hohe Regelgüte für typische Anwendungsfälle aufgezeigt. Somit sollte dieser Regler vor allem innerhalb der mobilen Robotik weiter untersucht und seine Funktionsfähigkeit an mehreren realen System verdeutlicht werden.
Als weitere Implikation wird ein einfaches mathematisch-physikalisches Modell zur Schätzung der Übertragungsqualität und -raten geboten, das für emergentes Verhalten von Netzwerkteilnehmern oder Sensornetzwerken weiterverwendet werden kann. Jedoch sollte dabei eine konkrete Anpassung an die frequenzeigenen Parameter durchgeführt und die Resultate dafür validiert werden. Weiterhin wird gezeigt, dass auch ein kostengünstiges Netzwerk mit geringer Übertragungsleistung in Kombination mit starker anwendungsspezifischer Datenkompression und mittels genügend Autonomie und Dezentralität seitens der Netzwerkteilnehmer brauchbar angewendet werden kann. Diese Datenratenreduktion ist auch für leistungsfähigere Netzwerke sinnvoll, falls viele Schwarmagenten realisiert werden oder anderweitige Daten, wie z. B. Videoübertragungen, durch das aufgebaute Knotenpunktnetzwerk geleitet werden sollen, sodass die Regelungsdaten die Netzwerke nur minimal belasten.
Die Konstruktion des Softwaresystems zeigt auf, wie auch heutzutage flexibel erweiterbare, hocheffizient parallelisierbare und stark interoperable Software mit C++ geschrieben werden kann, die sich aufgrund einer sehr geringen Anzahl an Bibliotheksabhängigkeiten auf verschiedenen Betriebssystemen und Prozessorarchitekturen ohne Funktionalitätsreduktion ausführen lässt. Dabei wird auch das Entwurfsmuster der Fabrik durch eine weitere Abstraktionsebene speichereffizienter weiterentwickelt, indem nur die Generatoren, die die eigentlichen Objekte erzeugen, gespeichert werden. Diese besitzen oft nur einen Konstruktor und sind mit Standardparametern befüllbar. Dadurch werden keine Instanzen der eigentlich speicherintensiven Objekte gespeichert. Dies unterscheidet die Implementierung deutlich von der ROS Implementierung. Weiterhin ermöglicht die strikte Trennung von Daten und Funktionen eine einfache Anbindung an verschiedene Visualisierungen in zwei oder auch drei Dimensionen, weil nur die Interpretation der Daten implementiert werden muss. Anschließend reicht zur Laufzeit ein Verweis auf die Datencontainer der entsprechenden Agenten.
Die Verwendung stützstellenbasierter Optimierer für die Kostenfunktionen impliziert, dass das Konzept der Potenzialfunktionen aus den 1990iger Jahren (vgl. [181] und [149]) deutlicher weiter gedacht werden kann. Die ursprünglichen Beschränkungen auf stetige differenzierbare Funktionen, deren Gradienten bekannt sind, entfallen und

die Freiheit der Kostenfunktionen bisheriger nichtlinearer modellprädiktiver Regler werden überboten. Jedoch ist damit keine Stabilität nach Ljapunov mehr garantiert. Somit ermöglicht die Freiheit die Formulierung einer Vielzahl komplexer Missionen auf Kosten der beweisbaren Stabilität. Die Güte muss empirisch gezeigt werden.
Aus der Realisierung unter starker Ressourcenbeschränkung folgt hauptsächlich die Validität für den gewählten Ansatz und die Funktionsfähigkeit des implementierten Softwaresystems. Ebenfalls kann so gezeigt werden, dass, analog zur kostengünstigen Netzwerktechnik, die Beschränkungen zu einer anderen Sichtweise führen und zu einer kreativen Problemlösung beitragen.

7.3. Ausblick

Aus dieser Arbeit ergeben sich einige neue und weiterführende Forschungsfragen sowie zu beweisende Aussagen. So sollte versucht werden, die Optimalität des Optimierers bzw. die Stabilität des gesamten Reglers formal für kleine Beispiele und einfache Kostenfunktionen zu zeigen. Dazu wird im Anhang C.5 ein Ansatz sowie die Definition grundlegender Mengen und Eigenschaften geboten. Ansonsten können weitere empirische Vergleiche mit anderen Algorithmen, die z. B. in der Softwarebibliothek „Open Motion Planning Library“ (kurz OMPL, vgl. [178]) enthalten sind, durchgeführt werden. Jedoch beachten die wenigsten dieser Algorithmen die gegebene Systemdynamik. Eine Abschlussarbeit entwickelt die Verknüpfung der Bibliotheken (vgl. [75]).
Weiterhin ist die Netzwerkschätzung für mehrere Frequenzen und Modulationsarten zu erweitern und mittels umfangreicher Experimentalreihen zu belegen. Dadurch kann ein einheitliches Modell auch für LoRa genutzt werden und ein dynamischer Wechsel der Kommunikationsmodule in dem Softwaresystem unterstützt werden. Durch die geringe Größe der stark komprimierten Regelungsdaten können diese als priorisierte Pakete im Router bzw. innerhalb des Netzwerkprotokolls behandelt werden, ohne das Netzwerk zu verstopfen. Dadurch lässt sich die Sicherheit der Roboter oder autonomen Fahrzeuge deutlich erhöhen. Zudem eignen sich kleine Datenpakete besonders für den Transport via verschlüsselter und gespreizter Frequenzbänder in Katastrophen- oder Krisengebieten, weil diese deutlich schwerer zu erfassen, zu manipulieren und zu stören sind. Zudem ist die Übertragungsrate dieser Netzwerke in der Regel gering.
Als nächster Schritt sollte die Visualisierung der Simulation und der Agenten sowie die Interaktion mit diesen deutlich verbessert werden und an die entsprechende Visualisierungsart angepasst werden. Eine vollständige intuitive Bedienung mittels Tastatur und Maus sollte für die zweidimensionale Echtzeitdarstellung verfolgt werden; dahingegen sollte die dreidimensionale Darstellung der Agenten in der virtuellen Realität an die Gestensteuerung angepasst werden. Zudem ist die Kommunikation mit weiteren Simulatoren und die Ko-Simulationsfähigkeit zu erweitern, um eine größere Verbreitung und Testung der Software zu erzielen. Ebenfalls ist die Integration weiterer Nachrichtenverteilungssysteme bzw. Robotermittelbausoftware voranzutreiben, um weitere existierende Roboter und Hardware einzubinden. Als Beispiel ist hier die

MAVlink-Schnittstelle als Standard für Flugdrohnen zu nennen (vgl. [1]).
Der Bereich der mathematischen Missionen kann nahezu endlos mit weiteren Missionen ergänzt werden. Hierbei ist vor allem eine automatische Parametrisierung zu entwickeln, die abhängig von der Missionsanzahl und den jeweiligen Prioritäten die Werte der einzelnen Parameter aufeinander abstimmt, denn die adäquate Parametrisierung ist der größte Nachteil des Potenzialfeldverfahrens.
Falls der reale Prototyp weiterentwickelt werden soll, um die Prototypenphase zu verlassen, so ist zunächst eine adäquate Geschwindigkeitsregelung unter Beachtung der kombinierten Kopplungsdaten der IMU, der Drehimpulsgeber und der globalen Positionssensoren zu erstellen. Dazu sollten diese in einem gemeinsamen Kalman-Filter verarbeitet werden. Das bereinigte Signal dient dann als Messgröße des Reglers. Daraufhin ist eine Energieverwaltung mit Batteriemanagementsystem auf Grundlage der Erkenntnisse aus der Abschlussarbeit Klindworth [80] zu entwickeln. Schließlich sollten die Platinen professionell gefertigt und miniaturisiert werden, sodass anschließend die Kühltechnik verbessert und das Gehäuse spritzwassergeschützt ausgelegt werden kann.

Abbildungsverzeichnis

Tabellenverzeichnis

Algorithmenverzeichnis

Quelltextverzeichnis

Literatur

[1] **A. Koubâa, A. Allouch, M. Alajlan, Y. Javed, A. Belghith und M. Khalgui:** „Micro Air Vehicle Link (MAVlink) in a Nutshell: A Survey“. In: *IEEE Access* 7 (2019), S. 87658–87680.

[2] **A. Puzicha und P. Buchholz:** „Dynamic Mission Control for Decentralized Mobile Robot Swarms“. In: *2022 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. 2022, S. 257–263.

[3] **J. Adamy:** *Nichtlineare Systeme und Regelungen*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2018.

[4] **G. Allaire, F. Jouve und A.-M. Toader:** „A level-set method for shape optimization“. In: *Comptes Rendus Mathematique* 334.12 (2002), S. 1125–1130.

[5] **A. Ammar, H. Bennaceur, I. Châari, A. Koubâa und M. Alajlan:** „Relaxed Dijkstra and A* with linear complexity for robot path planning problems in large-scale grid environments“. In: *Soft Computing* 20.10 (2016), S. 4149–4171.

[6] **Arduino S.r.l.:** *Nano*. Hrsg. von Arduino S.r.l. 2019. URL: `https://docs.arduino.cc/hardware/nano` (besucht am 27. 06. 2023).

[7] **J. Arroyo, C. Manna, F. Spiessens und L. Helsen:** „Reinforced model predictive control (RL-MPC) for building energy management“. In: *Applied Energy* 309 (2022), S. 118346.

[8] **K. Arulkumaran, M. P. Deisenroth, M. Brundage und A. A. Bharath:** „A Brief Survey of Deep Reinforcement Learning“. In: *IEEE Signal Processing Magazine* 34.6 (2017), S. 26–38.

[9] **AVM Computersysteme Vertriebs GmbH:** *Nutzdatenrate der WLAN-Verbindung ermitteln*. Hrsg. von AVM Computersysteme Vertriebs GmbH. -langsam-geringe-Datenrate/, 2019. URL: `https://avm.de/service/fritzbox/fritzbox-7490/wissensdatenbank/publication/show/514_WLAN-Verbindung-langsam-geringe-Datenrate/` (besucht am 20. 05. 2019).

[10] **F. Barczik:** „Entwurf einer tretraedischen Driftkompensation für präszise Positionierung und Lagerkennung mittels IMU und GPS“. Bachelorarbeit. Dortmund: Technische Universität Dortmund, 2023-05-17.

[11] **B. Beachkofski und R. Grandhi:** „Improved Distributed Hypercube Sampling“. In: *43rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*. Reston, Virigina: American Institute of Aeronautics and Astronautics, 2002.

[12] **R. Bellman:** *Dynamic programming*. Princeton, NJ: Princeton Univ. Pr, 1984.

[13] **M. de Berg, M. van Kreveld, M. Overmars und O. C. Schwarzkopf:** *Computational Geometry*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000.

[14] **R. Berto, P. Napoletano und M. Savi:** *A LoRa-Based Mesh Network for Peer-to-Peer Long-Range Communication*. Bd. 21. Sensors. 2021.

[15] **F. Biscani und D. Izzo:** „A parallel global multiobjective framework for optimization: pagmo". In: *Journal of Open Source Software* 5.53 (2020), S. 2338.

[16] **H. Bittel und L. Strom:** *Rauschen: Eine Einführung zum Verständnis elektrischer Schwankungserscheinungen*. Berlin, Heidelberg, New York: Springer, 1971.

[17] **C. Ebi, F. Schaltegger, A. Rüst und F. Blumensaat:** „Synchronous LoRa Mesh Network to Monitor Processes in Underground Infrastructure". In: *IEEE Access* 7 (2019), S. 57663–57677.

[18] **G. C. Calafiore, F. Dabbene und R. Tempo:** „Research on probabilistic methods for control system design". In: *Automatica* 47.7 (2011), S. 1279–1293.

[19] **S. Carpin, M. Lewis, J. Wang, S. Balakirsky und C. Scrapper:** „USARSim: a robot simulator for research and education". In: *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE, 2007, S. 1400–1405.

[20] **E. Castello, T. Yamamoto, F. D. Libera, W. Liu, A. F. T. Winfield, Y. Nakamura und H. Ishiguro:** „Adaptive foraging for simulated and real robotic swarms: the dynamical response threshold approach". In: *Swarm Intelligence* 10.1 (2016), S. 1–31.

[21] **R. P. Centelles, F. Freitag, R. Meseguer, L. Navarro, S. F. Ochoa und R. M. Santos:** *A LoRa-Based Communication System for Coordinated Response in an Earthquake Aftermath*. Bd. 31. Proceedings. 2019.

[22] **B. Chazelle:** „Triangulating a simple polygon in linear time". In: *Discrete & Computational Geometry* 6.3 (1991), S. 485–524.

[23] **T. Chung:** *DARPA Subterranean (SubT) Challenge*. Hrsg. von DEFENSE ADVANCED RESEARCH PROJECTS AGENCY. 2017. URL: `https://www.darpa.mil/program/darpa-subterranean-challenge`.

[24] **D. Solpico, M. I. Tan, E. J. Manalansan, F. A. Zagala, J. A. Leceta, D. F. Lanuza, J. Bernal, R. D. Ramos, R. J. Villareal, X. M. Cruz, J. A. dela Cruz, D. J. Lagazo, J. L. Honrado, G. Abrajano, N. J. Libatique und G. Tangonan:** „Application of the V-HUB Standard using LoRa Beacons, Mobile Cloud, UAVs, and DTN for Disaster-Resilient Communications". In: *2019 IEEE Global Humanitarian Technology Conference (GHTC)*. IEEE, 2019, S. 1–8.

[25] **P. N. Dave und J. B. Patil:** „Modeling and control of nonlinear unmanned ground all terrain vehicle". In: *2015 International Conference on Trends in Automation, Communications and Computing Technology (I-TACT-15)* (2015), S. 1–7.

[26] **Q. B. Diep und I. Zelinka:** „The Movement of Swarm Robots in an Unknown Complex Environment“. In: *AETA 2018 - Recent Advances in Electrical Engineering and Related Sciences: Theory and Application*. Hrsg. von I. Zelinka, P. Brandstetter, T. Trong Dao, V. Hoang Duy und S. B. Kim. Cham: Springer International Publishing, 2020, S. 949–959.

[27] **G. Dulac-Arnold, N. Levine, D. J. Mankowitz, J. Li, C. Paduraru, S. Gowal und T. Hester:** „Challenges of real-world reinforcement learning: definitions, benchmarks and analysis“. In: *Machine Learning* 110.9 (2021), S. 2419–2468.

[28] **C. Dünnermann:** „Kartendatenextraktion für eine modellbasierte Echtzeitsimulation“. Bachelorarbeit. Dortmund: Technische Universität Dortmund, 2021-02-15.

[29] **R. Dutter:** *Geostatistik: Eine Einführung mit Anwendungen*. Mathematische Methoden in der Technik. Wiesbaden: Vieweg+Teubner Verlag, 1985.

[30] **E. Sakic und W. Kellerer:** „Response Time and Availability Study of RAFT Consensus in Distributed SDN Control Plane“. In: *IEEE Transactions on Network and Service Management* 15.1 (2018), S. 304–318.

[31] **E. Todorov:** „General duality between optimal control and estimation“. In: *2008 47th IEEE Conference on Decision and Control*. IEEE, 2008, S. 4286–4292.

[32] **I. Eigenseher:** „GPU basierte Parallelisierung stützstellenbasierter nichtlinearer modellprädiktiver Regler für autonome Roboterschwärme unter Wissensbeschränkungen“. Bachelorarbeit. Dortmund: Technische Universität Dortmund, 2023-01-18.

[33] **T. Elanko:** „Transportkettenentwicklung für eine verteilte nichtlineare modellprädiktive Regelung autonomer Roboterschwärme unter Wissensbeschränkungen“. Bachelorarbeit. Dortmund: Technische Universität Dortmund, 2021-05-19.

[34] **J. Euler:** „Optimal Cooperative Control of UAVs for Dynamic Data-Driven Monitoring Tasks“. Dissertation. Darmstadt: Technische Universität Darmstadt, 2018.

[35] **Y. Eun und H. Bang:** „Cooperative Control of Multiple Unmanned Aerial Vehicles Using the Potential Field Theory“. In: *Journal of Aircraft* 43.6 (2006), S. 1805–1814.

[36] **F. Ferrari, M. Zimmerling, L. Thiele und O. Saukh:** „Efficient network flooding and time synchronization with Glossy“. In: *Proceedings of the 10th ACM/IEEE International Conference on Information Processing in Sensor Networks* (2011), S. 73–84.

[37] **T. Falkenhahn:** „Simulative Analyse von Angriffen auf Routingverfahren in MANETs“. Bachelorarbeit. Dortmund: Technische Universität Dortmund, 2021-03-29.

[38] **J. Fan, Z. Wang, Y. Xie und Z. Yang:** „A Theoretical Analysis of Deep Q-Learning“. In: *Proceedings of the 2nd Conference on Learning for Dynamics and Control*. Hrsg. von A. M. Bayen, A. Jadbabaie, G. Pappas, P. A. Parrilo, B. Recht, C. Tomlin und M. Zeilinger. Bd. 120. Proceedings of Machine Learning Research. PMLR, 2020, S. 486–489.

[39] **Y. Fan, G. Feng, Y. Wang und C. Song:** „Distributed event-triggered control of multi-agent systems with combinational measurements“. In: *Automatica* 49.2 (2013), S. 671–675.

[40] **G. Ferré und A. Giremus:** „LoRa Physical Layer Principle and Performance Analysis“. In: *ICECS 2018 25th IEEE International Conference on Electronics Circuits and Systems*. IEEE, 2018.

[41] **D. Fox, W. Burgard und S. Thrun:** „The dynamic window approach to collision avoidance“. In: *IEEE Robotics & Automation Magazine* 4.1 (1997), S. 23–33.

[42] **U. Freyer:** *Nachrichten-Übertragungstechnik: Grundlagen, Komponenten, Verfahren und Anwendungen der Informations-, Kommunikations- und Medientechnik*. 7. Aufl. Lernbücher der Technik. München: Hanser, 2017.

[43] **H. Fukushima, K. Kon und F. Matsuno:** „Distributed Model Predictive Control for Multi-Vehicle Formation with Collision Avoidance Constraints“. In: *Proceedings of the 44th IEEE Conference on Decision and Control*. IEEE, 2005, S. 5480–5485.

[44] **G. Kaur und P. Thakur:** „Routing Protocols in MANET: An Overview“. In: *2019 2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)*. 2019, S. 935–941.

[45] **F. Garcia und E. Rachelson:** „Markov Decision Processes“. In: *Markov decision processes in artificial intelligence*. ISTE. London: Wiley, 2013, S. 1–38.

[46] **A. Gerwins:** „Globale Pfadplanung im komplexen Umfeld für eine verteilte nichtlineare modellprädikative Regelung autonomer Roboterschwärme unter Wissensbeschränkungen“. Bachelorarbeit. Dortmund: Technische Universität Dortmund, 2023-04-19.

[47] **N. Grabenschröer:** „Rendezvous-basierte Verhaltensstrategien für eine verteilte nichtlineare modellprädiktive Regelung autonomer Roboterschwärme unter Wissensbeschränkungen“. Bachelorarbeit. Dortmund: Technische Universität Dortmund, 2021-04-21.

[48] **O. W. Grajoszek:** „Reinforcement Learning-gestützte Bewegungsplanung von Aktoren“. Bachelorarbeit. Dortmund: Technische Universität Dortmund, 2022-08-15.

[49] **K. Greff, R. K. Srivastava, J. Koutnik, B. R. Steunebrink und J. Schmidhuber:** „LSTM: A Search Space Odyssey“. In: *IEEE transactions on neural networks and learning systems* 28.10 (2017), S. 2222–2232.

[50] **L. Grüne und J. Pannek:** *Nonlinear Model Predictive Control: Theory and Algorithms*. 2nd ed. 2017. SpringerLink Bücher. Cham: Springer, 2017.

[51] **G. Guennebaud, B. Jacob u. a.:** *Eigen v3*. 2010.

[52] **G. Guo, L. Ding und Q.-L. Han:** „A distributed event-triggered transmission strategy for sampled-data consensus of multi-agent systems". In: *Automatica* 50.5 (2014), S. 1489–1496.

[53] **H. Huh und J. Y. Kim:** „LoRa-based Mesh Network for IoT Applications". In: *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*. IEEE, 2019, S. 524–527.

[54] **H. Lee und K. Ke:** „Monitoring of Large-Area IoT Sensors Using a LoRa Wireless Mesh Network System: Design and Evaluation". In: *IEEE Transactions on Instrumentation and Measurement* 67.9 (2018), S. 2177–2187.

[55] **H. T. Friis:** „A Note on a Simple Transmission Formula". In: *Proceedings of the IRE* 34.5 (1946), S. 254–256.

[56] **J. H. Halton:** „Algorithm 247: Radical-inverse quasi-random point sequence". In: *Commun. ACM* 7.12 (1964), S. 701–702.

[57] **P. Hämäläinen, J. Rajamäki und C. K. Liu:** „Online Control of Simulated Humanoids Using Particle Belief Propagation". In: *Proc. SIGGRAPH '15*. New York, NY, USA: ACM, 2015.

[58] **D. Heeger, M. Garigan, E. E. Tsiropoulou und J. Plusquellic:** „Secure Energy Constrained LoRa Mesh Network". In: *Ad-Hoc, Mobile, and Wireless Networks*. Hrsg. von L. A. Grieco, G. Boggia, G. Piro, Y. Jararweh und C. Campolo. Cham: Springer International Publishing, 2020, S. 228–240.

[59] **J. D. Hernandez, E. Vidal, G. Vallicrosa, E. Galceran und M. Carreras:** „Online path planning for autonomous underwater vehicles in unknown environments". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, S. 1152–1157.

[60] **K. Hester u. a.:** *Meshtastic*. 2020. URL: `https://github.com/meshtastic/Meshtastic-device` (besucht am 22. 02. 2022).

[61] **K. Hester u. a.:** *Meshtastic Android*. 2020. URL: `https://github.com/meshtastic/Meshtastic-Android` (besucht am 22. 02. 2022).

[62] **A.-C. Hildebrandt, M. Klischat, D. Wahrmann, R. Wittmann, F. Sygulla, P. Seiwald, D. Rixen und T. Buschmann:** „Real-Time Path Planning in Unknown Environments for Bipedal Robots". In: *IEEE Robotics and Automation Letters* 2.4 (2017), S. 1856–1863.

[63] **D. M. Himmelblau:** *Applied nonlinear programming*. New York und Düsseldorf: McGraw-Hill, 1972.

[64] **S. Hochreiter und J. Schmidhuber:** „Long short-term memory". In: *Neural computation* 9.8 (1997), S. 1735–1780.

[65] **D. Hsu, R. Kindel, J.-C. Latombe und S. M. Rock:** „Randomized Kinodynamic Motion Planning with Moving Obstacles“. In: *The International Journal of Robotics Research* 21 (2002), S. 233–255.

[66] **A. S. Huang, E. Olson und D. C. Moore:** „LCM: Lightweight Communications and Marshalling“. In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2010, S. 4057–4062.

[67] **D. Huang, T. T. Allen, W. I. Notz und R. A. Miller:** „Sequential kriging optimization using multiple-fidelity evaluations“. In: *Structural and Multidisciplinary Optimization* 32.5 (2006), S. 369–382.

[68] **A. Ihler und D. McAllester:** „Particle Belief Propagation“. In: *International Conference on Artificial Intelligence and Statistics* 5 (2009), S. 256–263.

[69] **Intel Corporation:** *Intel RealSense Depth Camera D435i*. Hrsg. von Intel Corporation. 2019. URL: https://www.intelrealsense.com/depth-camera-d435i/.

[70] **J. Amiryan und M. Jamzad:** „Adaptive motion planning with artificial potential fields using a prior path“. In: *2015 3rd RSI International Conference on Robotics and Mechatronics (ICROM)* (2015), S. 731–736.

[71] **J. D. Gammell, S. S. Srinivasa und T. D. Barfoot:** „Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic“. In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems* (2014), S. 2997–3004.

[72] **J. G. Panicker, M. Azman und R. Kashyap:** „A LoRa Wireless Mesh Network for Wide-Area Animal Tracking“. In: *2019 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)*. IEEE, 2019, S. 1–5.

[73] **J. Kennedy und R. Eberhart:** „Particle swarm optimization“. In: *Proceedings of ICNN'95 - International Conference on Neural Networks*. 1995, 1942–1948 vol.4.

[74] **J. Yang, H. Zhang, Y. Ling, C. Pan und W. Sun:** „Task Allocation for Wireless Sensor Network Using Modified Binary Particle Swarm Optimization“. In: *IEEE Sensors Journal* 14.3 (2014), S. 882–892.

[75] **Z. Ji:** „Entwicklung und Analyse von Vergleichstests für Open Motion Planning Library (OMPL) und Advanced Control Particle Belief Propagation (ACPBP)“. Bachelorarbeit. Dortmund: Technische Universität Dortmund, 2023-08-21.

[76] **K. Kim und J. Kim:** „Path Optimization for Cooperative Mapping Using Multiple Robots with Limited Sensing Capabilities“. In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, S. 1499–1506.

[77] **D. Kaluzny:** „Adaptive Multiskalenrepräsentation von geotopologischen Karten für modellbasierte Echtzeitsimulationen“. Bachelorarbeit. Dortmund: Technische Universität Dortmund, 2021-10-11.

[78] **E. Kement:** „Vergleichende Analyse Gradienten- und Stützstellen-basierter modellprädiktiver Regler für reale Roboterschwärme“. Bachelorarbeit. Dortmund: Technische Universität Dortmund, 2022-05-24.

[79] **B. King, A. Rennie und G. Bennett:** „An efficient triangle mesh slicing algorithm for all topologies in additive manufacturing“. In: *The International Journal of Advanced Manufacturing Technology* 112.3 (2021), S. 1023–1033.

[80] **Y. Klindworth:** „Entwicklung eines energiegewahren Verhaltens für eine verteilte nichtlineare modellprädiktive Regelung autonomer Roboterschwärme unter Wissensbeschränkungen“. Bachelorarbeit. Dortmund: Technische Universität Dortmund, 2022-04-07.

[81] **A. Klöckner:** „Behavior Trees for UAV Mission Management“. In: *INFORMATIK 2013: Informatik angepasst an Mensch, Organisation und Umwelt*. Hrsg. von Matthias Horbach. Bd. P-220. GI-Edition-Lecture Notes in Informatics (LNI) - Proceedings. Köllen Druck + Verlag GmbH, Bonn, 2013, S. 57–68.

[82] **N. Koenig und A. Howard:** „Design and Use Paradigms for Gazebo, An Open-Source Multi-Robot Simulator“. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2004, S. 2149–2154.

[83] **A. Koubaa:** *Robot Operating System (ROS)*. Bd. 895. Cham: Springer International Publishing, 2021.

[84] **A. Kuntz, F. Schmidt-Eisenlohr, O. Graute, H. Hartenstein und M. Zitterbart:** „Introducing probabilistic radio propagation models in OMNeT++ mobility framework and cross validation check with NS-2“. In: *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*. Hrsg. von Sándor Molnár, John R. Heath, Olivier Dalle und Gabriel A. Wainer. ICST/ACM, 2008, S. 72.

[85] **L. Barnes, M. Fields und K. Valavanis:** „Unmanned ground vehicle swarm formation control using potential fields“. In: *2007 Mediterranean Conference on Control & Automation*. IEEE, 2007, S. 1–8.

[86] **L. E. Kavraki, P. Svestka, J.-C. Latombe und M. H. Overmars:** „Probabilistic roadmaps for path planning in high-dimensional configuration spaces“. In: *IEEE Transactions on Robotics and Automation* 12.4 (1996), S. 566–580.

[87] **L. Palmieri, S. Koenig und K. O. Arras:** „RRT-based nonholonomic motion planning using any-angle path biasing“. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)* (2016), S. 2775–2781.

[88] **J. Lächele, A. Franchi, H. H. Bülthoff und P. R. Giordano:** „SwarmSimX: Real-Time Simulation Environment for Multi-robot Systems“. In: *Simulation, Modeling, and Programming for Autonomous Robots - Third International Conference*. Hrsg. von I. Noda, N. Ando, D. Brugali und J. J. Kuffner. Bd. 7628. Lecture Notes in Computer Science. Springer, 2012, S. 375–387.

[89] **B. Lahres, G. Raýman und S. Strich:** *Objektorientierte Programmierung: Das umfassende Handbuch*. 5., aktualisierte Auflage. Rheinwerk Computing. Bonn: Rheinwerk Verlag, 2021.

[90] **C. Lamini, S. Benhlima und A. Elbekri:** „Genetic Algorithm Based Approach for Autonomous Mobile Robot Path Planning“. In: *Procedia Computer Science* 127 (2018), S. 180–189.

[91] **L. Lamport:** „Fast Paxos“. In: *Distributed Computing* 19.2 (2006), S. 79–103.

[92] **L. Lamport:** „Time, Clocks, and the Ordering of Events in a Distributed System“. In: *Commun. ACM* 21.7 (1978), S. 558–565.

[93] **M. Lapan:** *Deep Reinforcement Learning: Das umfassende Praxis-Handbuch*. 1. Auflage. Frechen: mitp, 2020.

[94] **LattePanda:** *LattePanda Alpha - an intel Core i5 x86 Single Board Computer*. Hrsg. von LattePanda. 2019. URL: `https://www.lattepanda.com/lattepanda-alpha` (besucht am 27. 06. 2023).

[95] **T. Laue:** „Eine Verhaltenssteuerung für autonome mobile Roboter auf der Basis von Potentialfeldern“. Diplomarbeit. Bremen: Universität Bremen, 5. Januar 2004.

[96] **S. M. LaValle und J. J. Kuffner:** „Randomized Kinodynamic Planning“. In: *The International Journal of Robotics Research* 20.5 (2001), S. 378–400.

[97] **K. Leppersjohann:** „Automatisierte Missionsabbildung für eine verteilte nichtlineare modellprädiktive Regelung autonomer Roboterschwärme unter Wissensbeschränkungen“. Bachelorarbeit. Dortmund: Technische Universität Dortmund, 2020-08-20.

[98] **H. Li und Y. Shi:** „Event-triggered robust model predictive control of continuous-time nonlinear systems“. In: *Automatica* 50.5 (2014), S. 1507–1513.

[99] **J. Li, S. Liu, B. Zhang und X. Zhao:** „RRT-A* Motion planning algorithm for non-holonomic mobile robot“. In: *2014 Proceedings of the SICE Annual Conference (SICE)*. IEEE, 2014, S. 1833–1838.

[100] **M. Li, C. Jiang, X. Song und H. Cao:** „Parameter Effects of the Potential-Field-Driven Model Predictive Controller for Shared Control“. In: *Automotive Innovation* 6.1 (2023), S. 48–61.

[101] **S. R. Lindemann und S. M. LaValle:** „Incrementally reducing dispersion by increasing Voronoi bias in RRTs“. In: *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*. Bd. Vol. 4. IEEE, 2004, S. 3251–3257.

[102] **X. Liu und D. Gong:** „A comparative study of A-star algorithms for search and rescue in perfect maze“. In: *2011 International Conference on Electric Information and Control Engineering*. IEEE, 2011, S. 24–27.

[103] **LTE-Anbieter.info:** *Maximale Datenrate der Luftschnittstelle bei LTE: Wie errechnet sich diese eigentlich?* Hrsg. von Maik Wildemann & Sebastian Schöne 2.0Promotion GbR. LTE-Anbieter.info, 2019. URL: https://www.lte-anbieter.info/technik/datenrate-luftschnittstelle.php (besucht am 20. 05. 2019).

[104] **J. Lubars, H. Gupta, S. Chinchali, L. Li, A. Raja, R. Srikant und X. Wu:** „Combining Reinforcement Learning with Model Predictive Control for On-Ramp Merging". In: *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2021, S. 942–947.

[105] **J. Lunze:** *Regelungstechnik 1*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010.

[106] **J. Lunze:** *Regelungstechnik 1*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2020.

[107] **J. Lunze:** *Regelungstechnik 2*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016.

[108] **M. A. Ponti, F. P. dos Santos, L. S. F. Ribeiro und G. B. Cavallari:** „Training Deep Networks from Zero to Hero: avoiding pitfalls and going beyond". In: *2021 34th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*. Elsevier, 2021, S. 9–16.

[109] **F. Maaßen:** „A Comparison of Fast-Recovery Mechanisms in Networks". Bachelorarbeit. Dortmund: Technische Universität Dortmund, 2022-05-17.

[110] **S. Macenski, F. Martin, R. White und J. Ginés Clavero:** „The Marathon 2: A Navigation System". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, S. 2718–2725.

[111] **J. Maczuga:** „Entwicklung eines realen mobilen ad hoc Netzwerks auf „LoRa"-Basis für Steuerungsdaten eines dezentralen Roboterschwarms unter Wissensbeschränkungen". Bachelorarbeit. Dortmund: Technische Universität Dortmund, 2022-03-04.

[112] **F. Mager, D. Baumann, R. Jacob, L. Thiele, S. Trimpe und M. Zimmerling:** „Feedback Control Goes Wireless: Guaranteed Stability over Low-power Multihop Networks". In: *ICCPS '19: Proceedings of the 10th ACM/IEEE International Conference on Cyber-Physical Systems* (2019), S. 97–108.

[113] **A. Makarow, C. Rösmann, M. Keller und T. Bertram:** „Vergleich der modellprädiktiven Trajektorienscharregelung mit konventionellen Regelungsverfahren". In: *Vierte IFToMM D-A-CH Konferenz 2018 : EPFL Lausanne* (2018).

[114] **P. Marin-Plaza, A. Hussein, D. Martin, A. d. La Escalera und A. Noureldin:** „Global and Local Path Planning Study in a ROS-Based Research Platform for Autonomous Vehicles". In: *Journal of Advanced Transportation* (2018), S. 1–10.

[115] **Michail G. Lagoudakis und Ronald E. Parr:** „Reinforcement Learning as Classification: Leveraging Modern Classifiers". In: *ICML'03: Proceedings of the*

Twentieth International Conference on International Conference on Machine Learning, S. 424–431.

[116] **O. Michel:** „Webots: Professional Mobile Robot Simulation“. In: *Journal of Advanced Robotics Systems* 1.1 (2004), S. 39–42.

[117] **V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg und D. Hassabis:** „Human-level control through deep reinforcement learning“. In: *Nature* 518.7540 (2015), S. 529–533.

[118] **D. Morgan, S.-J. Chung und F. Y. Hadaegh:** „Decentralized Model Predictive Control of Swarms of Spacecraft Using Sequential Convex Programming“. In: *Journal of Guidance, Control, and Dynamics* 37.6 (2014), S. 1725–1740.

[119] **R. Nader:** „Konsens-basierte Logistikmissionen für eine verteilte nichtlineare modellprädiktive Regelung autonomer Roboterschwärme unter Wissensbeschränkungen“. Bachelorarbeit. Dortmund: Technische Universität Dortmund, 2021-03-31.

[120] **K. Naderi, J. Rajamäki und P. Hämäläinen:** „RT-RRT*: A Real-time Path Planning Algorithm Based on RRT*“. In: *Proceedings of the 8th ACM SIGGRAPH Conference on Motion in Games*. MIG '15. ACM, 2015, S. 113–118.

[121] **W. Naeem, R. Sutton, J. Chudley, F. R. Dalgleish und S. Tetlow:** „A genetic algorithm-based model predictive control autopilot design and its implementation in an autonomous underwater vehicle“. In: *Proceedings of the Institution of Mechanical Engineers, Part M: Journal of Engineering for the Maritime Environment* 218.3 (2004), S. 175–188.

[122] **S. Narayanan und P. Kumar:** „Dynamics of Nonlinear Oscillators with Discontinuous Nonlinearities Subjected to Harmonic and Stochastic Excitations“. In: *Journal of The Institution of Engineers (India): Series C* 102.6 (2021), S. 1321–1363.

[123] **J. Nieto, E. Slawinski, V. Mut und B. Wagner:** „Online path planning based on Rapidly-Exploring Random Trees“. In: *2010 IEEE International Conference on Industrial Technology*. IEEE, S. 1451–1456.

[124] **J. Nocedal, A. Wächter und R. A. Waltz:** „Adaptive Barrier Update Strategies for Nonlinear Interior Methods“. In: *SIAM Journal on Optimization* 19.4 (2009), S. 1674–1693.

[125] **J. Nocedal und S. J. Wright:** *Numerical Optimization*. Second Edition. Springer Series in Operations Research and Financial Engineering. New York, NY: Springer Science+Business Media LLC, 2006.

[126] **C. Nwankpa, W. Ijomah, A. Gachagan und S. Marshall:** *Activation Functions: Comparison of trends in Practice and Research for Deep Learning*. URL: http://arxiv.org/pdf/1811.03378v1.

[127] **H. Oulahrir:** „Konsens-basierte Logistikmissionen für eine verteilte nichtlineare modellprädiktive Regelung autonomer Roboterschwärme unter Wissensbeschränkungen“. Bachelorarbeit; Dortmund: Technische Universität Dortmund, 2022-09-12.

[128] **L. Palmieri und K. O. Arras:** „Distance metric learning for RRT-based motion planning with constant-time inference“. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, S. 637–643.

[129] **N. Palmieri, X.-S. Yang, F. D. Rango und S. Marano:** „Comparison of bio-inspired algorithms applied to the coordination of mobile robots considering the energy consumption“. In: *Neural Computing and Applications* 31.1 (2019), S. 263–286.

[130] **N. Palmieri, X.-S. Yang, F. D. Rango und A. F. Santamaria:** „Self-adaptive decision-making mechanisms to balance the execution of multiple tasks for a multi-robots team“. In: *Neurocomputing* 306 (2018), S. 17–36.

[131] **L. E. Parker:** „Distributed intelligence: overview of the field and its application in multi-robot systems“. In: *Journal of Physical Agents (JoPha)* 2.1 (2008), S. 5–14.

[132] **C. Peng und T. C. Yang:** „Event-triggered communication and control co-design for networked control systems“. In: *Automatica* 49.5 (2013), S. 1326–1332.

[133] **J. Peters:** „Policy gradient methods“. In: *Scholarpedia* 5.11 (2010), S. 3698.

[134] **C. Pinciroli, V. Trianni, R. O'Grady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. Di Caro, F. Ducatelle, M. Birattari, L. M. Gambardella und M. Dorigo:** „ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems“. In: *Swarm Intelligence* 6.4 (2012), S. 271–295.

[135] **L. Pitonakova, M. Giuliani, A. Pipe und A. Winfield:** „Feature and Performance Comparison of the V-REP, Gazebo and ARGoS Robot Simulators“. In: *Towards Autonomous Robotic Systems*. Hrsg. von M. Giuliani, T. Assaf und M. E. Giannaccini. Cham: Springer International Publishing, 2018, S. 357–368.

[136] **Pololu Corporation:** *Pololu 12V, 15A Step-Down Voltage Regulator D24V150F12*. Hrsg. von Pololu Corporation. 2019. URL: `https://www.pololu.com/product/2885` (besucht am 27. 06. 2023).

[137] **Pololu Corporation:** *Pololu Dual G2 High-Power Motor Driver 18v18 Shield for Arduino*. Hrsg. von Pololu Corporation. 2019. URL: `https://www.pololu.com/product/2515` (besucht am 27. 06. 2023).

[138] **M. J. D. Powell:** „A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation“. In: *Advances in Optimization and Numerical Analysis*. Hrsg. von S. Gomez und J.-P. Hennart. Dordrecht: Springer Netherlands, 1994, S. 51–67.

[139] **A. Puzicha:** „Control of decentralized systems under uncertainty“. In: *Dagstuhl 2020 - Gemeinsamer Workshop des Graduiertenkollegs 2340 und des HPI Forschungskollegs* 2020 (2020).

[140] **A. Puzicha:** „Control of decentralized systems under uncertainty“. In: *Proceedings of the 2021 Joint Workshop of the German Research Training Groups in Computer Science* 2021 (2021), S. 44.

[141] **A. Puzicha:** „Modeling and analysis of a distributed non-linear model-predictive control for swarms of autonomous robots with limited communication skills (in German)“. Masterthesis. Dortmund: TU Dortmund University, 2019.

[142] **A. Puzicha und P. Buchholz:** „A Simulation Environment for Autonomous Robot Swarms with Limited Communication Skills“. In: *Simulation Tools and Techniques*. Hrsg. von H. Song und D. Jiang. Bd. 370. Cham: Springer International Publishing, 2021, S. 206–226.

[143] **A. Puzicha und P. Buchholz:** „Decentralized model predictive control for autonomous robot swarms with restricted communication skills in unknown environments“. In: *Procedia Computer Science* 186 (2021), S. 555–562.

[144] **A. Puzicha und P. Buchholz:** „Mission-based autonomy core for decentralized mobile UGV swarms“. In: *ISR Europe 2022 : 54th International Symposium on Robotics in conjunction with: automatica June 20 – 21, 2022, Munich*. Hrsg. von VDE Verband der Elektrotechnik Elektronik Informationstechnik e.V. Berlin: VDE VERLAG GMBH, 2022, S. 319–326.

[145] **A. Puzicha und P. Buchholz:** „Real-Time Simulation of Robot Swarms with Restricted Communication Skills“. In: *2020 IEEE/ACM 24th International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*. 2020, S. 1–8.

[146] **C. Qin, D. Klabjan und D. Russo:** „Improving the Expected Improvement Algorithm“. In: *31st Conference on Neural Information Processing Systems (NIPS 2017* (2017), S. 1–11.

[147] **M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler und A. Ng:** „ROS: an open-source Robot Operating System“. In: *ICRA Workshop on Open Source Software* 3 (2009).

[148] **C. Ranisch, H. Koch und T. Streul:** „Datenbasierte, adaptive Regelung von Servomotoren für Industrieroboter“. In: *e & i Elektrotechnik und Informationstechnik* 139.2 (2022), S. 250–259.

[149] **J. H. Reif und H. Wang:** „Social potential fields: A distributed behavioral control for autonomous robots“. In: *Robotics and Autonomous Systems* 27.3 (1999), S. 171–194.

[150] **M. Riedmiller:** „Neural Fitted Q Iteration – First Experiences with a Data Efficient Neural Reinforcement Learning Method“. In: *Machine learning: ECML*

2005. Hrsg. von J. Gama. Bd. 3720. Lecture notes in computer science Lecture notes in artificial intelligence. Berlin und Heidelberg: Springer, 2005, S. 317–328.

[151] **G. F. Riley und T. R. Henderson:** „The ns-3 Network Simulator“. In: *Modeling and Tools for Network Simulation*. Hrsg. von K. Wehrle, M. Güneş und J. Gross. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, S. 15–34.

[152] **T. Ritter:** „PDE-Based Dynamic Data-Driven Monitoring of Atmospheric Dispersion Processes“. Dissertation. Darmstadt: Technische Universität Darmstadt, 2017.

[153] **F. Roettger:** „Reviewing On-Policy/Off-Policy Critic Learning in the Context of Temporal Differences and Residual Learning“. In: *Reinforcement Learning Algorithms: Analysis and Applications*. Hrsg. von B. Belousov, H. Abdulsamad, P. Klink, S. Parisi und J. Peters. Cham: Springer International Publishing, 2021, S. 15–24.

[154] **D. Rolnick, A. Ahuja, J. Schwarz, T. P. Lillicrap und G. Wayne:** „Experience Replay for Continual Learning“. In: *33rd Conference on Neural Information Processing Systems (NeurIPS 2019)* (2019), S. 1–14.

[155] **C. Rösmann:** „Time-optimal nonlinear model predictive control: Direct Transcription Methods with Variable Discretization and Structural Sparsity Exploitation“. Dissertation. Dortmund: Technische Universität Dortmund, 2019.

[156] **C. Rösmann, F. Hoffmann und T. Bertram:** „Integrated online trajectory planning and optimization in distinctive topologies“. In: *Robotics and Autonomous Systems* 88 (2017), S. 142–153.

[157] **C. Rösmann, F. Hoffmann und T. Bertram:** „Kinodynamic trajectory optimization and control for car-like robots“. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, S. 5681–5686.

[158] **C. Rösmann, F. Hoffmann und T. Bertram:** „Timed-Elastic-Bands for time-optimal point-to-point nonlinear model predictive control“. In: *2015 European Control Conference (ECC)*. IEEE, 2015, S. 3352–3357.

[159] **C. Rösmann, M. Krämer, A. Makarow, F. Hoffmann und T. Bertram:** „Exploiting Sparse Structures in Nonlinear Model Predictive Control with Hypergraphs“. In: *2018 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*. IEEE, 2018, S. 1332–1337.

[160] **C. Rösmann, A. Makarow und T. Bertram:** „Online Motion Planning based on Nonlinear Model Predictive Control with Non-Euclidean Rotation Groups“. In: *2021 European Control Conference (ECC)*. IEEE, 2021, S. 1583–1590.

[161] **D. E. Rumelhart, G. E. Hinton und R. J. Williams:** „Learning representations by back-propagating errors“. In: *Nature* 323.6088 (1986), S. 533–536.

[162] **J. Rütter:** „Entwurf und Implementierung effizienter Strukturen zum Aufbau einer Potentialkarte aus Sensordaten in Echtzeit“. Bachelorarbeit. Dortmund: Technische Universität Dortmund, 2021-02-19.

[163] **M. Schranz, M. Umlauft und W. Elmenreich:** „Bottom-up Job Shop Scheduling with Swarm Intelligence in Large Production Plants“. In: *Proceedings of the 11th International Conference on Simulation and Modeling Methodologies, Technologies and Applications*. SCITEPRESS - Science and Technology Publications, 2021, S. 327–334.

[164] **J. Schulman, F. Wolski, P. Dhariwal, A. Radford und O. Klimov:** *Proximal Policy Optimization Algorithms*. 2017. URL: http://arxiv.org/pdf/1707.06347v2.

[165] **E. Schulz:** „Automatisierte Transportmissionsabbildung für eine verteilte nichtlineare modellprädiktive Regelung autonomer Roboterschwärme unter Wissensbeschränkungen“. Bachelorarbeit. Dortmund: Technische Universität Dortmund, 2020-08-21.

[166] **J. Schwarzrock, I. Zacarias, A. L. Bazzan, R. Q. de Araujo Fernandes, L. H. Moreira und E. P. de Freitas:** „Solving task allocation problem in multi Unmanned Aerial Vehicles systems using Swarm intelligence“. In: *Engineering Applications of Artificial Intelligence* 72 (2018), S. 10–20.

[167] **A. Sekar:** *What is the relationship between data rate, SNR, and RSSI?* Hrsg. von I. Aruba Networks. airheads community, 2014. URL: https://community.arubanetworks.com/t5/Controller-Based-WLANs/What-is-the-relationship-between-data-rate-SNR-and-RSSI/ta-p/178312 (besucht am 20. 05. 2019).

[168] **E. Seref:** „Entwicklung und Optimierung eines mathematischen Modells für Pick-und-Place-Roboter mit isochroner Kinematik“. Bachelorarbeit. Dortmund: Technische Universität Dortmund, 2023-08-14.

[169] **A. Shaker:** „Autonome Durchführung bodenvorbereitender Maßnahmen für Bauvorhaben mittels eines Roboterschwarms unter Wissensbeschränkungen“. Bachelorarbeit. Dortmund: Technische Universität Dortmund, 2021-12-01.

[170] **SLAMTEC:** *RPlidar A2*. Hrsg. von SLAMTEC. 2019. URL: https://www.slamtec.ai/home/rplidar_a2/ (besucht am 27. 06. 2023).

[171] **J. Snoeyink und M. van Kreveld:** „Linear-time reconstruction of Delaunay triangulations with applications“. In: *Algorithms — ESA '97*. Hrsg. von R. Burkard und G. Woeginger. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, S. 459–471.

[172] **M. Sokolov, I. Afanasyev, R. Lavrenov, A. Sagitov, L. Sabirova und E. Magid:** „Modelling a crawler-type UGV for urban search and rescue in Gazebo environment“. In: *International Conference on Artificial Life and Robotics (ICAROB 2017)*, IEEE, 2017, S. 360–362.

[173] **M. Sokolov, R. Lavrenov, A. Gabdullin, I. Afanasyev und E. Magid:** „3D modelling and simulation of a crawler robot in ROS/Gazebo“. In: *Proceedings of the 4th International Conference on Control, Mechatronics and Automation*. ACM, 2016, S. 61–65.

[174] **SparkFun Electronics:** *T'REX ROBOT TANK CHASSIS*. Hrsg. von SparkFun Electronics. 2019. URL: https://cdn.sparkfun.com/datasheets/Robotics/T'RexUserManual.pdf (besucht am 27. 06. 2023).

[175] **D. Stahl und J. Hauth:** „PF-MPC: Particle filter-model predictive control“. In: *Systems & Control Letters* 60.8 (2011), S. 632–643.

[176] **R. C. Staudemeyer und E. R. Morris:** *Understanding LSTM – a tutorial into Long Short-Term Memory Recurrent Neural Networks*. 2019. URL: http://arxiv.org/pdf/1909.09586v1.

[177] **A. Strobel:** „Verteilte nichtlineare modellprädiktive Regelung von unbemannten Luftfahrzeug-Schwärmen“. Dissertation. Darmstadt: Technische Universität Darmstadt, 2016.

[178] **I. A. Şucan, M. Moll und L. E. Kavraki:** „The Open Motion Planning Library“. In: *IEEE Robotics & Automation Magazine* 19.4 (2012), S. 72–82.

[179] **R. S. Sutton und A. Barto:** *Reinforcement learning, second edition: An introduction*. Second edition. Adaptive computation and machine learning. Cambridge, Massachusetts und London, England: The MIT Press, 2018.

[180] **R. S. Sutton und A. G. Barto:** *Reinforcement Learning: An Introduction (Adaptive computation and machine learning)*. MIT Press, 1998.

[181] **T. Balch und R. C. Arkin:** „Behavior-based formation control for multirobot teams“. In: *IEEE Transactions on Robotics and Automation* 14.6 (1998), S. 926–939.

[182] **T. Smith, S. Mukhopadhyay, R. R. Murphy, T. Manzini und I. Rodriguez:** „Path Coverage Optimization for USV with Side Scan Sonar for Victim Recovery“. In: *2022 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. IEEE, 2022, S. 160–165.

[183] **B. Tang:** „Orthogonal Array-Based Latin Hypercubes“. In: *Journal of the American Statistical Association* 88.424 (1993), S. 1392–1397.

[184] **W. Teekeng, A. Thammano, P. Unkaw und J. Kiatwuthiamorn:** „A new algorithm for flexible job-shop scheduling problem based on particle swarm optimization“. In: *Artificial Life and Robotics* 21.1 (2016), S. 18–23.

[185] **O. Toul:** „Automatisierte Typenverteilung autonomer Roboterschwärme unter Wissensbeschränkungen“. Bachelorarbeit. Dortmund: Technische Universität Dortmund, 2021-10-11.

[186] **T. Umland und R. Vollmar:** „Verteilte Algorithmen“. In: *Transputerpraktikum*. Hrsg. von T. Umland und R. Vollmar. Wiesbaden: Vieweg+Teubner Verlag, 1992, S. 68–79.

[187] **V. D. Pham, V. Kisel, R. Kirichek, A. Koucheryavy und A. Shestakov:** „Evaluation of A Mesh Network based on LoRa Technology“. In: *2021 23rd International Conference on Advanced Communication Technology (ICACT)*. IEEE, 2021, S. 1–6.

[188] **K. G. Vamvoudakis, Y. Wan, F. L. Lewis und D. Cansever:** *Handbook of Reinforcement Learning and Control*. Bd. 325. Cham: Springer International Publishing, 2021.

[189] **M. Veneri und M. Massaro:** „The effect of Ackermann steering on the performance of race cars“. In: *Vehicle System Dynamics* 59.6 (2021), S. 907–927.

[190] **C. J. C. H. Watkins und P. Dayan:** „Q-learning“. In: *Machine Learning* 8.3-4 (1992), S. 279–292.

[191] **Waveshare Electronics:** *L76X Multi-GNSS Module, GPS, BDS, QZSS*. Hrsg. von Waveshare Electronics. 2019. URL: https://www.waveshare.com/l76x-gps-module.htm (besucht am 27. 06. 2023).

[192] **S. D. Whitehead und L.-J. Lin:** „Reinforcement learning of non-Markov decision processes“. In: *Artificial Intelligence* 73.1 (1995), S. 271–306.

[193] **R. J. Williams und D. Zipser:** „A Learning Algorithm for Continually Running Fully Recurrent Neural Networks“. In: *Neural computation* 1.2 (1989), S. 270–280.

[194] **P. Wilmott:** *Grundkurs Machine Learning*. 1. Auflage. Bonn: Rheinwerk Verlag, 2020.

[195] **F. Witzel:** „Angriffs- und Abwehrstrategien auf mobile Ad-hoc-Netzwerke autonomer Roboterschwärme“. Bachelorarbeit. Dortmund: Technische Universität Dortmund, 2021-10-08.

[196] **M. Xanthidis, I. Rekleitis und J. M. O'Kane:** *RRT+ : Fast Planning for High-Dimensional Configuration Spaces*. 2016. URL: https://arxiv.org/abs/1612.07333.

[197] **J. Xin, J. Zhong, F. Yang, Y. Cui und J. Sheng:** „An Improved Genetic Algorithm for Path-Planning of Unmanned Surface Vehicle“. In: *Sensors (Basel, Switzerland)* 19.11 (2019), S. 1–23.

[198] **Y. Li, D. Dong und X. Guo:** „Mobile Robot Path Planning based on Improved Genetic Algorithm With A-star Heuristic Method“. In: *2020 IEEE 9th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*. IEEE, 2020, S. 1306–1311.

[199] **Y. Peng, D. Qu, Y. Zhong, S. Xie, J. Luo und J. Gu:** „The obstacle detection and obstacle avoidance algorithm based on 2-D lidar“. In: *2015 IEEE International Conference on Information and Automation*. IEEE, 2015, S. 1648–1653.

[200] **A. Yang, W. Naeem, M. Fei und X. Tu:** „A cooperative formation-based collision avoidance approach for a group of autonomous vehicles“. In: *International Journal of Adaptive Control and Signal Processing* 31.4 (2017), S. 489–506.

[201] **K. Yang, S. Moon, S. Yoo, J. Kang, N. L. Doh, H. B. Kim und S. Joo:** „Spline-Based RRT Path Planner for Non-Holonomic Robots“. In: *Journal of Intelligent & Robotic Systems* 73.1-4 (2014), S. 763–782.

[202] **I. Yavrucuk und O. Uzol:** „Panel Method-Based Motion Planning for Swarming MAVs with Probabilistic Target Tracking“. In: *AIAA Guidance, Navigation and Control Conference and Exhibit*. Reston, Virigina: American Institute of Aeronautics and Astronautics, 2007, S. 1–8.

[203] **S. Zacher:** *Übungsbuch Regelungstechnik: Klassische, modell- und wissensbasierte Verfahren*. Wiesbaden: Springer Fachmedien Wiesbaden, 2017.

[204] **D. Zhan und H. Xing:** „Expected improvement for expensive optimization: a review“. In: *Journal of Global Optimization* 78.3 (2020), S. 507–544.

[205] **J.-S. Zhao, X. Liu, Z.-J. Feng und J. S. Dai:** „Design of an Ackermann-type steering mechanism“. In: *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science* 227.11 (2013), S. 2549–2562.

Vorveröffentlichte Beiträge

Teile des hier vorgestellten Inhaltes sind bereits auf Konferenzen und in Fachzeitschriften veröffentlicht worden. Diese Publikationen sowie weitere Beiträge andere Wissenschaftler werden in der folgenden Liste aufgeführt:

[2] **A. Puzicha und P. Buchholz:** „Dynamic Mission Control for Decentralized Mobile Robot Swarms“. In: *2022 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. 2022, S. 257–263.

[139] **A. Puzicha:** „Control of decentralized systems under uncertainty“. In: *Dagstuhl 2020 - Gemeinsamer Workshop des Graduiertenkollegs 2340 und des HPI Forschungskollegs* 2020 (2020).

[140] **A. Puzicha:** „Control of decentralized systems under uncertainty“. In: *Proceedings of the 2021 Joint Workshop of the German Research Training Groups in Computer Science* 2021 (2021), S. 44.

[141] **A. Puzicha:** „Modeling and analysis of a distributed non-linear model-predictive control for swarms of autonomous robots with limited communication skills (in German)“. Masterthesis. Dortmund: TU Dortmund University, 2019.

[142] **A. Puzicha und P. Buchholz:** „A Simulation Environment for Autonomous Robot Swarms with Limited Communication Skills“. In: *Simulation Tools and Techniques*. Hrsg. von H. Song und D. Jiang. Bd. 370. Cham: Springer International Publishing, 2021, S. 206–226.

[143] **A. Puzicha und P. Buchholz:** „Decentralized model predictive control for autonomous robot swarms with restricted communication skills in unknown environments“. In: *Procedia Computer Science* 186 (2021), S. 555–562.

[144] **A. Puzicha und P. Buchholz:** „Mission-based autonomy core for decentralized mobile UGV swarms“. In: *ISR Europe 2022 : 54th International Symposium on Robotics in conjunction with: automatica June 20 – 21, 2022, Munich*. Hrsg. von VDE Verband der Elektrotechnik Elektronik Informationstechnik e.V. Berlin: VDE VERLAG GMBH, 2022, S. 319–326.

[145] **A. Puzicha und P. Buchholz:** „Real-Time Simulation of Robot Swarms with Restricted Communication Skills“. In: *2020 IEEE/ACM 24th International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*. 2020, S. 1–8.

Betreute Abschlussarbeiten

Verschiedene Ideen und Ansätze sind in betreuten Abschlussarbeiten untersucht worden und resultieren zum Teil aus deren Ergebnissen. Der Quelltext und die Messdaten sind in einer überarbeiteten Variante in diese Arbeit eingeflossen. Nicht bestandene Arbeiten sind nicht aufgeführt, deren Erkenntnisse sind jedoch auch eingeflossen. Die Abschlussarbeiten sind folgende:

[10] **F. Barczik:** „Entwurf einer tretraedischen Driftkompensation für präszise Positionierung und Lagerkennung mittels IMU und GPS“. Bachelorarbeit. Dortmund: Technische Universität Dortmund, 2023-05-17.

[28] **C. Dünnermann:** „Kartendatenextraktion für eine modellbasierte Echtzeitsimulation“. Bachelorarbeit. Dortmund: Technische Universität Dortmund, 2021-02-15.

[32] **I. Eigenseher:** „GPU basierte Parallelisierung stützstellenbasierter nichtlinearer modellprädiktiver Regler für autonome Roboterschwärme unter Wissensbeschränkungen“. Bachelorarbeit. Dortmund: Technische Universität Dortmund, 2023-01-18.

[33] **T. Elanko:** „Transportkettenentwicklung für eine verteilte nichtlineare modellprädiktive Regelung autonomer Roboterschwärme unter Wissensbeschränkungen“. Bachelorarbeit. Dortmund: Technische Universität Dortmund, 2021-05-19.

[37] **T. Falkenhahn:** „Simulative Analyse von Angriffen auf Routingverfahren in MANETs“. Bachelorarbeit. Dortmund: Technische Universität Dortmund, 2021-03-29.

[46] **A. Gerwins:** „Globale Pfadplanung im komplexen Umfeld für eine verteilte nichtlineare modellprädikative Regelung autonomer Roboterschwärme unter Wissensbeschränkungen“. Bachelorarbeit. Dortmund: Technische Universität Dortmund, 2023-04-19.

[47] **N. Grabenschröer:** „Rendezvous-basierte Verhaltensstrategien für eine verteilte nichtlineare modellprädiktive Regelung autonomer Roboterschwärme unter Wissensbeschränkungen“. Bachelorarbeit. Dortmund: Technische Universität Dortmund, 2021-04-21.

[48] **O. W. Grajoszek:** „Reinforcement Learning-gestützte Bewegungsplanung von Aktoren“. Bachelorarbeit. Dortmund: Technische Universität Dortmund, 2022-08-15.

[75] **Z. Ji:** „Entwicklung und Analyse von Vergleichstests für Open Motion Planning Library (OMPL) und Advanced Control Particle Belief Propagation (ACPBP)“. Bachelorarbeit. Dortmund: Technische Universität Dortmund, 2023-08-21.

[77] **D. Kaluzny:** „Adaptive Multiskalenrepräsentation von geotopologischen Karten für modellbasierte Echtzeitsimulationen“. Bachelorarbeit. Dortmund: Technische Universität Dortmund, 2021-10-11.

[78] **E. Kement:** „Vergleichende Analyse Gradienten- und Stützstellen-basierter modellprädiktiver Regler für reale Roboterschwärme“. Bachelorarbeit. Dortmund: Technische Universität Dortmund, 2022-05-24.

[80] **Y. Klindworth:** „Entwicklung eines energiegewahren Verhaltens für eine verteilte nichtlineare modellprädiktive Regelung autonomer Roboterschwärme unter Wissensbeschränkungen“. Bachelorarbeit. Dortmund: Technische Universität Dortmund, 2022-04-07.

[97] **K. Leppersjohann:** „Automatisierte Missionsabbildung für eine verteilte nichtlineare modellprädiktive Regelung autonomer Roboterschwärme unter Wissensbeschränkungen“. Bachelorarbeit. Dortmund: Technische Universität Dortmund, 2020-08-20.

[109] **F. Maaßen:** „A Comparison of Fast-Recovery Mechanisms in Networks“. Bachelorarbeit. Dortmund: Technische Universität Dortmund, 2022-05-17.

[111] **J. Maczuga:** „Entwicklung eines realen mobilen ad hoc Netzwerks auf „LoRa“-Basis für Steuerungsdaten eines dezentralen Roboterschwarms unter Wissensbeschränkungen“. Bachelorarbeit. Dortmund: Technische Universität Dortmund, 2022-03-04.

[119] **R. Nader:** „Konsens-basierte Logistikmissionen für eine verteilte nichtlineare modellprädiktive Regelung autonomer Roboterschwärme unter Wissensbeschränkungen“. Bachelorarbeit. Dortmund: Technische Universität Dortmund, 2021-03-31.

[127] **H. Oulahrir:** „Konsens-basierte Logistikmissionen für eine verteilte nichtlineare modellprädiktive Regelung autonomer Roboterschwärme unter Wissensbeschränkungen“. Bachelorarbeit; Dortmund: Technische Universität Dortmund, 2022-09-12.

[162] **J. Rütter:** „Entwurf und Implementierung effizienter Strukturen zum Aufbau einer Potentialkarte aus Sensordaten in Echtzeit“. Bachelorarbeit. Dortmund: Technische Universität Dortmund, 2021-02-19.

[165] **E. Schulz:** „Automatisierte Transportmissionsabbildung für eine verteilte nichtlineare modellprädiktive Regelung autonomer Roboterschwärme unter Wissensbeschränkungen“. Bachelorarbeit. Dortmund: Technische Universität Dortmund, 2020-08-21.

[168] **E. Seref:** „Entwicklung und Optimierung eines mathematischen Modells für Pick-und-Place-Roboter mit isochroner Kinematik“. Bachelorarbeit. Dortmund: Technische Universität Dortmund, 2023-08-14.

[169] **A. Shaker:** „Autonome Durchführung bodenvorbereitender Maßnahmen für Bauvorhaben mittels eines Roboterschwarms unter Wissensbeschränkungen“. Bachelorarbeit. Dortmund: Technische Universität Dortmund, 2021-12-01.

[185] **O. Toul:** „Automatisierte Typenverteilung autonomer Roboterschwärme unter Wissensbeschränkungen“. Bachelorarbeit. Dortmund: Technische Universität Dortmund, 2021-10-11.

[195] **F. Witzel:** „Angriffs- und Abwehrstrategien auf mobile Ad-hoc-Netzwerke autonomer Roboterschwärme“. Bachelorarbeit. Dortmund: Technische Universität Dortmund, 2021-10-08.

A

Publikationen und Abschlussarbeiten

Es werden der Inhalt, der Einfluss auf die vorliegende Arbeit und Beitragsverteilung der Veröffentlichung erläutert. Weiterhin werden die Abschlussarbeiten den jeweiligen Kapiteln zugeordnet.

A.1. Publikationen

In diesem Abschnitt wird für jede Veröffentlichung detailliert der Beitrag der Autoren und der Einfluss sowie der Inhalt dargelegt:

Masterarbeit [141] Aus der Masterarbeit heraus sind die grundlegenden Ideen der Arbeit und die Kontakte für Kooperationen entstanden. Sie hat die erste Version des Simulators realisiert, welche sich jedoch vollständig von dem hier vorgestellten Konzept unterscheidet. Lediglich ein paar der Visualisierungskomponenten sind übernommen worden. Aus den fehlenden Möglichkeiten zur Integration realer Agenten ist das neue Konzept abgeleitet worden. Die Fähigkeit des originalen CPBP-Algorithmus, die Explorationsmission und die einfachen Konvoieskortierungsmission für Roboterschwärme zu lösen, hat das Kapitel 5 und die Verbesserungen zu ACPBP motiviert.

Konferenzpapier [145] Das Konferenzpapier beschäftigt sich mit Laufzeit und Prozessanalyse der in der Masterarbeit entwickelten Simulationsversion. Herr Buchholz hat die Einleitung und übernahm die allgemeine Korrektur verfasst. Die Beschreibung der MPC, der virtuellen Testumgebung, der parallelen Simulation und der Analyse hat Herr Puzicha verfasst. Der Schluss ist gemeinsam formuliert worden.

Kurzfassung: „The paper presents a new approach and a related software environment for the parallel simulation of swarms of autonomous robots in real time. The software environment has been developed for model based analysis of algorithms to control large swarms of distributed autonomous mobile robots communicating over an unreliable and capacity restricted wireless network. It includes a physical simulation of static obstacles, dynamic obstacles with scriptable movement, soil condition, active jammers, static and dynamic link obstacles with configurable damping as well as noise floors. The simulated ground based mobile robots use control particle belief propagation (C-PBP) as a randomized

and sample based model predictive closed loop controller in combination with cost functions to evaluate the situations. We emphasize where the use of shared memory parallelism is beneficial and which inaccuracies in computations are acceptable to increase performance without losing realism.“

Tagungspapier [139] Das Tagungspapier beschreibt grob die Idee der Arbeit und beinhaltet einen kleinen Ausblick auf den Forschungsstand, welcher zum Teil in die Einleitung eingeflossen ist.

Konferenzpapier [142] Das Konferenzpapier beschreibt die in der Masterarbeit entwickelte Simulationsversion. Dabei werden alle Aspekte von Regelungstechnik, über Netzwerktechnik und Missionen, bis zur Visualisierung erläutert. Somit ist dieses Papier ähnlich zu dieser Arbeit aufgebaut, verwendet jedoch nur Erkenntnisse aus der Masterarbeit und beinhaltet keine Integration realer Komponenten. Herr Buchholz hat die Einleitung und übernahm die allgemeine Korrektur verfasst. Bis auf die gemeinsam verfasste Schlussfolgerung ist der restliche Anteil an der Veröffentlichung von Herrn Puzicha verfasst worden.

Kurzfassung: „We present a novel real-time simulation tool for modeling and analyzing a swarm of distributed autonomous mobile robots communicating over an unreliable and capacity restricted communication network. The robots are setup as ground vehicles and use C-PBP [57] as model predictive closed loop controller. This tool offers the ability to simulate rural as well as completely urban scenarios with static obstacles, dynamic obstacles with scripted movement, soil condition, noise floor, active jammers and static and dynamic obstacles for the links with adjustable damping. The goal of this simulation is the analysis of swarm behavior of robots for given missions such as terrain exploration, convoy escorting or creation of a mobile ad hoc network in disaster areas under realistic environmental conditions. “

Konferenzpapier [143] Das Konferenzpapier legt die grundlegende Idee des Simulators und der Explorations-, Fromations-, Transport-, MANET- und einfachen Eskortierungsmission dar. Der Fokus der Arbeit liegt auf den Missionen und der Laufzeiteffizienz des Simulators. Die Erkenntnisse aus den Missionen sind in Kapitel 5.4, 5.5, 5.6, 5.10 und 5.11 eingeflossen. Sie sind allerdings in der Simulationsversion der Masterarbeit untersucht worden. Herr Buchholz hat die Einleitung und übernahm die allgemeine Korrektur verfasst. Bis auf die gemeinsam verfasste Schlussfolgerung ist der restliche Anteil an der Veröffentlichung von Herrn Puzicha verfasst worden.

Kurzfassung: „The paper presents a new approach and a related real-time parallel simulation tool for modeling and analyzing a swarm of more than 60 distributed autonomous mobile robots communicating over an unreliable and capacity restricted wireless communication network. It includes a physical simulation of static obstacles, dynamic obstacles with scriptable movement, soil condition, active jammers, static and dynamic link obstacles with configurable damping as well as thermal noise. The simulated ground based mobile robots use CPBP[57] as

probabilistic model predictive closed loop controller in combination with gradient free cost functions to evaluate complex unsteady goodness aggregations. The goal of this approach is the development of connection aware swarm behavior for complex missions such as terrain exploration, formations, convoy escorting or creation of a mobile ad hoc network in dynamic disaster areas under realistic environmental conditions. The missions can be combined in any manner and the target extraction of the high-level commands for each agent is done implicitly. To validate the developed behavior, the independent software control kernel is able to be used on real robots as well. “

Tagungspapier [140] Das Tagungspapier beschreibt grob die Idee der Arbeit und beinhaltet einen kleinen Ausblick auf den Forschungsstand, welcher zum Teil in die Einleitung eingeflossen ist.

Konferenzpapier [144] Das Konferenzpapier präsentiert sowohl die Erweiterungsaspekte von CPBP hinzu ACPBP als auch das Konzept des Autonomiekerns. Folglich enthält es Inhalte aus den Abschnitten 2.4.3 und 4.2.2. Herr Buchholz hat die Einleitung und übernahm die allgemeine Korrektur verfasst. Bis auf die gemeinsam verfasste Schlussfolgerung, ist der restliche Anteil an der Veröffentlichung von Herrn Puzicha verfasst worden.

Kurzfassung: „Planning of missions by fully decentralized and autonomous robot swarms in dynamically changing disaster areas is a tough challenge. Therefore, this paper provides a novel real time autonomy core for unmanned ground vehicles (UGVs) that generates usable, but not necessarily optimal trajectories based on specific missions. These are defined implicitly, without assigning goal states to single agents, by direct potential functions which may include nonlinearities, unsteadiness, and discrete parts for specifying complex behavior which is achieved without state machines or behavior trees. In addition, the autonomy core is designed to exchange small data packages between agents via unreliable mesh networks based on UDP or LongRange (LoRa) or a physically simulated mesh network inside the simulation.The availability of information from other agents improves the behavior of the swarm but is not required to fulfill a mission. Interfaces to common robotic middleware are provided to integrate simulations and real agents.“

Konferenzpapier [2] Das Konferenzpapier beinhaltet zum Teil die formale Beschreibung der MPC aus Kapitel 2.2.2, fokussiert sich jedoch auf die formale Beschreibung der Missionen Zielansteuerung 5.8, Pfadfolge 5.9, Schwarmtransport 5.11, Rendezvous 5.12 und MANET 5.5. Die durchgeführten Experimente zur Analyse der Missionen unterscheiden sich dabei im Umfang oder in ihrem Aufbau. Herr Buchholz hat die Einleitung und übernahm die allgemeine Korrektur verfasst. Bis auf die gemeinsam verfasste Schlussfolgerung, ist der restliche Anteil an der Veröffentlichung von Herrn Puzicha verfasst worden.

Kurzfassung: „Planning missions by truly autonomous robots is a challenge. This paper presents a novel approach to design mission functions for optimization-based controllers that generate trajectories without explicit goal specifications for

each robot. Potential fields are used to implicitly describe the goal of a mission. This allows one to model a great variety of missions including nonlinearities, discontinuities, and discrete parts. The proposed control algorithm is designed for swarms in which the communication is based on unreliable mesh networks requiring a completely decentralized control. The selection of the missions presented in this paper is mostly requested for disaster areas, defense and security operations, and logistics. Furthermore, experiments express the functionality of the chosen mission functions and the performance of the entire approach.“

A.2. Abschlussarbeiten

Dieser Abschnitt ordnet die betreuten Abschlussarbeiten den Kapiteln dieser Arbeit zu, in die sie eingeflossen sind. Alle nicht eingeflossenen Arbeiten werden unspezifiziert zusammengefasst.

Kapitel 2 [78]

Kapitel 3 [111]

Kapitel 4 [28], [162],[77], [32]

Kapitel 5 [97], [165], [119], [47], [33], [195], [185], [169], [80], [127], [46]

Sonstige Arbeiten [37], [109], [48], [10], [168], [75]

B

Experimente, Daten und Quelltext

B.1. Experimentalrechner

Dieser Abschnitt beschreibt die Spezifikationen der für Experimente verwendeten Rechner.

B.1.1. Regelungstechnik

Der Rechner für die Bewertung der Regelungstechnik hat folgende Spezifikation:

Tabelle B.1.: Spezifikation des Evaluationsrechners für die Regelungstechnik.

Eigenschaft	Wert
Prozessor	AMD® Ryzen™ 9 3900X
Basistakt	3,80 GHz
Arbeitsspeicher	16 GiB
Festplattenspeichertyp	„Solid State Drive“

B.1.2. Simulation

Der Rechner für die Bewertung der Simulation hat folgende Spezifikation:

Tabelle B.2.: Spezifikation des Evaluationsrechners für die Simulation.

Eigenschaft	Wert
Prozessor	Intel® Core™ i9-9900X CPU 9th Generation
CPU-Takt	3,50 GHz
Arbeitsspeicher	64 GiB
Festplattenspeichertyp	„Solid State Drive“
Grafikkarte	NVIDIA® Quadro™ P2000

B.1.3. Missionen

Der Rechner für die Bewertung der Missionen hat folgende Spezifikation:

Tabelle B.3.: Spezifikation des Evaluationsrechners für die Missionen.

Eigenschaft	Wert
Prozessor	Intel® Core™ i9-9900K CPU 9th Generation
CPU-Takt	3,60 GHz
Arbeitsspeicher	32 GiB
Festplattenspeichertyp	Solid State Drive
Grafikkarte	NVIDIA® GeForce™ RTX 2080 Ti

B.2. Umgebungen

Dieser Abschnitt beinhaltet die Umgebungen, die im Vergleichsexperiment aus Abschnitt 2.5.1 verwendet werden.

Tabelle B.4.: Kurzübersicht über die Umgebungen.

Umgebung	Roboter	Hindernisse	Fahrzeuge	Umgebungseinflüsse
1 (B.2.1)	1	1	1	0
2 (B.2.2)	7	11	4	6
3 (B.2.3)	30	0	1	0
4 (B.2.4)	30	84	1	7
5 (B.2.5)	6	8	1	7

B.2.1. Umgebung 1

Quelltext B.1: Umgebung 1

```
environment: null
radiation:
    - type: "Constant"
      goal_value: 0.0
staticobstacles:
    - type: "LineSegmentBoundedQuadratic"
      pos_start:
        - 200
        - 100.0
      pos_end:
        - 1000.0
        - 100.0
      goal_value: 900.0
      radius: 15.0
gatheringpoints:
    - pos:
      - 50.00
      - 200.00
    - pos:
      - 200.00
      - 50.00
goods:
    - pos:
      - 100.00
      - 75.00
    - pos:
      - 100.00
      - 200.00
    - pos:
      - 75.00
      - 100.00
dynamicobstacles: null
radiationstaticobstacles: null
radiationdynamicobstacles: null
konvoi:
    - type: "CommunicationTruck"
      pos:
        - 20.0
        - 20.0
        - 0.0
      velocity:
        - 0.0
        - 0.0
        - 0.0
```

```
robots:
    - pos:
        - 200.0
        - 200.0
        - 0.0
#    - pos:
#        - 210.0
#        - 210.0
#        - 0.0
#    - pos:
#        - 210.0
#        - 200.0
#        - 0.0
radiostations: null
messagebridges:
    - type: "UDP"
      id: 66
      sender_port: 4242
      receiver_port: 4242
      pos:
        - 100.0
        - 100.0
        - 0.0
    - type: "UDP"
      id: 67
      sender_port: 4242
      receiver_port: 4242
      pos:
        - 110.0
        - 110.0
        - 0.0
    - type: "LORA"
      id: 77
      port: "COM3"
      pos:
        - 120.0
        - 120.0
        - 0.0
```

B.2.2. Umgebung 2

Quelltext B.2: Umgebung 2

```
environment:
    - type: "RadialBoundedLinear"
      pos: [50.0, 200.0]
      goal_value: 36.0
      radius: 150.0
    - type: "RadialBoundedLinear"
      pos:
        - 350.0
        - 300.0
      goal_value: 36.0
      radius: 150.0
    - type: "RadialBoundedLinear"
      pos:
        - 400.0
        - 100.0
      goal_value: 20.0
      radius: 75.0
    - type: "RadialBoundedLinear"
      pos:
        - 175.0
        - 325.0
      goal_value: 12.0
      radius: 50.0
radiation:
    - type: "ConstantWithNoise"
      goal_value: 0.0
      noise_scale: 10.0
    - type: "RadialBoundedLinear"
      pos:
        - 175.0
        - 325.0
      goal_value: 30.0
      radius: 150.0
staticobstacles:
    - type: "RectangleQuadratic"
      pos:
        - 100.0
        - 100.0
      goal_value: 600.0
      width:  40.0
      height: 20.0
      radius: 10.0
    - type: "SquareQuadratic"
      pos:
        - 150.0
        - 150.0
      goal_value: 900.0
      side: 30.0
      radius: 0.0
    - type: "RadialBoundedQuadratic"
      pos:
        - 255.0
        - 155.0
      goal_value: 600.0
      radius: 20.0
    - type: "RadialBoundedLinear"
      pos:
        - 340.0
        - 120.0
      goal_value: 400.0
      radius: 40.0
    - type: "LineSegmentBoundedQuadratic"
      pos_start:
        - 255.0
        - 150.0
      pos_end:
        - 270.0
        - 120.0
      goal_value: 900.0
      radius: 15.0
    - type: "LineSegmentBoundedQuadratic"
      pos_start:
       - 270.0
       - 120.0
      pos_end:
        - 310.0
        - 120.0
      goal_value: 900.0
      radius: 15.0
    - type: "Track"
      track_width: 1.435
      pos_start:
       - 200.0
       - 100.0
      pos_end:
        - 310.0
        - 120.0
      goal_value: 0.0
      radius: 15.0
```

```
      track_id: 5
    - type: "Track"
      track_width: 1.435
      pos_start:
       - 200.0
       - 70.0
      pos_end:
        - 380.0
        - 90.0
      goal_value: 0.0
      radius: 15.0
      track_id: 4
gatheringpoints:
#    - pos:
#      - 50.00
#      - 200.00
    - pos:
      - 200.00
      - 50.00
goods:
    - pos:
      - 100.00
      - 75.00
    - pos:
      - 100.00
      - 200.00
    - pos:
      - 75.00
      - 100.00
dynamicobstacles:
    - type: "MoveLinear"
      velocity:
        - 2.0
        - 0.0
      rotation_velocity: 0.07
      staticobject:
        type: "RadialBoundedQuadratic"
        pos:
          - 255.0
          - 155.0
        goal_value: 600.0
        radius: 20.0
radiationstaticobstacles:
    - type: "RadiationStaticLineSegment"
      pos_start:
        - 100.0
        - 100.0
      pos_end:
        - 200.0
        - 150.0
      damping: -20.0
radiationdynamicobstacles:
    - type: "RadiationMoveLinear"
      velocity:
        - 3.0
        - 0.0
      rotation_velocity: 0.0
      staticobject:
        type: "RadiationStaticLineSegment"
        pos_start:
          - 50.0
          - 100.0
        pos_end:
          - 150.0
          - 150.0
        damping: -20.0
konvoi:
    - type: "CommunicationTruck"
      pos:
        - 200.0
        - 175.0
        - 3.0
      velocity:
        - 0.0
        - 0.0
        - 0.0
    - type: "CommunicationTruck"
      pos:
        - 20.0
        - 20.0
        - 0.0
      velocity:
        - 0.0
        - 0.0
        - 0.0
    - type: "Train"
      velocity:
        - 4.0
        - 0.0
        - 0.0
      track_id: 5
      goal_value: 900.0
    - type: "Train"
      velocity:
        - 5.0
        - 0.0
        - 0.0
      track_id: 4
      goal_value: 300.0

robots:
    - pos:
        - 200.0
        - 200.0
        - 0.0
    - pos:
        - 210.0
        - 210.0
        - 0.0
    - pos:
        - 210.0
        - 200.0
        - 0.0
    - pos:
        - 210.0
        - 205.0
        - 0.0
    - pos:
        - 200.0
        - 210.0
        - 0.0
    - pos:
        - 205.0
        - 205.0
        - 0.0
    - pos:
        - 205.0
        - 210.0
        - 0.0
radiostations:
    - pos:
        - 30.0
        - 30.0
        - 0.0
    - pos:
        - 175.0
```

```
        - 200.0
        - 0.0
messagebridges: null
```

B.2.3. Umgebung 3

Quelltext B.3: Umgebung 3

```
environment: null
radiation:
    - type: "Constant"
      goal_value: 0.0
staticobstacles: null
gatheringpoints: null
goods: null
dynamicobstacles: null
radiationstaticobstacles: null
radiationdynamicobstacles: null
konvoi:
    - type: "CommunicationTruck"
      pos:
        - 500.0
        - 475.0
        - 3.0
      velocity:
        - 0.0
        - 0.0
        - 0.0
robots:
    - pos:
        - 500.0
        - 500.0
        - 0.0
    - pos:
        - 510.0
        - 510.0
        - 0.0
    - pos:
        - 510.0
        - 500.0
        - 0.0
    - pos:
        - 510.0
        - 516.0
        - 0.0
    - pos:
        - 510.0
        - 503.0
        - 0.0
    - pos:
        - 510.0
        - 508.0
        - 0.0
    - pos:
        - 510.0
        - 507.0
        - 0.0
    - pos:
        - 450.0
        - 450.0
        - 0.0
    - pos:
        - 500.0
        - 500.0
        - 0.0
    - pos:
        - 510.0
        - 510.0
        - 0.0
    - pos:
        - 510.0
        - 500.0
        - 0.0
    - pos:
        - 510.0
        - 516.0
        - 0.0
    - pos:
        - 510.0
        - 503.0
        - 0.0
    - pos:
        - 510.0
        - 508.0
        - 0.0
    - pos:
        - 510.0
        - 507.0
        - 0.0
    - pos:
        - 500.0
        - 500.0
        - 0.0
    - pos:
        - 510.0
        - 510.0
        - 0.0
    - pos:
        - 510.0
        - 500.0
        - 0.0
    - pos:
        - 510.0
        - 516.0
        - 0.0
    - pos:
        - 510.0
        - 503.0
        - 0.0
    - pos:
        - 510.0
        - 508.0
        - 0.0
    - pos:
        - 510.0
        - 507.0
        - 0.0
    - pos:
        - 500.0
        - 500.0
        - 0.0
    - pos:
        - 510.0
        - 510.0
        - 0.0
    - pos:
        - 510.0
        - 500.0
        - 0.0
```

```
    - pos:
        - 510.0
        - 516.0
        - 0.0
    - pos:
        - 510.0
        - 503.0
        - 0.0
    - pos:
        - 510.0
        - 508.0
        - 0.0
    - pos:
        - 510.0
        - 507.0
        - 0.0
radiostations: null
messagebridges: null
```

B.2.4. Umgebung 4

Quelltext B.4: Umgebung 4

```
environment:
    - type: "RadialBoundedLinear"
      pos: [50.0, 200.0]
      goal_value: 36.0
      radius: 150.0
    - type: "RadialBoundedLinear"
      pos:
        - 550.0
        - 500.0
      goal_value: 36.0
      radius: 150.0
    - type: "RadialBoundedLinear"
      pos:
        - 450.0
        - 400.0
      goal_value: 20.0
      radius: 75.0
    - type: "RadialBoundedLinear"
      pos:
        - 575.0
        - 625.0
      goal_value: 12.0
      radius: 50.0
radiation:
    - type: "ConstantWithNoise"
      goal_value: 0.0
      noise_scale: 10.0
    - type: "RadialBoundedLinear"
      pos:
        - 475.0
        - 425.0
      goal_value: 30.0
      radius: 150.0
    - type: "RadialBoundedLinear"
      pos:
        - 575.0
        - 525.0
      goal_value: 30.0
      radius: 150.0
staticobstacles:
    - type: "RadialBoundedQuadratic"
      pos:
        - 455.0
        - 455.0
      goal_value: 600.0
      radius: 20.0
    - type: "RadialBoundedQuadratic"
      pos:
        - 435.0
        - 415.0
      goal_value: 600.0
      radius: 20.0
    - type: "RadialBoundedQuadratic"
      pos:
        - 515.0
        - 565.0
      goal_value: 600.0
      radius: 20.0
    - type: "RadialBoundedQuadratic"
      pos:
        - 665.0
        - 545.0
      goal_value: 600.0
      radius: 20.0
    - type: "RadialBoundedQuadratic"
      pos:
        - 455.0
        - 655.0
      goal_value: 600.0
      radius: 20.0
    - type: "RadialBoundedLinear"
      pos:
        - 540.0
        - 520.0
      goal_value: 400.0
      radius: 40.0
    - type: "LineSegmentBoundedQuadratic"
      pos_start:
        - 655.0
        - 650.0
      pos_end:
        - 570.0
        - 520.0
      goal_value: 900.0
      radius: 15.0
    - type: "LineSegmentBoundedQuadratic"
      pos_start:
        - 570.0
        - 520.0
      pos_end:
        - 410.0
        - 420.0
      goal_value: 900.0
      radius: 15.0
    - type: "LineSegmentBoundedQuadratic"
      pos_start:
        - 530.0
        - 500.0
      pos_end:
        - 410.0
        - 470.0
      goal_value: 900.0
      radius: 15.0
    - type: "LineSegmentBoundedQuadratic"
      pos_start:
        - 580.0
        - 520.0
      pos_end:
        - 610.0
        - 520.0
```

```
    goal_value: 900.0
    radius: 15.0
  - type: "RadialBoundedQuadratic"
    pos:
      - 455.0
      - 455.0
    goal_value: 600.0
    radius: 20.0
  - type: "RadialBoundedQuadratic"
    pos:
      - 435.0
      - 415.0
    goal_value: 600.0
    radius: 20.0
  - type: "RadialBoundedQuadratic"
    pos:
      - 515.0
      - 565.0
    goal_value: 600.0
    radius: 20.0
  - type: "RadialBoundedQuadratic"
    pos:
      - 665.0
      - 545.0
    goal_value: 600.0
    radius: 20.0
  - type: "RadialBoundedQuadratic"
    pos:
      - 455.0
      - 655.0
    goal_value: 600.0
    radius: 20.0
  - type: "RadialBoundedLinear"
    pos:
      - 540.0
      - 520.0
    goal_value: 400.0
    radius: 40.0
  - type: "LineSegmentBoundedQuadratic"
    pos_start:
      - 655.0
      - 650.0
    pos_end:
      - 570.0
      - 520.0
    goal_value: 900.0
    radius: 15.0
  - type: "LineSegmentBoundedQuadratic"
    pos_start:
      - 570.0
      - 520.0
    pos_end:
      - 410.0
      - 420.0
    goal_value: 900.0
    radius: 15.0
  - type: "LineSegmentBoundedQuadratic"
    pos_start:
      - 530.0
      - 500.0
    pos_end:
      - 410.0
      - 470.0
    goal_value: 900.0
    radius: 15.0
  - type: "LineSegmentBoundedQuadratic"
    pos_start:
      - 580.0
      - 520.0
    pos_end:
      - 610.0
      - 520.0
    goal_value: 900.0
    radius: 15.0
gatheringpoints: null
goods: null
dynamicobstacles:
  - type: "MoveLinear"
    velocity:
      - 2.0
      - 0.0
    rotation_velocity: 0.07
    staticobject:
      type: "RadialBoundedQuadratic"
      pos:
        - 555.0
        - 555.0
      goal_value: 600.0
      radius: 20.0
  - type: "MoveLinear"
    velocity:
      - 2.0
      - 0.0
    rotation_velocity: 0.04
    staticobject:
      type: "RadialBoundedQuadratic"
      pos:
        - 555.0
        - 555.0
      goal_value: 600.0
      radius: 30.0
  - type: "MoveLinear"
    velocity:
      - 1.0
      - 0.0
    rotation_velocity: 0.07
    staticobject:
      type: "RadialBoundedQuadratic"
      pos:
        - 555.0
        - 555.0
      goal_value: 600.0
      radius: 40.0
  - type: "MoveLinear"
    velocity:
      - 3.0
      - 0.0
    rotation_velocity: 0.01
    staticobject:
      type: "RadialBoundedQuadratic"
      pos:
        - 555.0
        - 555.0
      goal_value: 600.0
      radius: 35.0
  - type: "MoveLinear"
    velocity:
      - 3.0
      - 0.0
    rotation_velocity: 0.5
    staticobject:
      type: "RadialBoundedQuadratic"
      pos:
        - 555.0
        - 555.0
      goal_value: 600.0
```

```
      radius: 10.0
  - type: "MoveLinear"
    velocity:
      - 2.0
      - 1.0
    rotation_velocity: 0.06
    staticobject:
      type: "RadialBoundedQuadratic"
      pos:
        - 555.0
        - 555.0
      goal_value: 600.0
      radius: 25.0
  - type: "MoveLinear"
    velocity:
      - 1.0
      - 2.0
    rotation_velocity: 0.07
    staticobject:
      type: "RadialBoundedQuadratic"
      pos:
        - 555.0
        - 555.0
      goal_value: 600.0
      radius: 20.0
  - type: "MoveLinear"
    velocity:
      - 2.0
      - 0.4
    rotation_velocity: 0.05
    staticobject:
      type: "RadialBoundedQuadratic"
      pos:
        - 555.0
        - 555.0
      goal_value: 600.0
      radius: 20.0
  - type: "MoveLinear"
    velocity:
      - 1.0
      - 0.1
    rotation_velocity: 0.01
    staticobject:
      type: "RadialBoundedQuadratic"
      pos:
        - 555.0
        - 555.0
      goal_value: 600.0
      radius: 160.0
  - type: "MoveLinear"
    velocity:
      - 0.0
      - 0.4
    rotation_velocity: 0.01
    staticobject:
      type: "RadialBoundedQuadratic"
      pos:
        - 555.0
        - 555.0
      goal_value: 600.0
      radius: 20.0
  - type: "MoveLinear"
    velocity:
      - 2.0
      - 0.0
    rotation_velocity: 0.07
    staticobject:
      type: "RadialBoundedQuadratic"
      pos:
        - 555.0
        - 555.0
      goal_value: 600.0
      radius: 20.0
  - type: "MoveLinear"
    velocity:
      - 2.0
      - 0.0
    rotation_velocity: 0.04
    staticobject:
      type: "RadialBoundedQuadratic"
      pos:
        - 555.0
        - 555.0
      goal_value: 600.0
      radius: 30.0
  - type: "MoveLinear"
    velocity:
      - 1.0
      - 0.0
    rotation_velocity: 0.07
    staticobject:
      type: "RadialBoundedQuadratic"
      pos:
        - 555.0
        - 555.0
      goal_value: 600.0
      radius: 40.0
  - type: "MoveLinear"
    velocity:
      - 3.0
      - 0.0
    rotation_velocity: 0.01
    staticobject:
      type: "RadialBoundedQuadratic"
      pos:
        - 555.0
        - 555.0
      goal_value: 600.0
      radius: 35.0
  - type: "MoveLinear"
    velocity:
      - 3.0
      - 0.0
    rotation_velocity: 0.5
    staticobject:
      type: "RadialBoundedQuadratic"
      pos:
        - 555.0
        - 555.0
      goal_value: 600.0
      radius: 10.0
  - type: "MoveLinear"
    velocity:
      - 2.0
      - 1.0
    rotation_velocity: 0.06
    staticobject:
      type: "RadialBoundedQuadratic"
      pos:
        - 555.0
        - 555.0
      goal_value: 600.0
      radius: 25.0
  - type: "MoveLinear"
    velocity:
      - 1.0
```

```
        - 2.0
      rotation_velocity: 0.07
      staticobject:
        type: "RadialBoundedQuadratic"
        pos:
          - 555.0
          - 555.0
        goal_value: 600.0
        radius: 20.0
    - type: "MoveLinear"
      velocity:
        - 2.0
        - 0.4
      rotation_velocity: 0.05
      staticobject:
        type: "RadialBoundedQuadratic"
        pos:
          - 555.0
          - 555.0
        goal_value: 600.0
        radius: 20.0
    - type: "MoveLinear"
      velocity:
        - 1.0
        - 0.1
      rotation_velocity: 0.01
      staticobject:
        type: "RadialBoundedQuadratic"
        pos:
          - 555.0
          - 555.0
        goal_value: 600.0
        radius: 160.0
    - type: "MoveLinear"
      velocity:
        - 0.0
        - 0.4
      rotation_velocity: 0.01
      staticobject:
        type: "RadialBoundedQuadratic"
        pos:
          - 555.0
          - 555.0
        goal_value: 600.0
        radius: 20.0
radiationstaticobstacles:
    - type: "RadiationStaticLineSegment"
      pos_start:
        - 100.0
        - 100.0
      pos_end:
        - 300.0
        - 100.0
      damping: -10.0
    - type: "RadiationStaticLineSegment"
      pos_start:
        - 100.0
        - 150.0
      pos_end:
        - 300.0
        - 150.0
      damping: -10.0
    - type: "RadiationStaticLineSegment"
      pos_start:
        - 100.0
        - 200.0
      pos_end:
        - 300.0
        - 200.0
      damping: -10.0
    - type: "RadiationStaticLineSegment"
      pos_start:
        - 100.0
        - 250.0
      pos_end:
        - 300.0
        - 250.0
      damping: -10.0
    - type: "RadiationStaticLineSegment"
      pos_start:
        - 100.0
        - 300.0
      pos_end:
        - 300.0
        - 300.0
      damping: -10.0
    - type: "RadiationStaticLineSegment"
      pos_start:
        - 100.0
        - 350.0
      pos_end:
        - 300.0
        - 350.0
      damping: -10.0
    - type: "RadiationStaticLineSegment"
      pos_start:
        - 100.0
        - 400.0
      pos_end:
        - 300.0
        - 400.0
      damping: -10.0
    - type: "RadiationStaticLineSegment"
      pos_start:
        - 100.0
        - 450.0
      pos_end:
        - 300.0
        - 450.0
      damping: -10.0
    - type: "RadiationStaticLineSegment"
      pos_start:
        - 100.0
        - 500.0
      pos_end:
        - 300.0
        - 500.0
      damping: -10.0
    - type: "RadiationStaticLineSegment"
      pos_start:
        - 100.0
        - 550.0
      pos_end:
        - 300.0
        - 550.0
      damping: -10.0
    - type: "RadiationStaticLineSegment"
      pos_start:
        - 100.0
        - 600.0
      pos_end:
        - 300.0
        - 600.0
      damping: -10.0
    - type: "RadiationStaticLineSegment"
      pos_start:
```

```
      - 100.0
      - 100.0
    pos_end:
      - 300.0
      - 100.0
    damping: -10.0
  - type: "RadiationStaticLineSegment"
    pos_start:
      - 100.0
      - 150.0
    pos_end:
      - 300.0
      - 150.0
    damping: -10.0
  - type: "RadiationStaticLineSegment"
    pos_start:
      - 100.0
      - 200.0
    pos_end:
      - 300.0
      - 200.0
    damping: -10.0
  - type: "RadiationStaticLineSegment"
    pos_start:
      - 100.0
      - 250.0
    pos_end:
      - 300.0
      - 250.0
    damping: -10.0
  - type: "RadiationStaticLineSegment"
    pos_start:
      - 100.0
      - 300.0
    pos_end:
      - 300.0
      - 300.0
    damping: -10.0
  - type: "RadiationStaticLineSegment"
    pos_start:
      - 100.0
      - 350.0
    pos_end:
      - 300.0
      - 350.0
    damping: -10.0
  - type: "RadiationStaticLineSegment"
    pos_start:
      - 100.0
      - 400.0
    pos_end:
      - 300.0
      - 400.0
    damping: -10.0
  - type: "RadiationStaticLineSegment"
    pos_start:
      - 100.0
      - 450.0
    pos_end:
      - 300.0
      - 450.0
    damping: -10.0
  - type: "RadiationStaticLineSegment"
    pos_start:
      - 100.0
      - 500.0
    pos_end:
      - 300.0
      - 500.0
    damping: -10.0
  - type: "RadiationStaticLineSegment"
    pos_start:
      - 100.0
      - 550.0
    pos_end:
      - 300.0
      - 550.0
    damping: -10.0
  - type: "RadiationStaticLineSegment"
    pos_start:
      - 100.0
      - 600.0
    pos_end:
      - 300.0
      - 600.0
    damping: -10.0
radiationdynamicobstacles:
  - type: "RadiationMoveLinear"
    velocity:
      - 0.5
      - 0.0
    rotation_velocity: 0.0
    staticobject:
      type: "RadiationStaticLineSegment"
      pos_start:
        - 50.0
        - 200.0
      pos_end:
        - 50.0
        - 300.0
      damping: -10.0
  - type: "RadiationMoveLinear"
    velocity:
      - 1.0
      - 0.0
    rotation_velocity: 0.0
    staticobject:
      type: "RadiationStaticLineSegment"
      pos_start:
        - 50.0
        - 350.0
      pos_end:
        - 50.0
        - 450.0
      damping: -10.0
  - type: "RadiationMoveLinear"
    velocity:
      - 0.75
      - 0.0
    rotation_velocity: 0.0
    staticobject:
      type: "RadiationStaticLineSegment"
      pos_start:
        - 50.0
        - 500.0
      pos_end:
        - 50.0
        - 600.0
      damping: -10.0
  - type: "RadiationMoveLinear"
    velocity:
      - 0.5
      - 0.0
    rotation_velocity: 0.0
    staticobject:
      type: "RadiationStaticLineSegment"
```

```
      pos_start:
        - 30.0
        - 200.0
      pos_end:
        - 30.0
        - 300.0
      damping: -10.0
  - type: "RadiationMoveLinear"
    velocity:
      - 1.0
      - 0.0
    rotation_velocity: 0.0
    staticobject:
      type: "RadiationStaticLineSegment"
      pos_start:
        - 30.0
        - 350.0
      pos_end:
        - 30.0
        - 450.0
      damping: -10.0
  - type: "RadiationMoveLinear"
    velocity:
      - 0.75
      - 0.0
    rotation_velocity: 0.0
    staticobject:
      type: "RadiationStaticLineSegment"
      pos_start:
        - 30.0
        - 500.0
      pos_end:
        - 30.0
        - 600.0
      damping: -10.0
  - type: "RadiationMoveLinear"
    velocity:
      - 0.7
      - 0.0
    rotation_velocity: 0.0
    staticobject:
      type: "RadiationStaticLineSegment"
      pos_start:
        - 50.0
        - 200.0
      pos_end:
        - 50.0
        - 300.0
      damping: -10.0
  - type: "RadiationMoveLinear"
    velocity:
      - 0.8
      - 0.0
    rotation_velocity: 0.0
    staticobject:
      type: "RadiationStaticLineSegment"
      pos_start:
        - 50.0
        - 350.0
      pos_end:
        - 50.0
        - 450.0
      damping: -10.0
  - type: "RadiationMoveLinear"
    velocity:
      - 1.1
      - 0.0
    rotation_velocity: 0.0
    staticobject:
      type: "RadiationStaticLineSegment"
      pos_start:
        - 50.0
        - 500.0
      pos_end:
        - 50.0
        - 600.0
      damping: -10.0
  - type: "RadiationMoveLinear"
    velocity:
      - 0.6
      - 0.0
    rotation_velocity: 0.0
    staticobject:
      type: "RadiationStaticLineSegment"
      pos_start:
        - 10.0
        - 100.0
      pos_end:
        - 10.0
        - 600.0
      damping: -10.0
  - type: "RadiationMoveLinear"
    velocity:
      - 0.5
      - 0.0
    rotation_velocity: 0.0
    staticobject:
      type: "RadiationStaticLineSegment"
      pos_start:
        - 50.0
        - 200.0
      pos_end:
        - 50.0
        - 300.0
      damping: -10.0
  - type: "RadiationMoveLinear"
    velocity:
      - 1.0
      - 0.0
    rotation_velocity: 0.0
    staticobject:
      type: "RadiationStaticLineSegment"
      pos_start:
        - 50.0
        - 350.0
      pos_end:
        - 50.0
        - 450.0
      damping: -10.0
  - type: "RadiationMoveLinear"
    velocity:
      - 0.75
      - 0.0
    rotation_velocity: 0.0
    staticobject:
      type: "RadiationStaticLineSegment"
      pos_start:
        - 50.0
        - 500.0
      pos_end:
        - 50.0
        - 600.0
      damping: -10.0
  - type: "RadiationMoveLinear"
    velocity:
      - 0.5
```

```
      - 0.0
    rotation_velocity: 0.0
    staticobject:
      type: "RadiationStaticLineSegment"
      pos_start:
        - 30.0
        - 200.0
      pos_end:
        - 30.0
        - 300.0
      damping: -10.0
  - type: "RadiationMoveLinear"
    velocity:
      - 1.0
      - 0.0
    rotation_velocity: 0.0
    staticobject:
      type: "RadiationStaticLineSegment"
      pos_start:
        - 30.0
        - 350.0
      pos_end:
        - 30.0
        - 450.0
      damping: -10.0
  - type: "RadiationMoveLinear"
    velocity:
      - 0.75
      - 0.0
    rotation_velocity: 0.0
    staticobject:
      type: "RadiationStaticLineSegment"
      pos_start:
        - 30.0
        - 500.0
      pos_end:
        - 30.0
        - 600.0
      damping: -10.0
  - type: "RadiationMoveLinear"
    velocity:
      - 0.7
      - 0.0
    rotation_velocity: 0.0
    staticobject:
      type: "RadiationStaticLineSegment"
      pos_start:
        - 50.0
        - 200.0
      pos_end:
        - 50.0
        - 300.0
      damping: -10.0
  - type: "RadiationMoveLinear"
    velocity:
      - 0.8
      - 0.0
    rotation_velocity: 0.0
    staticobject:
      type: "RadiationStaticLineSegment"
      pos_start:
        - 50.0
        - 350.0
      pos_end:
        - 50.0
        - 450.0
      damping: -10.0
  - type: "RadiationMoveLinear"
    velocity:
      - 1.1
      - 0.0
    rotation_velocity: 0.0
    staticobject:
      type: "RadiationStaticLineSegment"
      pos_start:
        - 50.0
        - 500.0
      pos_end:
        - 50.0
        - 600.0
      damping: -10.0
  - type: "RadiationMoveLinear"
    velocity:
      - 0.6
      - 0.0
    rotation_velocity: 0.0
    staticobject:
      type: "RadiationStaticLineSegment"
      pos_start:
        - 10.0
        - 100.0
      pos_end:
        - 10.0
        - 600.0
      damping: -10.0
konvoi:
  - type: "CommunicationTruck"
    pos:
      - 500.0
      - 475.0
      - 3.0
    velocity:
      - 0.0
      - 0.0
      - 0.0
robots:
  - pos:
      - 500.0
      - 500.0
      - 0.0
  - pos:
      - 510.0
      - 510.0
      - 0.0
  - pos:
      - 510.0
      - 500.0
      - 0.0
  - pos:
      - 510.0
      - 516.0
      - 0.0
  - pos:
      - 510.0
      - 503.0
      - 0.0
  - pos:
      - 510.0
      - 508.0
      - 0.0
  - pos:
      - 510.0
      - 507.0
      - 0.0
  - pos:
      - 450.0
```

```
        - 450.0
        - 0.0
    - pos:
        - 500.0
        - 500.0
        - 0.0
    - pos:
        - 510.0
        - 510.0
        - 0.0
    - pos:
        - 510.0
        - 500.0
        - 0.0
    - pos:
        - 510.0
        - 516.0
        - 0.0
    - pos:
        - 510.0
        - 503.0
        - 0.0
    - pos:
        - 510.0
        - 508.0
        - 0.0
    - pos:
        - 510.0
        - 507.0
        - 0.0
    - pos:
        - 450.0
        - 450.0
        - 0.0
    - pos:
        - 500.0
        - 500.0
        - 0.0
    - pos:
        - 510.0
        - 510.0
        - 0.0
    - pos:
        - 510.0
        - 500.0
        - 0.0
    - pos:
        - 510.0
        - 516.0
        - 0.0
    - pos:
        - 510.0
        - 503.0
        - 0.0
    - pos:
        - 510.0
        - 508.0
        - 0.0
    - pos:
        - 510.0
        - 507.0
        - 0.0
    - pos:
        - 500.0
        - 500.0
        - 0.0
    - pos:
        - 510.0
        - 510.0
        - 0.0
    - pos:
        - 510.0
        - 500.0
        - 0.0
    - pos:
        - 510.0
        - 516.0
        - 0.0
    - pos:
        - 510.0
        - 503.0
        - 0.0
    - pos:
        - 510.0
        - 508.0
        - 0.0
    - pos:
        - 510.0
        - 507.0
        - 0.0
radiostations: null
messagebridges: null
```

B.2.5. Umgebung 5

Quelltext B.5: Umgebung 5

```
environment:
    - type: "RadialBoundedLinear"
      pos: [800.0, 200.0]
      goal_value: 36.0
      radius: 150.0
    - type: "RadialBoundedLinear"
      pos:
        - 350.0
        - 300.0
      goal_value: 36.0
      radius: 150.0
    - type: "RadialBoundedLinear"
      pos:
        - 400.0
        - 400.0
      goal_value: 20.0
      radius: 75.0
    - type: "RadialBoundedLinear"
      pos:
        - 175.0
        - 525.0
      goal_value: 12.0
      radius: 50.0
    - type: "RadialBoundedLinear"
      pos:
        - 775.0
        - 425.0
      goal_value: 40.0
      radius: 100.0
    - type: "RadialBoundedLinear"
      pos:
        - 590.0
        - 333.0
      goal_value: 10.0
      radius: 200.0
radiation:
```

```
    - type: "ConstantWithNoise"
      goal_value: 0.0
      noise_scale: 10.0
staticobstacles:
    - type: "RadialBoundedLinear"
      pos:
        - 215.0
        - 155.0
      goal_value: 500.0
      radius: 10.0
    - type: "RadialBoundedLinear"
      pos:
        - 515.0
        - 455.0
      goal_value: 500.0
      radius: 15.0
    - type: "RadialBoundedLinear"
      pos:
        - 340.0
        - 520.0
      goal_value: 400.0
      radius: 20.0
    - type: "RadialBoundedLinear"
      pos:
        - 740.0
        - 320.0
      goal_value: 400.0
      radius: 20.0
    - type: "LineSegmentBoundedLinear"
      pos_start:
        - 250.0
        - 150.0
      pos_end:
        - 270.0
        - 140.0
      goal_value: 300.0
      radius: 5.0
    - type: "LineSegmentBoundedLinear"
      pos_start:
        - 170.0
        - 50.0
      pos_end:
        - 160.0
        - 40.0
      goal_value: 300.0
      radius: 10.0
    - type: "LineSegmentBoundedLinear"
      pos_start:
        - 670.0
        - 50.0
      pos_end:
        - 760.0
        - 120.0
      goal_value: 300.0
      radius: 10.0
    - type: "LineSegmentBoundedLinear"
      pos_start:
        - 230.0
        - 550.0
      pos_end:
        - 460.0
        - 320.0
      goal_value: 300.0
      radius: 10.0
gatheringpoints:
    - pos:
      - 300.00
      - 300.00
    - pos:
      - 600.00
      - 450.00
    - pos:
      - 650.00
      - 80.00
goods:
    - pos:
      - 565.00
      - 391.00
    - pos:
      - 432.00
      - 516.00
    - pos:
      - 130.00
      - 439.00
    - pos:
      - 584.00
      - 220.00
    - pos:
      - 713.00
      - 563.00
    - pos:
      - 310.00
      - 146.00
    - pos:
      - 177.00
      - 527.00
    - pos:
      - 288.00
      - 159.00
    - pos:
      - 593.00
      - 148.00
    - pos:
      - 249.00
      - 243.00
    - pos:
      - 525.00
      - 14.00
    - pos:
      - 753.00
      - 570.00
    - pos:
      - 575.00
      - 523.00
    - pos:
      - 342.00
      - 467.00
    - pos:
      - 611.00
      - 195.00
    - pos:
      - 781.00
      - 410.00
    - pos:
      - 209.00
      - 132.00
    - pos:
      - 388.00
      - 567.00
    - pos:
      - 82.00
      - 320.00
    - pos:
      - 400.00
      - 485.00
    - pos:
```

```
      - 91.00
      - 545.00
    - pos:
      - 877.00
      - 49.00
    - pos:
      - 327.00
      - 101.00
    - pos:
      - 815.00
      - 261.00
    - pos:
      - 257.00
      - 588.00
    - pos:
      - 903.00
      - 371.00
    - pos:
      - 556.00
      - 188.00
dynamicobstacles: null
radiationstaticobstacles: null
radiationdynamicobstacles: null
konvoi:
    - type: "CommunicationTruck"
      pos:
        - 400.0
        - 150.0
        - 3.0
      velocity:
        - 0.0
        - 0.0
        - 0.0
robots:
    - pos:
        - 150.0
        - 100.0
        - 0.0
    - pos:
        - 150.0
        - 150.0
        - 0.0
    - pos:
        - 110.0
        - 100.0
        - 0.0
    - pos:
        - 100.0
        - 110.0
        - 0.0
    - pos:
        - 135.0
        - 90.0
        - 0.0
    - pos:
        - 190.0
        - 200.0
        - 0.0
radiostations: null
messagebridges: null
```


B.3. Konfiguration der Optimierer

Es werden die Konfigurationen der Algorithmen CPBP und ACPBP aufgelistet, so wie sie in den Experimenten verwendet worden sind. Veränderungen dieser Parameter sind stets explizit bei der Beschreibung des Experiments aufgeführt.

Tabelle B.5.: Konfiguration von CPBP und ACPBP

Parameter	CPBP	ACPBP
$\mathbf{u}_-$	$[-0{,}5 \quad -0{,}897\,597\,901\,02]$	$[-0{,}5 \quad -0{,}897\,597\,901\,02]$
$\mathbf{u}_+$	$[3{,}0 \quad 0{,}897\,597\,901\,02]$	$[3{,}0 \quad 0{,}897\,597\,901\,02]$
$\boldsymbol{\mu}_{k,\wp}^{(n)}$	$[1{,}0 \quad 0{,}0]$	$[1{,}0 \quad 0{,}0]$
$\mathbf{C}_e$	$\mathrm{diag}([10 \quad 1])$	$\mathrm{diag}([10 \quad 1])$
σ_0	$[2 \quad 0{,}872\,665]$	$[2 \quad 0{,}872\,665]$
σ_1	$[3 \quad 1{,}745\,329]$	$[3 \quad 1{,}745\,329]$
σ_2	$[30 \quad 17{,}453\,29]$	$[30 \quad 17{,}453\,29]$
σ_x	$[0{,}75 \quad 0{,}75 \quad 0{,}436\,332]$	$[0{,}75 \quad 0{,}75 \quad 0{,}436\,332]$
σ_m	0,25	0,25
K	60	60
N	24	24

B.4. Autonomiekern

Dieser Quelltext veranschaulicht die Kompaktheit der Klassenbeschreibung, die mittels der Modularität, dem Konzept und der Textgenerierungsmakros erzielt wird. Die Makros erzeugen auf Grundlage des gewünschten Namens und Typs Zugriffsfunktionen („Getter“ und „Setter“) für Variablen, Referenzen und Zeiger. Ebenso werden Im- und Exportfunktionen sowie Löschoperatoren für Zeiger generiert. Der größte Nachteil dieser Realisierung ist, dass automatische Dokumentationswerkzeuge Makros in der Regel nicht auflösen können.

Quelltext B.6: Kompakter Entwurf der IRobot-Schnittstelle mit Hilfe der Komponentenstruktur und der Textgenerierungsmakros.

```
#pragma once
#include <puzi-component/puzi-component.h>
#include <puzi-controller/puzi-controller.h>
#include <puzi-actuator/puzi-actuator.h>
#include <puzi-communication/puzi-communication.h>
#include <puzi-network/puzi-network.h>
#include <puzi-model/puzi-model.h>
#include <puzi-mission/puzi-mission.h>
#include <puzi-sensor/puzi-sensor.h>
#include <puzi-system/puzi-system.h>
#include <puzi-map/puzi-map.h>
#include <puzi-sync/puzi-sync.h>
#include <puzi-world/IWorldObject.h>
#include <puzi-world/IVehicle.h>
#include <puzi-world/IStation.h>
#include <puzi-equipment/puzi-equipment.h>

namespace puzi
{
    class IRobot : virtual public IVehicle, virtual public IStation
    {
    public:
        IRobot(ICommunicationComponent* COMP(communication) = nullptr,
            IActuator* COMP(actuator) = nullptr, IController* COMP(controller) = nullptr,
            MapAnalyzerComponent* COMP(map) = nullptr, MissionModule* COMP(mission) = nullptr,
            IOdometry* COMP(odometry) = nullptr, IGPS* COMP(gps) = nullptr,
            ILidar* COMP(lidar) = nullptr, ObstacleDetector* COMP(obstacle_detector) = nullptr,
            TimerComponent* COMP(timer) = nullptr, TimeSyncComponent* COMP(time_sync) = nullptr,
            ModelUpdateComponent* COMP(model_update) = nullptr,
            ModelCreatorComponent* COMP(model_creator) = nullptr,
            RobotModelUpdateComponent* COMP(robot_model_update) = nullptr,
            VehicleModelUpdateComponent* COMP(vehicle_model_update) = nullptr,
            ModelAliveSenderComponent* COMP(alive_sender) = nullptr, ISystem* system = nullptr,
            IStorage* COMP(storage) = nullptr,
            PositionDataContainer* CONTAINER(position) = nullptr,
            MapDataContainer* CONTAINER(map) = nullptr,
            ObstacleDataContainer* CONTAINER(obstacle) = nullptr,
            PlanDataContainer* CONTAINER(plan) = nullptr,
            TimeDataContainer* CONTAINER(time) = nullptr,
            SensorDataContainer* CONTAINER(sensor) = nullptr,
            ModelDataContainer* CONTAINER(model) = nullptr,
            const Eigen::Vector2f& foot_print_size = Eigen::Vector2f(1.f, 1.f))
            : IVehicle(CONTAINER(position), COMP(odometry), foot_print_size),
            IStation(COMP(communication), COMP(alive_sender), COMP(time_sync), foot_print_size),
            C_MEMBER(COMP(actuator)), C_MEMBER(COMP(controller)), C_MEMBER(COMP(map)),
            C_MEMBER(COMP(mission)), C_MEMBER(COMP(gps)), C_MEMBER(COMP(lidar)),
            C_MEMBER(COMP(obstacle_detector)), C_MEMBER(COMP(timer)),
            C_MEMBER(COMP(model_update)), C_MEMBER(COMP(model_creator)),
            C_MEMBER(COMP(robot_model_update)), C_MEMBER(COMP(vehicle_model_update)),
            C_MEMBER(system), C_MEMBER(CONTAINER(map)), C_MEMBER(CONTAINER(obstacle)),
            C_MEMBER(CONTAINER(plan)), C_MEMBER(CONTAINER(time)), C_MEMBER(CONTAINER(sensor)),
```

```
            C_MEMBER(CONTAINER(model)), C_MEMBER(COMP(storage))
        {
            OWNER_FALSE(CONTAINER(map))
            OWNER_FALSE(CONTAINER(obstacle))
            OWNER_FALSE(CONTAINER(plan))
            OWNER_FALSE(CONTAINER(time))
            OWNER_FALSE(CONTAINER(sensor))
            OWNER_FALSE(CONTAINER(model))

            OWNER_FALSE(COMP(communication))
            OWNER_FALSE(COMP(alive_sender))
            OWNER_FALSE(COMP(time_sync))
            OWNER_FALSE(COMP(storage))
        }
        inline virtual void initComponents() override
        {
            IVehicle::initComponents();
            IStation::initComponents();
        }
        inline virtual void connectComponents() override
        {
            IVehicle::connectComponents();
            IStation::connectComponents();
            IStation::setPositionDataContainer(M_CONTAINER(position));
            M_COMP(alive_sender)->setModelType(ModelBasicType::STATION);
        }
#ifdef YAML_SUPPORT
        /**
         * Loads the configuration from a file (yaml)
         */
        inline virtual bool loadConfiguration(const YAML::Node& node)
        {
            YAML_GETTER(MapDataContainer*, CONTAINER(map))
            YAML_GETTER(ObstacleDataContainer*, CONTAINER(obstacle))
            YAML_GETTER(PlanDataContainer*, CONTAINER(plan))
            YAML_GETTER(TimeDataContainer*, CONTAINER(time))
            YAML_GETTER(SensorDataContainer*, CONTAINER(sensor))
            YAML_GETTER(ModelDataContainer*, CONTAINER(model))
            YAML_GETTER(ISystem*, system)

            OWNER_TRUE(CONTAINER(map))
            OWNER_TRUE(CONTAINER(obstacle))
            OWNER_TRUE(CONTAINER(plan))
            OWNER_TRUE(CONTAINER(time))
            OWNER_TRUE(CONTAINER(sensor))
            OWNER_TRUE(CONTAINER(model))

            YAML_GETTER(IController*, COMP(controller))
            YAML_GETTER(IActuator*, COMP(actuator))
            YAML_GETTER(MapAnalyzerComponent*, COMP(map))
            YAML_GETTER(MissionModule*, COMP(mission))
            YAML_GETTER(ModelUpdateComponent*, COMP(model_update))
            YAML_GETTER(IGPS*, COMP(gps))
            YAML_GETTER(ILidar*, COMP(lidar))
            YAML_GETTER(ObstacleDetector*, COMP(obstacle_detector))
            M_COMP(obstacle_detector)->setILidar(M_COMP(lidar));
            YAML_GETTER(TimerComponent*, COMP(timer))

            YAML_GETTER(ModelCreatorComponent*, COMP(model_creator))
            YAML_GETTER(RobotModelUpdateComponent*, COMP(robot_model_update))
            YAML_GETTER(VehicleModelUpdateComponent*, COMP(vehicle_model_update))
            YAML_GETTER(IStorage*, COMP(storage))
            IVehicle::loadConfiguration(node);
            IStation::loadConfiguration(node);
            IStation::setPositionDataContainer(M_CONTAINER(position));
            return true;
        }
        /**
```

```
         * Saves the configuration to a file (yaml)
         */
        inline virtual bool saveConfiguration(YAML::Node& node) const
        {
            YAML_SETTER(CONTAINER(map))
            YAML_SETTER(CONTAINER(obstacle))
            YAML_SETTER(CONTAINER(plan))
            YAML_SETTER(CONTAINER(time))
            YAML_SETTER(CONTAINER(sensor))
            YAML_SETTER(CONTAINER(model))
            YAML_SETTER(system)

            YAML_SETTER(COMP(controller))
            YAML_SETTER(COMP(actuator))
            YAML_SETTER(COMP(map))
            YAML_SETTER(COMP(mission))
            YAML_SETTER(COMP(model_update))
            YAML_SETTER(COMP(gps))
            YAML_SETTER(COMP(lidar))
            YAML_SETTER(COMP(obstacle_detector))
            YAML_SETTER(COMP(timer))

            YAML_SETTER(COMP(model_creator))
            YAML_SETTER(COMP(robot_model_update))
            YAML_SETTER(COMP(vehicle_model_update))
            YAML_SETTER(COMP(storage))
            IVehicle::saveConfiguration(node);
            YAML_SAVE_PARENT(IStation)
        }
#endif
        COMP_INIT
        {
            IVehicle::init();
            IStation::init();
        }
        COMP_START
        {
            IVehicle::start();
            IStation::start();
        }
        COMP_ACTIVATE
        {
            IVehicle::activate();
            IStation::activate();
        }
        COMP_DEACTIVATE
        {
            IVehicle::deactivate();
            IStation::deactivate();
        }
        COMP_STOP
        {
            IVehicle::stop();
            IStation::stop();
        }
        COMP_UNLOAD
        {
            IVehicle::unload();
            IStation::unload();
        }
        inline PositionDataContainer* getPositionDataContainer()
        {
            return IVehicle::getPositionDataContainer();
        }
        inline void setPositionDataContainer(PositionDataContainer* CONTAINER(position)) override
        {
            IVehicle::setPositionDataContainer(CONTAINER(position));
            if (m_initialized)
```

```
                IStation::setPositionDataContainer(CONTAINER(position));
        }
    protected:
        CREATE_FULL_PTR_OWNER(MapDataContainer, CONTAINER(map))
        CREATE_FULL_PTR_OWNER(ObstacleDataContainer, CONTAINER(obstacle))
        CREATE_FULL_PTR_OWNER(PlanDataContainer, CONTAINER(plan))
        CREATE_FULL_PTR_OWNER(TimeDataContainer, CONTAINER(time))
        CREATE_FULL_PTR_OWNER(SensorDataContainer, CONTAINER(sensor))
        CREATE_FULL_PTR_OWNER(ModelDataContainer, CONTAINER(model))
        CREATE_FULL_PTR(ISystem, system)

        CREATE_FULL_PTR(IController, COMP(controller))
        CREATE_FULL_PTR(IActuator, COMP(actuator))
        CREATE_FULL_PTR(MapAnalyzerComponent, COMP(map))
        CREATE_FULL_PTR(MissionModule, COMP(mission))
        CREATE_FULL_PTR(ModelUpdateComponent, COMP(model_update))
        CREATE_FULL_PTR(IGPS, COMP(gps))
        CREATE_FULL_PTR(ILidar, COMP(lidar))
        CREATE_FULL_PTR(ObstacleDetector, COMP(obstacle_detector))
        CREATE_FULL_PTR(TimerComponent, COMP(timer))

        CREATE_FULL_PTR(ModelCreatorComponent, COMP(model_creator))
        CREATE_FULL_PTR(VehicleModelUpdateComponent, COMP(vehicle_model_update))
        CREATE_FULL_PTR(RobotModelUpdateComponent, COMP(robot_model_update))

        CREATE_FULL_PTR_OWNER(IStorage, COMP(storage))
    };
} // namespace puzi
```

C

Grundlagen der Regelungstechnik, Zuverlässigkeitsanalyse und Beweisidee

Dieses Kapitel enthält die Grundlagen des Proportional-Integral-Differential-Reglers mit „Anti-Windup“, die Aussage des Abtasttheorems und einen formalen Ansatz zur Konvergenzberechnung auf Grundlage der Zuverlässigkeitsanalyse.

C.1. Proportional-Integral-Differential-Regler mit „Anti-Windup“

Der Proportional-Integral-Differential-Regler (kurz PID) mit „Anti-Windup“ erhält den Reglerabweichungsvektor **e** und reicht diesen an die drei Stellglieder des Reglers weiter. Diese geben den summierten Stellgrößenvektor aus, der außerhalb der Systembeschränkung liegt. Daher wird dieser begrenzt. Wird die Stellgröße kontinuierlich begrenzt so wächst der Integralanteil stetig an. Dies kann dazu führen, dass bei einem Wechsel des Arbeitspunktes oder nach dem Einregeln des Systems der Integralanteil für eine längere Zeit existent bleibt, bis er abgebaut worden ist. Dieses Verhalten kann die Dynamik des Reglers reduzieren und dazu führen, dass das System nicht ausgeregelt werden kann. Folglich wird der Integralanteil um den Anteil reduziert, der außerhalb der Beschränkung liegt. Der Subtrahend wird mit Faktor a_f skaliert. In Abbildung C.1 beschreiben k_d, k_p, k_i die Verstärkungsfaktoren und $[\mathbf{u}]_{\min}^{\max}$ die begrenzte Stellgröße (vgl. [3]).

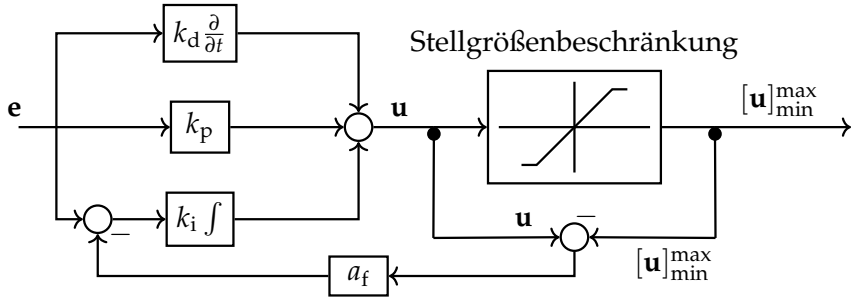


Abbildung C.1.: Proportional-Integral-Differential-Regler mit „Anti-Windup“

C.2. Abtasttheorem

Das Nyquist-Shannon-Abtasttheorem ist ein Prinzip der digitalen Signalverarbeitung, das den Zusammenhang zwischen dem Frequenzbereich eines Signals und der Abtastrate, die zur Vermeidung einer Signalverzerrung, dem so genannten Aliasing, erforderlich ist, definiert. Das Theorem besagt, dass die Abtastrate mehr als das Doppelte der Bandbreite des Signals betragen muss, um Aliasing-Verzerrungen zu vermeiden. Somit muss ein Systemausgang stets mit mehr als der doppelten Rate abgetastet werden, als das Inverse der kleinste Zeitkonstante im System beträgt (vgl. [3]).

C.3. Hindernisdimensionen

Für die Vermeidung von Kollisionen bzw. für die Zuverlässigkeitsanalyse muss gelten, dass die Agenten in der Lage sind, dynamischen Hindernissen auszuweichen. Dazu muss deren Systemdynamik, hier vereinfacht als maximale Geschwindigkeit v_+^F eines Fahrzeuges F, beschränkt sein. Für die Bestimmung dieser Geschwindigkeit sei die maximale Geschwindigkeit der Agenten mit v_+^R, deren Hindernisdetektionszeit mit t_d und die Trajektorienberechnungszeit mit t_c gegeben. Die Gesamtzeit t_{ges}, die ein Agent für das Ausweichen zur Verfügung hat, berechnet sich aus v_+^F und dem Sensorradius $r(\overset{\ltimes}{\mathbf{s}})$ eines Sensors $\overset{\ltimes}{\mathbf{s}}$. Fahrzeuge mit einem weiteren Abstand können nicht erfasst werden.

$$t_{\text{ges}} = \frac{r(\overset{\ltimes}{\mathbf{s}})}{v_+^F} \tag{C.3.1}$$

$$= t_d + t_c + t_{\text{rest}} \tag{C.3.2}$$

Fährt das Fahrzeug F frontal auf den Agenten zu, so hat dieser t_{rest} Zeit, um der halben Fahrzeugbreite $\frac{b^F}{2}$ auszuweichen.

$$t_{\text{rest}} = \frac{b^F}{2v_+^R} \tag{C.3.3}$$

$$\Rightarrow v_+^F(b^F) = \frac{r(\overset{\ltimes}{\mathbf{s}})}{t_d + t_c + \frac{b^F}{2v_+^R}} \tag{C.3.4}$$

Somit kann die maximale Fahrzeuggeschwindigkeit v_+^F anhand dessen Breite b^F bestimmt werden. Unabhängig von der Geschwindigkeit der Agenten gibt es eine obere Schranke, die durch den Sensor und die Reaktionszeiten t_d und t_c bestimmt wird.

C.4. Zuverlässigkeitsanalyse

Dieser Abschnitt beruht auf der Veröffentlichung Calafiore, Dabbene und Tempo [18]. Die Zuverlässigkeit wird auf Grundlage einer Verletzungswahrscheinlichkeit V einer

unsicheren Eigenschaft $\mathbf{q} \in \mathbb{Q}$ definiert, für die ein unzulässiger Bereich $\mathbb{Q}_{\text{viol}} \subset \mathbb{Q}$ existiert.

$$V = \frac{\mathbb{Q}_{\text{viol}}}{\mathbb{Q}} \tag{C.4.1}$$

Die Zuverlässigkeit R bildet die komplementäre Wahrscheinlichkeit.

$$R = 1 - V \tag{C.4.2}$$

Für ein regelungstechnisches System sei $\mathbf{u}_k$ mit $k = 0, 1, \ldots, K$ eine zeitdiskrete Stellgrößenfolge mit Horizontlänge K und $\mathbf{x}_k$ dessen Systemausgang bzw. Systemzustand. Die Größen seien jeweils durch $\mathbf{u}_-, \mathbf{u}_+, \mathbf{x}_-$ und $\mathbf{x}_+$ begrenzt. Die Stellgröße wird mittels einer Kostenfunktion J bewertet und soll minimiert bzw. stabilisiert werden. Dafür lässt sich folgendes Optimierungsproblem definieren:

$$\min \eta \tag{C.4.3}$$

mit den Nebenbedingungen

$$J(\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_K) \leq \eta \tag{C.4.4}$$

$$\mathbf{u}_- \leq \mathbf{u}_k \leq \mathbf{u}_+ \tag{C.4.5}$$

$$\mathbf{x}_- \leq \mathbf{x}_k \leq \mathbf{x}_+ \tag{C.4.6}$$

$$k = 0, 1, \ldots, K \tag{C.4.7}$$

Das System enthalte nun eine unsichere Eigenschaft definiert über den Parameter $\mathbf{q}$. So lässt sich eine Leistungsfunktion $f(\xi, \mathbf{q})$ über den Vektor ξ und Eigenschaft q definieren.

$$\xi := \begin{bmatrix} \eta & \mathbf{u}_0^T & \mathbf{u}_1^T \ldots \mathbf{u}_K^T \end{bmatrix}^T \tag{C.4.8}$$

$$f(\xi, \mathbf{q}) = \max\{J - \eta, \max_{k=1,\ldots,K}\{\mathbf{x}_k - \mathbf{x}_+, \mathbf{x}_- - \mathbf{x}_k\}, \tag{C.4.9}$$

$$\max_{k=1,\ldots,T}\{\mathbf{u}_k - \mathbf{u}_+, \mathbf{u}_- - \mathbf{u}_k\}\}$$

$$V(\xi) = \mathfrak{P}\{\mathbf{q} \in \mathbb{Q} : f(\xi, \mathbf{q}) > 0\} \tag{C.4.10}$$

$$V(\xi) \leq \epsilon \tag{C.4.11}$$

C.4.1. Schätzung der Wahrscheinlichkeit und der Extrema

Es sei eine Leistungsfunktion $f(\mathbf{q})$ für ein allgemeines unsicheres dynamisches System definiert, wobei $\mathbf{q} \in \mathbb{Q}$ ein Vektor von zufälligen unsicheren Parametern und $\mathbb{Q} \subseteq \mathbb{R}^l$ ein gegebener Unsicherheitsbereich ist. Ein Beispiel für ein System sei:

$$f(\mathbf{q}) = \begin{cases} \infty \text{ , wenn System instabil} \\ \text{ein beschränkter Wert, sonst} \end{cases} \tag{C.4.12}$$

Problem C.4.1 (Zuverlässigkeitsschätzung)**:** *Sei $\gamma > 0$, schätze die Zuverlässigkeit der Eigenschaft $f(\mathbf{q}) \leq \gamma$.*

$$R = \mathfrak{P}\{f(\mathbf{q}) \leq \gamma\} \tag{C.4.13}$$

$\Gamma(\cdot) \in \{0,1\}$ bezeichnet die Indikatorfunktion und wird zu 1 ausgewertet, falls die Aussage wahr ist. Mit Hilfe des Monte Carlo Ansatzes kann die empirische Zuverlässigkeit mittels N unabhängig gleichverteilter Stichproben bestimmt werden.

$$\hat{R}_N = \frac{1}{N}\sum_{i=1}^{N}\Gamma(f(q^{(i)}) \leq \gamma) \tag{C.4.14}$$

$$\lim_{N\to\infty}\hat{R}_N \to R \tag{C.4.15}$$

Auf Basis der Hoeffending-Ungleichung (vgl. [18]) kann die Obergrenze des empirischen Fehlers festgelegt werden:

$$\mathfrak{P}^N\{|\hat{R}_N - R| \geq \epsilon_H\} \leq 2\cdot\exp\{-2N\epsilon_H^2\} \tag{C.4.16}$$

Sei $\epsilon_H \in (0,1)$ eine a priori Genauigkeit und $\delta \in (0,1)$ mit $2\exp\{-2N\epsilon_H^2\} \leq \delta$ eine Konfidenzintervallbreite. Die additive Chernoff-Schranke liefert anhand dessen die erforderliche Anzahl von Stichproben:

$$N \geq \frac{1}{2\epsilon_H^2}\log\frac{2}{\delta} \tag{C.4.17}$$

Somit gilt $\hat{R}_N$ ist $\epsilon-$nah an R.

Problem C.4.2 (Leistungsniveauschätzung)**:** *Für ein $\epsilon \in (0,1)$ kann ein Leistungsniveau γ geschätzt werden, bei dem $f(q) \leq \gamma$ mit einer Zuverlässigkeit von mindestens $1-\epsilon$ gilt. Das bedeutet, es ist ein γ zu finden, für das gilt:*

$$\mathfrak{P}\{f(q) \leq \gamma\} \geq 1-\epsilon \tag{C.4.18}$$

Hier für kann ein erneut ein empirischer Wert ermittelt werden.

$$\gamma_N = \max_{i=1,\ldots,N} f(q^i) \tag{C.4.19}$$

Dabei berechnet sich die Stichprobengröße N für kleine ϵ und einer Glaubwürdigkeit bzw. Konfidenz $(1-\delta)$ wie folgt (vgl. [18]):

$$N \geq \frac{1}{\epsilon}\log\frac{1}{\delta} \tag{C.4.20}$$

C.5. Beweisidee für die Konvergenzwahrscheinlichkeit eines Reglers

Aufbauend auf den Erkenntnissen und Ideen der Zuverlässigkeit kann nun versucht werden die Wahrscheinlichkeit der Konvergenz eines Reglers zu bestimmen. Dazu werden Mengen und Berechnungen sowie Analogien Schritt für Schritt betrachtet.

1. Bestimme zunächst das gesamte Volumen bzw. die Gesamtenge, definiert durch das Kreuzprodukt der Trajektorienräume.

Definition C.5.1 (Gesamtvolumen)**:** *Das gesamte Volumen ist der konvexe, abgeschlossene und kompakte Raum der zeitlichen Trajektorien:*

$$\mathcal{E} := \mathcal{X} \times \mathbb{T} \tag{C.5.1}$$

Seien $\mathcal{X}$ der Zustandsraum, $\mathbb{T} = \mathbb{R}_0^+$ die Zeit und $\mathcal{U}$ der Stellgrößenraum.

2. Mit Hilfe der Definition des Zielvolumens kann das optimale Volumen beschrieben werden.

Definition C.5.2 (Zielvolumen)**:** *Das Zielvolumen $\mathcal{G}$ für einen zeitkontinuierlichen Zielzustand $\overset{\circ}{\mathbf{x}}_{\mathbf{g}}(t) \in \mathcal{X}$ mit dem Ziel $\mathbf{g}$ beschreibt die Menge an Zuständen, von denen aus ein Zielzustand unter Minimierung einer Kostenfunktion erreicht wird. Das Zielvolumen definiert die Menge an Zuständen $\overset{\circ}{\mathbf{x}}(t) \in \mathbb{X}_{t_n}(T)$ aus der Menge der zeitabhängigen zulässigen Zustände $\mathbb{X}_{t_n}(T)$ vom Startzeitpunkt t_n bis zur Zielzeit T mit $t \in [t_n, T]$, für die gilt, dass eine optimale zeitabhängige Stellgröße $\overset{\circ}{\mathbf{u}}(t) \in \mathbb{U}_{t_n}^{\mathbb{X}_{t_n}(T)}$ existiert, die $\overset{\circ}{\mathbf{x}}(t)$ mit Hilfe der Übertragungsfunktion $\mathfrak{f}$ in $\mathbf{x}_{\mathbf{g}}(T)$ zum Zeitpunkt T überführt, und dass für jede andere Stellgröße $\overset{\circ}{\mathbf{u}}'(t) \in \mathbb{U}_{t_n}^{\mathbb{X}_{t_n}(T)}$, die nicht in $\mathbf{x}_{\mathbf{g}}(T)$ endet, gilt, dass deren Kosten, bewertet durch die Kostenfunktion l, größer sind, als die durch $\overset{\circ}{\mathbf{u}}(t)$ generierten Kosten. $\mathbb{U}_{t_n}^{\mathbb{X}_{t_n}(T)}$ beschreibt die Menge der zeit- und zustandsabhängigen zulässigen Stellgrößen. Sei*

$$t \in [t_n, T] \tag{C.5.2}$$

$$\begin{aligned}
&\mathcal{E} \supseteq \mathcal{G}(\mathbf{x}_{\mathbf{g}}(t), t_n, T, l(\overset{\circ}{\mathbf{x}}(t), \overset{\circ}{\tilde{\mathbf{u}}}(\overset{\circ}{\mathbf{x}}(t), t), t), \mathfrak{f}(x(t), \overset{\circ}{\tilde{\mathbf{u}}}(\overset{\circ}{\mathbf{x}}(t), t), t)) := \{\overset{\circ}{\mathbf{x}}(t) \in \mathbb{X}_{t_n}(T) | \\
&\exists \overset{\circ}{\tilde{\mathbf{u}}}(\overset{\circ}{\mathbf{x}}(t), t) \in \mathbb{U}_{t_n}^{\mathbb{X}_{t_n}(T)}(\overset{\circ}{\mathbf{x}}(t), t_n, T).\overset{\circ}{\mathbf{x}}_{\mathbf{g}}(T) = \int_{t_n}^{T} \mathfrak{f}(\overset{\circ}{\mathbf{x}}(t), \overset{\circ}{\tilde{\mathbf{u}}}(\overset{\circ}{\mathbf{x}}(t), t), t) dt. \\
&\forall \overset{\circ}{\tilde{\mathbf{u}}}'(\overset{\circ}{\mathbf{x}}(t), t) \in \mathbb{U}_{t_n}^{\mathbb{X}_{t_n}}(\overset{\circ}{\mathbf{x}}(t), t_n, T).\overset{\circ}{\mathbf{x}}_{\mathbf{g}}(T) \neq \int_{t_n}^{T} \mathfrak{f}(\overset{\circ}{\mathbf{x}}(t), \overset{\circ}{\tilde{\mathbf{u}}}'(\overset{\circ}{\mathbf{x}}(t), t), t) dt \implies \\
&\qquad \int_{t_n}^{T} l(\overset{\circ}{\mathbf{x}}(t), \overset{\circ}{\tilde{\mathbf{u}}}(\overset{\circ}{\mathbf{x}}(t), t), t) dt \leq \int_{t_n}^{T} l(\overset{\circ}{\mathbf{x}}(t), \overset{\circ}{\mathbf{x}}(t), \overset{\circ}{\tilde{\mathbf{u}}}'(\overset{\circ}{\mathbf{x}}(t), t), t) dt\}
\end{aligned} \tag{C.5.3}$$

Definition C.5.3 (Optimales Volumen)**:** *Das optimale Volumen $\mathbb{O} \subseteq \mathcal{E}$ besitzt folgende Eigenschaften:*

a) Das optimale Volumen ist nicht leer: $\mathbb{O} \neq \emptyset$.

b) Es wird als die Vereinigung der Zielvolumina definiert, deren Zielzustände die

globalen Minima der Kostenfunktion l über die gesamte Zeit bilden:

$$\mathbb{O}(t_n, T, l(\overset{\circ}{\mathbf{x}}(t), \overset{\circ}{\tilde{\mathbf{u}}}(\overset{\circ}{\mathbf{x}}(t), t), t), \mathfrak{f}(\overset{\circ}{\mathbf{x}}(t), \overset{\circ}{\tilde{\mathbf{u}}}(\overset{\circ}{\mathbf{x}}(t), t), t)) \tag{C.5.4}$$

$$:= \bigcup \mathcal{G}(\overset{\circ}{\mathbf{x}}_*(t), t_n, T, l(\overset{\circ}{\mathbf{x}}(t), \overset{\circ}{\tilde{\mathbf{u}}}(\overset{\circ}{\mathbf{x}}(t), t), t), \mathfrak{f}(\overset{\circ}{\mathbf{x}}(t), \overset{\circ}{\tilde{\mathbf{u}}}(\overset{\circ}{\mathbf{x}}(t), t), t))$$

$$|\exists \overset{\circ}{\tilde{\mathbf{u}}}(\overset{\circ}{\mathbf{x}}_*(t), t) \in \mathbb{U}_{t_n}^{\mathbb{X}_{t_n}}(\overset{\circ}{\mathbf{x}}_*(t), t_n, T) \forall \overset{\circ}{\mathbf{x}} \in \mathbb{X}_{t_n}(T) \exists \overset{\circ}{\tilde{\mathbf{u}}}' \in \mathbb{U}_{t_n}^{\mathbb{X}_{t_n}}(\overset{\circ}{\mathbf{x}}(t), t_n, T).$$

$$\int_{t_n}^{T} l(\overset{\circ}{\mathbf{x}}_*(t), \overset{\circ}{\tilde{\mathbf{u}}}(\overset{\circ}{\mathbf{x}}_*(t), t), t) \leq \int_{t_n}^{T} l(\overset{\circ}{\mathbf{x}}(t), \overset{\circ}{\tilde{\mathbf{u}}}'(\overset{\circ}{\mathbf{x}}_*(t), t), t) \tag{C.5.5}$$

c) Das optimale Volumen $\mathbb{O}$ beinhaltet mindestens einen Häufungspunkt oder ein Intervall und ist keine finite bzw. diskrete Menge, weil sonst das Kardinalitätsverhältnis von $\mathbb{O}$ zu $\mathcal{E}$ gegen Null geht und somit die Wahrscheinlichkeit in diese Menge zu gelangen gegen Null geht.

3. Als nächstes wird das erreichbare Zustandsvolumen eines Startzustandes als die Menge der erreichbaren Zustände definiert.

Definition C.5.4 (Erreichbares Volumen)**:** *Das erreichbare Volumen $\mathcal{R}$ ist die Menge an Zuständen $\overset{\circ}{\mathbf{x}}_{\mathcal{R}}(t)$, die von einem initialen $\overset{\circ}{\mathbf{x}}_0(t)$ Zustand aus erreicht werden können.*

$$\mathcal{R}(\overset{\circ}{\mathbf{x}}_0(t), t_n, T) := \{\overset{\circ}{\mathbf{x}}_{\mathcal{R}}(t) \in \mathbb{X}_{t_n}(T) \tag{C.5.6}$$
$$|\exists \overset{\circ}{\tilde{\mathbf{u}}}(\overset{\circ}{\mathbf{x}}(t), t) \in \mathbb{U}_{t_n}^{\mathbb{X}_{t_n}}(\overset{\circ}{\mathbf{x}}_0(t), t_n, T).\mathfrak{f}(\overset{\circ}{\mathbf{x}}_0, \overset{\circ}{\tilde{\mathbf{u}}}(\overset{\circ}{\mathbf{x}}_0, T), T) = \overset{\circ}{\mathbf{x}}_{\mathcal{R}}(t)\}$$

Durch den Schnitt des erreichbaren Volumens und des optimalen Volumens kann die Wahrscheinlichkeit zur Konvergenz angegeben werden.

Definition C.5.5 (Konvergenzzustandswahrscheinlichkeit)**:** *Die Wahrscheinlichkeit einen konvergierenden Zustand zu erreichen wird als das Kardinalitätsverhältnis des Schnittes des optimalen Volumens $\mathbb{O}$ mit dem erreichbaren Volumen $\mathcal{R}$ zum erreichbaren Volumen selbst definiert:*

$$\mathfrak{P}_{\mathcal{W}} = \frac{|\mathbb{O} \cap \mathcal{R}|}{|\mathcal{E} \cap \mathcal{R}|} = \frac{|\mathbb{O} \cap \mathcal{R}|}{|\mathcal{R}|} \tag{C.5.7}$$

$$0 \leq \mathfrak{P}_{\mathcal{W}}(\overset{\circ}{\mathbf{x}}_0(t), t_n, T) = \frac{\int_{t_n}^{T} \frac{|\mathbb{O}(t, T, l(\cdot), \mathfrak{f}(\cdot))|}{|\mathcal{R}(\overset{\circ}{\mathbf{x}}_0(t), t_n, t)|} dt}{T - t_n} \leq 1 \tag{C.5.8}$$

4. Anschließend kann von einem Zustand $\overset{\circ}{\mathbf{x}}(t)$ aus die Konvergenzwahrscheinlichkeit $\mathfrak{P}_{\text{Kon}}$ skizziert werden, indem die Wahrscheinlichkeit $\mathfrak{P}_{\text{Eintreten in } \mathbb{O}}$ in die Menge $\mathbb{O}$ überzugehen mit der rekursiven Wahrscheinlichkeit $\mathfrak{P}_{\text{bleiben in } \mathbb{O} \text{ (rekursiv)}}$

des Folgezustands $f(\overset{\circ}{\mathbf{x}}(t), \overset{\circ}{\tilde{\mathbf{u}}}(\overset{\circ}{\mathbf{x}}(t), t), t)$ in der Menge zu bleiben multipliziert wird.

$$\begin{aligned}\mathfrak{P}_{\text{Kon}}(\overset{\circ}{\mathbf{x}}(t)) &= \mathfrak{P}_{\text{Eintreten in } \mathbb{O}} \cdot \mathfrak{P}_{\text{drinbleiben in } \mathbb{O} \text{ (rekursiv)}} \\ &= \int_{\mathbb{U}} \omega(\overset{\circ}{\tilde{\mathbf{u}}}(\overset{\circ}{\mathbf{x}}(t), t)) \mathrm{d}\overset{\circ}{\tilde{\mathbf{u}}}(\overset{\circ}{\mathbf{x}}(t), t) \cdot \mathfrak{P}_{\text{Kon}}(f(\overset{\circ}{\mathbf{x}}(t), \overset{\circ}{\tilde{\mathbf{u}}}(\overset{\circ}{\mathbf{x}}(t), t), t)) \end{aligned} \tag{C.5.9}$$

Dabei sei ω die Wahrscheinlichkeitsdichtefunktion der Stellgröße. Das bedeutet, mit welcher Wahrscheinlichkeit die jeweilige Stellgröße gewählt wird. Gegebenenfalls kann daraus die Anzahl an Stichproben für stützstellenbasierte Regler abgeleitet werden oder umgekehrt bestimmt die Stützstellenanzahl die Wahrscheinlichkeit, in die Menge $\mathbb{O}$ zu gelangen und dort zu bleiben. In Kombination mit dem Konzept der P-praktischen Stabilität aus der Regelungstechnik (vgl. [3]) könnte daraus die Stabilitätswahrscheinlichkeit bestimmt werden. Dabei gilt, je größer $P \subseteq \mathbb{O}$ ist, desto wahrscheinlicher ist die Stabilität. Die Zuverlässigkeit der Stabilität sowie die Größe von P können anschließend mit der Zuverlässigkeitsanalyse (siehe Abschnitt C.4) berechnet werden.

Eidesstattliche Versicherung

Hiermit versichere ich, Alexander Puzicha, dass die Dissertation von mir selbstständig angefertigt wurde und alle von mir genutzten Hilfsmittel angegeben wurden. Ich versichere, dass alle in Anspruch genommenen Quellen und Hilfen in der Dissertation vermerkt wurden.

Dortmund, 29.08.2023 Unterschrift: Alexander Puzicha, M. Sc.