Benchmarking von Verhaltens-Algorithmen für die Applikation in Motorsport-Szenarien

Vom Fachbereich Maschinenbau an der
Technischen Universität Darmstadt
zur Erlangung des Grades eines
Doktor-Ingenieurs (Dr.-Ing.)
genehmigte

Dissertation

vorgelegt von

Marco Heinz Sippel, M. Sc.

aus Berlin

Berichterstatter:	Prof. Dr. rer. nat. Hermann Winner
Mitberichterstatter:	Prof. Dr. Ing. Jürgen Adamy

Tag der Einreichung:	17.08.2021
Tag der mündlichen Prüfung:	13.01.2022

Darmstadt 2021

D 17

Vorwort

Diese Dissertation wurde während meiner Zeit am Fachgebiet Fahrzeugtechnik von Juli 2018 bis Juli 2021. Meine Arbeit begann mit der Arbeit im BMWi-Projekt KOLLERAM, das mich zwei Jahre lang begleitete. Durch meine anschließende Arbeit im V&V-Methoden-Projekt konnte ich die vorliegende Arbeit abschließen.

Mein besonderer Dank gilt meinem Doktorvater Prof. Dr. rer. nat. Hermann Winner für die tolle Betreuung dieser Arbeit in zahlreichen Rücksprachen, für die konstruktiven Gespräche, das ehrliche Feedback, die guten Anregungen und sein immer offenes Ohr für die fachlichen und anderen Probleme, die die Verfassung dieser Arbeit mit sich brachte.

Ebenfalls danke ich Prof. Dr.-Ing. Jürgen Adamy für die Übernahme des Korreferats für diese Arbeit.

Meinen Kollegen bei FZD danke ich besonders für das Interesse im Rahmen der Doktorandenseminare und darüber hinaus. Ganz besonders in der doch schwierigen Corona-Zeit konnte man hier stets darauf zählen, jemanden zum Quatschen in dem ein oder anderen digitalen Sozialraum zu finden.

Ein besonderes Dankeschön geht an Philip Felizarta, der mich mit vielen hilfreichen Erklärungen zu den Methoden der Spieltheorie unterstützt hat und der insbesondere in der Endphase der Arbeit mein zuverlässiger Sparringspartner für Ideen rund um die Spieltheorie war.

Ebenfalls danke ich meinem Freund und Kollegen Philipp Rosenberger dafür, dass er diese Arbeit im Rahmen eines sonntäglichen Weißwurstfrühstücks ermöglicht und mich wieder auf den Weg zu FZD geführt hat.

Ebenso danke ich Maren Henzel, die durch die Beantragung des KOLLERAM-Projekts, die meine Arbeit für FZD erst ermöglicht hat.

Natürlich möchte ich meiner Familie für die immerwährende, jahrelange moralische und finanzielle Unterstützung und Hilfe danken. Ohne euch wäre diese Arbeit sicherlich nicht möglich gewesen!

Zuletzt danke ich meiner Frau Helena, die mich seit über einem Jahrzehnt unterstützt und mich für die Zeit ertragen hat. Durch dich habe ich gelernt, was die wichtigen Dinge im Leben sind.

München im August 2021.

Inhaltsverzeichnis

Formelzeichen und Indizes

Lateinische Buchstaben:

Zeichen	Einheit	Bezeichnung
$\mathscr{a}$	1	Index Referenzobjekt
a	$\frac{\mathrm{m}}{\mathrm{s}^2}$	Beschleunigung
$\mathscr{b}$	1	Index Parameterisierung
$\boldsymbol{b}$	$-$	Offset-Vektor
c_w	1	Luftwiderstandsbeiwert
$\mathscr{d}$	1	Index Strecke
$\mathfrak{d}t$	s	Länge eines Zeitschrittes
$\boldsymbol{e}$	$-$	Normierter Richtungsvektor
h	m	Abstand Kreismittelpunkt zu Sekante (Apothema)
i	1	Index Beschleunigungs-Randbedingung
j	1	Index Streckensegment
$\mathscr{j}$	1	Index Streckensegment in Abhängigkeit der letzten Lösung
$\mathscr{k}$	1	Höhe Dimension mElo-Verfahren
k_{ac}	1	Anzahl Beschleunigungs-Randbedingungen
k_n	1	Anzahl berechneter Zeitschritte pro Iteration
l	m	Länge
m	kg	Fahrzeugmasse
n	1	Index Zeitschritt
$\boldsymbol{n}$	1	Anzahl
p	1	Häufigkeit
$\boldsymbol{p}$	$-$	Payoff-Matrix
r	1	Rang eines Spielers
$\mathfrak{s}$	1	Slack-Variable
Δs	m	Länge Streckensegment
t	s	Zeitpunkt
u	1	Aktion des neuronalen Netzes
v	$\frac{\mathrm{m}}{\mathrm{s}}$	Geschwindigkeit
w	m	Breite
$\boldsymbol{x}_{l\|u,n}$	$-$	Obere und untere Grenzen zu Zeitpunkt n
$\boldsymbol{x}_n$	$-$	Zustand zum Zeitpunkt n
$\mathscr{A}$	1	Anzahl Referenzobjekte
A	m^2	Projizierte Stirnfläche
$\boldsymbol{A}$	$-$	Zustands-Matrix
$\mathscr{B}$	1	Anzahl Parametrisierungen
$\boldsymbol{B}$	$-$	Input-Matrix
$\boldsymbol{\mathscr{C}}$	$-$	Interaktions-Matrix für mElo-Verfahren
$\boldsymbol{C}$	$-$	Matrix mit Randbedingungen

Zeichen	Einheit	Bezeichnung
$\mathscr{D}$	1	Anzahl Strecken
D	$\frac{\mathrm{m}}{\mathrm{s}^2}$	Verzögerung (deceleration)
J	1	Anzahl Streckensegmente
$\mathscr{J}$	$\frac{\mathrm{m}}{\mathrm{s}^3}$	Ruck (jerk)
$\mathbb{K}$	$-$	Menge fehlender Matrix-Einträge
$\mathscr{M}$	1	Anzahl Aufgaben
$\boldsymbol{M}$	$-$	Drehmatrix
$\mathscr{N}$	1	Anzahl Spieler
P	W	Leistung (power)

Griechische Buchstaben:

Zeichen	Einheit	Bezeichnung
λ	1	Reduktionsfunktion
μ	1	Reibkoeffizient zwischen Reifen und Fahrbahn
ψ_c	rad	Kurswinkel
ψ_{tr}	rad	Winkel der Tangente an ein Streckensegment
τ	s	Zeitspanne
ξ	1	Kostenterm des Optimierungsproblems

Indizes:

Zeichen	Bezeichnung
acc	Beschleunigung (acceleration)
Æ	Aerodynamik
art	Künstlich (artificial)
beam	Strahlen des virtuellen Abstandssensors
co	Konstant (constant)
curv	Krümmung (curvature)
drag	Luftwiderstand (drag)
est	Geschätzt (estimated)
fo	Folgendes Fahrzeug
lat	Lateral
loss	Verlust
max	Maximal
min	Minimal

Zeichen	Bezeichnung
OT	Überholen (overtake)
PD	Nebeneinanderfahren (parallel driving)
po	Potenz
ref	Referenz
red	Reduziert
RS	Beschränkter Raum (restricted space)
SC	Räumliche Randbedingung (spatial constraint)
steer	Lenkung (steering)
SZ	Sichere Zone
tr	Strecke (track)
veh	Fahrzeug (vehicle)
width	Breite (width)
x	In x-Richtung
y	In y-Richtung

Abkürzungsverzeichnis

CDF	engl. Cumulative Distribution Function
CL	engl. Convolutional Layer
GP	Grand-Prix
KI	Künstliche Intelligenz
KPI	engl. Key-Performance Indicator
LCADM	engl. Linear Constraints with Advanced Decision Making
LSTM	engl. Long Short Term Memory
maxent NG	Nash-Gleichgewicht maximaler Entropie
MDP	engl. Markov Decision Process
mElo	multidimensionales Elo
ML	Maschinelles Lernen
MPC	engl. Model Predictive Controller
NN	Neuronales Netzwerk
PPO	engl. Proximal Policy Optimization
ReLU	engl. Rectified Linear Unit
RL	engl. Reinforcement Learning
RRT	engl. Rapidly exploring Random-Tree
SSP	Schere-Stein-Papier
SSPFW	Schere-Stein-Papier-Feuer-Wasser
TORCS	The Open Racing Car Simulator
TTC	engl. Time-to-Collision
VT	Verkehrsteilnehmer

Abbildungsverzeichnis

Tabellenverzeichnis

Kurzzusammenfassung

Im Zuge der Entwicklung des autonomen Fahrens existieren Bemühungen, den Motorsport als Technologie-Treiber und Showcase in diese Entwicklung mit einzubinden. Die Herausforderungen, die der Motorsport für die autonome Fahrt präsentiert, sind vielfältig und beinhalten neben der Wahrnehmung der Umwelt bei hohen Geschwindigkeiten auch die Regelung des Fahrzeugs und die Trajektorienplanung. Während die Trajektorienplanung für ein einzelnes Fahrzeug eine von vielen Autoren untersuchte Aufgabe ist, existieren für die Verhaltensplanung der Interaktion von mehreren Fahrzeugen bislang nur wenige Ansätze. Diese Ansätze basieren häufig auf unterschiedlichen Prinzipien und Fahrzeugmodellen. Somit ist die Evaluation häufig zwar nachvollziehbar, aber nicht auf andere Methoden extrapolierbar. Diese Arbeit startet daher mit der initialen Frage, wie und unter welchen Bedingungen möglich ist, verschiedene Algorithmen, die Verhalten für autonome Rennfahrzeugen erzeugen, miteinander zu vergleichen.

Die Analyse des Stands der Forschung offenbart, dass bisherige Ansätze für den Vergleich bislang häufig auf Reward-Funktionen aus dem Bereich des maschinellen Lernens zurückgreifen oder Metriken für die Evaluation nutzen, deren Relevanz für das betrachtete Szenario nicht genauer untersucht wird. Bestehende Ranking-Verfahren bieten die Möglichkeit, große Zahlen an menschlichen Spielern in eine Rangfolge zu bringen. Für Computeralgorithmen existieren dagegen aufgrund von Überlegenheiten nach dem Schere-Stein-Papier-Prinzip unterschiedlicher Strategien nur Methoden für kleine Anzahlen an Spielern.

Auf Basis der Analyse von Benchmarkings und den im Motorsport verfolgten Zielen werden die Randbedingungen für ein Szenario anhand der Szenario-Definition von Ulbricht *et al.*[1] mit Bezug auf eine qualitative Analyse und eine quantitative Einordnung diskutiert. Danach wird eine Methode vorgestellt, die genutzt werden kann, um auch eine Vielzahl an Algorithmen in einer Rangfolge anzuordnen. Eine vorgeschlagene Methode, die versucht, einzelne Testergebnisse zu prognostizieren und die Ergebnisse des Rankings in einem allgemein definierten Szenario offenbaren die Schwachstellen der untersuchten Verhaltensalgorithmen. Die Erstellung einer Rangfolge ist danach nur unter der Bedingung möglich, dass ein konkretes Test-Szenario definiert ist. Ein allgemeingültiger direkter Vergleich ist, aufgrund mangelnder Vergleichbarkeit für breite Parametervariationen, nicht möglich.

Diese Arbeit präsentiert ein umfassendes Konzept, um Verhaltensalgorithmen zu analysieren und weiterzuentwickeln. Das vorgestellte Ranking-Verfahren erlaubt außerdem

[1] Ulbrich, S. et al.: Begriffsdefinitionen für das automatisierte Fahren (2015).

die Bildung einer Rangfolge bei Voraussetzung eines im Vorfeld definierten Testszenarios. Für die Evaluierung des Bewertungsverfahrens wird ein Trajektorienplaner entwickelt und umgesetzt. Bedingt durch die kurzen Berechnungszeiten wird die Kombination des modellprädiktiven Planers mit einem neuronalen Netz ermöglicht. Auf diese Weise eröffnet sich das Feld der Verhaltensoptimierung durch maschinelles Lernen, das durch besonders hohe Anzahl zu berechnender Zeitschritte mit anderen Ansätzen der Trajektorienberechnung keine praktikable Umsetzung bietet. Auf diese Weise wird ein Agent implementiert, der die Aufgabe erfüllt, ein anderes Fahrzeug am Überholen zu hindern und auch ein Überholen zu ermöglichen. Der vorgestellte deterministische Ansatz für das Problem aggressive Überholmanöver auszuführen, zeigt in der ersten Analyse eine vielversprechende Performance. Allerdings treten für den Ansatz bei der Fahrt auf realen Streckenlayouts noch eine hohe Anzahl an Kollisionen auf.

Abstract

As autonomous driving develops, efforts exist to incorporate motorsports into this development as a technology driver and showcase. The challenges presented by motorsports for autonomous driving are many and include vehicle control and trajectory planning in addition to perception of the environment at high speeds. While trajectory planning for a single vehicle is a task studied by many authors, few approaches exist to date for behavioral planning of the interaction of multiple vehicles. These approaches are often based on different principles and vehicle models. Thus, the evaluation is often comprehensible but not extrapolable to other methods. This work therefore starts with the initial question of how and under which conditions it is possible to compare different algorithms that generate behavior for autonomous racing vehicles.

The analysis of the state of the art reveals that previous approaches for comparison so far often rely on reward functions from the field of machine learning or use metrics for evaluation whose relevance for the scenario under consideration is not investigated in more detail. Existing ranking methods offer the possibility to rank large numbers of human players. For computer algorithms, on the other hand, only methods for small numbers of players exist due to superiority based on the rock-paper-scissors principle of different strategies.

Based on the analysis of benchmarks and the goals pursued in motorsports, the boundary conditions for a scenario are discussed using the scenario definition of Ulbricht *et al.*[2] with reference to a qualitative analysis and a quantitative ranking. A method is then presented that can be used to rank even a large number of algorithms. A proposed method that attempts to predict individual test results and the results of ranking in a generally defined scenario reveal the weaknesses of the behavioral algorithms studied. According to this, ranking is possible only under the condition that a concrete test scenario is defined. A generally valid direct comparison is not possible due to the lack of comparability for broad parameter variations.

This work presents a comprehensive concept to analyze and further develop behavioral algorithms. The presented ranking procedure also allows the formation of a ranking given a predefined test scenario. For the evaluation of the ranking procedure a trajectory planner is developed and implemented. Due to the short computation times, the combination of the model predictive planner with a neural network is made possible. In this way, the field of behavior optimization by machine learning opens up, which does not offer a feasible implementation with other approaches of trajectory computation due to a particularly high number of time steps to be computed. In this way, an agent

[2] Ulbrich, S. et al.: Begriffsdefinitionen für das automatisierte Fahren (2015).

is implemented that performs the task of preventing another vehicle from overtaking and also enabling overtaking. The presented deterministic approach to the problem of performing aggressive overtaking maneuvers shows promising performance in the initial analysis. However, a high number of collisions still occur for the approach when driving on real track layouts.

1. Einleitung

1.1. Motivation

Das automatisierte Fahren stellt eine der größten Herausforderungen in der heutigen Entwicklung der Kraftfahrzeuge von morgen dar. Im Zuge der Automatisierung von Kraftfahrzeugen gibt es zunehmend mehr Bemühungen, auch Rennwagen in diese Entwicklung mit einzubeziehen. Dabei sind nicht nur die Wahrnehmung der Umwelt und Regelung der Fahrzeuge von hoher Komplexität. Die Planung der Trajektorien und der Interaktion von Fahrzeugen auf der Strecke ist für den automatisierten Rennsport ebenfalls eine Herausforderung. Um das Hauptziel im Rennsport, die Ziellinie als Erster zu überqueren, zu erreichen, sind einerseits das Überholen von anderen Fahrzeugen und andererseits von diesen nicht überholt zu werden, die vordringlichen Ziele in der Trajektorienplanung. Es existiert im Rennsport zwar eine Ideallinie, die ein in vielen Fällen gutes Ergebnis für die Fahraufgabe liefert, das Fahrzeug alleine auf der Strecke zu bewegen. Ein Überholen ist aber nur durch eine abweichende Linienwahl möglich, die gemäß der Eigenschaften der Ideallinie zunächst ein langsameres Vorankommen erwarten lässt. Es ist darüber hinaus damit zu rechnen, dass ein vorausfahrendes Fahrzeug versucht, das folgende Fahrzeug durch eine Linienwahl abweichend von der Ideallinie aktiv am Überholen zu hindern.

Aktuelle Forschungsarbeiten beinhalten bereits einige Ansätze für die Umsetzung von Trajektorienplanern, auch unter der Einbeziehung von anderen Verkehrsteilnehmern in Rennstreckenszenarios. Da die Ansätze häufig auf Algorithmen verschiedener Grundansätze und Funktionsprinzipien basieren, ist nur in seltenen Fällen möglich, Algorithmen und ihre Performance innerhalb eines Szenarios miteinander zu vergleichen. Es existieren bereits erste Ansätze für Bewertungsmaßstäbe, um einen Algorithmus bei der alleinigen Fahrt auf der Rennstrecke zu evaluieren. Diese beziehen sich insbesondere auf die Regelung des Fahrzeugs und stützen sich auf die Analyse des Fahrstils von menschlichen Rennfahrern.[3] Ein umfassendes Konzept, das eine Beurteilung des Zusammenspiels mit anderen Rennteilnehmern ermöglicht, existiert allerdings nicht und ist auch laut einer Studie von Doubek *et al*.[4] nur eine zweitrangige Priorität bisheriger Forschung. Die Thematik gewinnt im Kontext des automatisierten Fahrens an Bedeutung [5], wenn auch die Interaktion von menschlichen Fahrern mit automatisierten Fahrzeugen im Vordergrund steht. Damit startet diese Arbeit mit der initialen Forschungsfrage:

[3] Vgl. Hermansdorfer, L. et al.: Benchmarking of a software stack for autonomous racing (2020).

[4] Doubek, F. et al.: What makes a good driver on public roads and race tracks? (2020).

[5] Vgl. Grasso, G. M. et al.: A flexible environment for autonomous driving interaction testing (2020); vgl. Zhou, M. et al.: Scalable Multi-Agent RL Training School for Autonomous Driving (2020).

Unter welchen Bedingungen ist es möglich, für eine Vielzahl von Algorithmen zur Verhaltens-Planung von autonomen Rennfahrzeugen Vergleichbarkeit innerhalb von Test-Szenarien herzustellen?

Diese Vergleichbarkeit ist nicht nur für die Analyse der Algorithmen selbst wichtig, da bislang keine Randbedingungen bekannt sind, die einen Maßstab für den Vergleich von Verhaltens-Algorithmen bilden. Die Arbeit von Stahl *et al.*[6] bietet an dieser Stelle einen ersten Open-Source-Ansatz für die Erstellung von Szenarien mit mehreren Fahrzeugen. Während im normalen Straßenverkehr das Ziel verfolgt wird, ohne Unfall von einem Ort A an einen anderen Ort B zu gelangen, ist im Motorsport das Ziel, eine bestimmte Strecke so schnell wie möglich im Vergleich zu anderen Fahrern zu absolvieren. Der Motorsport bietet in der Realität also die Möglichkeit für eine Vielzahl von Fahrern, sich im Wettkampf zu messen. Der Hauptunterschied zwischen normalem Straßenverkehr und einem Motorsportszenario ist damit der Wettbewerbsgedanke und der direkte Vergleich zwischen den Teilnehmern innerhalb eines Wettbewerbs. Um die Möglichkeit für einen direkten Vergleich für den autonomen Motorsport ebenfalls zu bieten, ist neben der Analyse und Bewertung von Einzeleigenschaften auch eine Möglichkeit notwendig, mehrere Algorithmen in eine Rangfolge zu bringen und analog zu Ranglisten mit menschlichen Spielern im Schach oder Go einen Maßstab für die Leistung verschiedener Algorithmen zu schaffen. Die zweite Frage, die im Laufe dieser Arbeit beantwortet wird, ist daher:

Wie und unter welchen Bedingungen ist die Bildung einer Rangfolge von Algorithmen zur Verhaltens-Planung von autonomen Rennfahrzeugen möglich?

1.2. Methodisches Vorgehen der Arbeit

Die Methodik, die in dieser Arbeit verfolgt wird, ist in Abbildung 1-1 zusammengefasst. Der Fragestellung folgend, wie es möglich ist, für eine Vielzahl von verschiedenen Algorithmen für die Verhaltensplanung im Kontext des autonomen Motorsports Vergleichbarkeit herzustellen, analysiert Kapitel 3 den Stand der Forschung und geht auf bereits bestehende Ansätze für die Evaluierung von Verhaltensalgorithmen ein. Die Analyse spieltheoretischer Ansätze zeigt auf, welche Möglichkeiten bestehen, um verschiedene Algorithmen in eine Rangfolge zu bringen und welche Randbedingungen für einen solchen Vergleich zu beachten sind.

Daran anschließend gibt Kapitel 4 einen Einblick in die Methode des Benchmarkings und zeigt Anforderungen auf, die an einen Benchmarking-Prozess gestellt werden, um eine möglichst hohe Aussagekraft zu besitzen. Die Analyse des Motorsports als solcher und die verschiedenen Betrachtungsebenen im Bezug auf die jeweilige Zielsetzung

[6] Stahl, T. et al.: An Open-Source Scenario Architect for Autonomous Vehicles (2020).

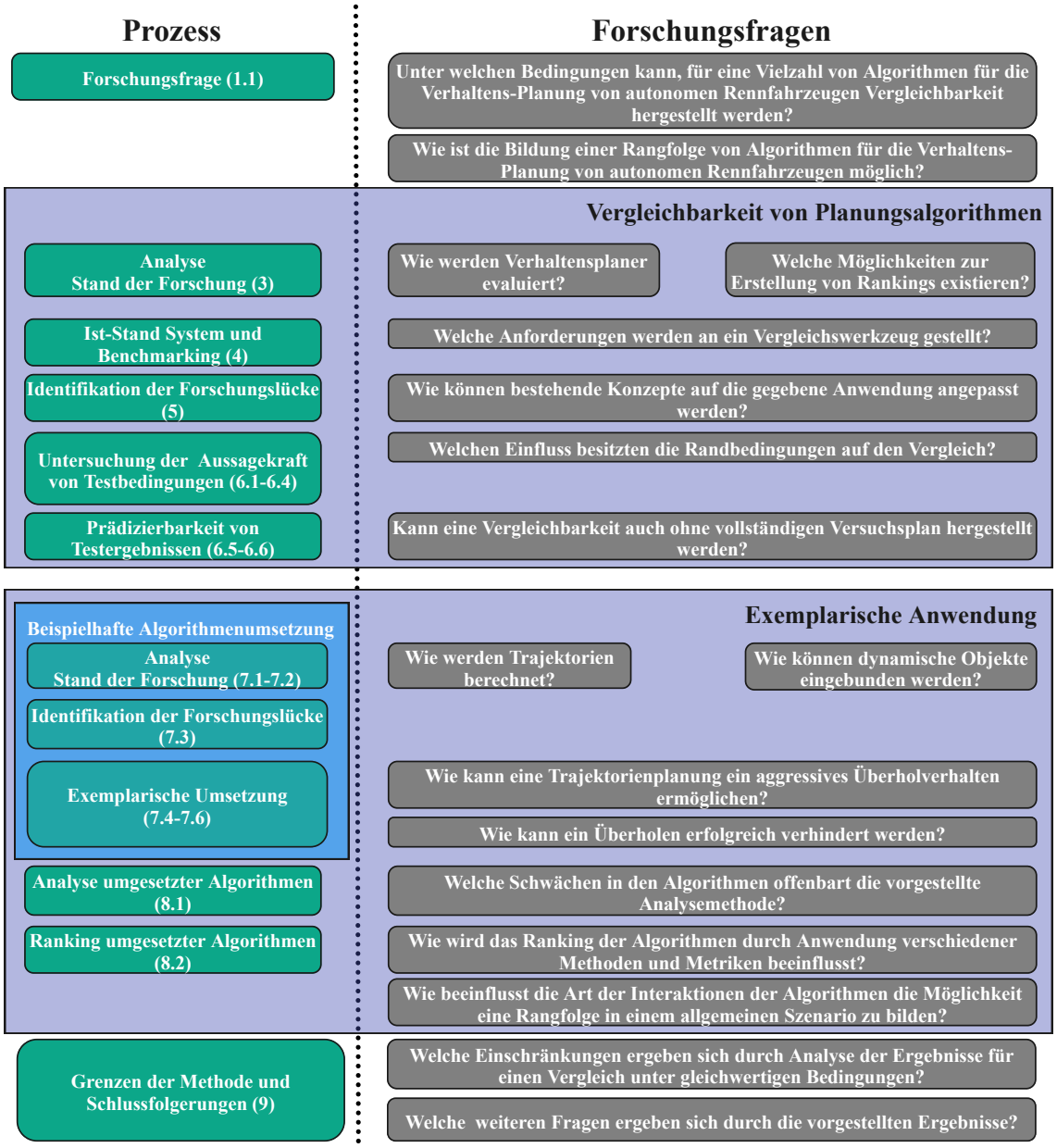


Abbildung 1-1.: Methodik der Arbeit

geben weiter Aufschluss über die durchführbaren Tests und deren Bedeutung in Bezug auf das Gesamtsystem.

Kapitel 5 formuliert unter Einbezug von Kapitel 3 und Kapitel 4 Forschungsfragen bezüglich der Vergleichbarkeit von Testszenarien. Weiter wird an dieser Stelle auf die Bildung eines Rankings von mehreren Algorithmen eingegangen und inwieweit Tests für eine Vielzahl an Algorithmen anwendbar sind, insbesondere in Bezug auf mögliche Inkompatibilitäten zwischen Algorithmen bzw. deren Modellierungen. Weiter werden die Implikationen durch bestehende Ranking-Verfahren und den Einfluss der Fahrzeug-Interaktion diskutiert.

Auf die umfassende Diskussion der Randbedingungen in Kapitel 6 folgt die Vorstellung

einer Methode, die darauf abzielt, im betrachteten Kontext mit Lücken im Testplan umzugehen. Diese Lücken im Testplan sind bedingt durch mangelnde Verfügbarkeit oder inkompatible Modellvorstellungen. Diese Methode wird anhand eines einfachen, abstrakten Beispiels evaluiert, die Implikationen für die Test-Gestaltung eines Vergleichswerkzeugs herausgearbeitet und das daraus resultierende Rankingverfahren vorgestellt.

Der beispielhaften Anwendung des vorgestellten Analysetools geht die Implementierung von mehreren Algorithmen in Kapitel 7 voraus. Das Kapitel analysiert dafür zunächst den Stand der Forschung in den Bereichen der Trajektorienberechnung für den Motorsportkontext unter Berücksichtigung anderer dynamischer Objekte auf der Strecke. Die Analyse der Lücken im Stand der Forschung bildet die Basis für die entwickelten Vergleichsalgorithmen, die das Ziel verfolgen, (a) ein Überholen von anderen Fahrzeugen erfolgreich zu verhindern und (b) ein aggressives Überholen durch einen Algorithmus zu ermöglichen.

Die in dieser Arbeit entwickelte Vergleichsmethode wird in Kapitel 8 auf die vorgestellten Algorithmen angewendet. Dabei steht zum einen die systematische Analyse der Schwachstellen im Vordergrund, zum anderen werden mögliche Ranking-Ansätze in Kombination mit verschiedenen Metriken verglichen und die Ergebnisse diskutiert.

Kapitel 9 fasst die Ergebnisse der Arbeit zusammen und geht auf die gewonnenen Erkenntnisse ein. Ebenso werden in dieser Arbeit nicht berücksichtigte Punkte beleuchtet, die Limitierungen der vorgestellten Methode aufgezeigt und potenzielle Themen für nachfolgende Forschung vorgestellt.

1.3. Hinweis zu Personenbezeichnungen

Für eine gute Lesbarkeit des Textes wird in dieser Arbeit für alle Personenbezeichnungen das generische Maskulinum verwendet. Die Begriffe werden dabei als abstrakte Begriffe verwendet und bezeichnen keine tatsächlichen Personen.

2. Begriffsdefinitionen

Im Folgenden werden die wichtigsten Begriffe erläutert, die die Grundlage dieser Arbeit bilden.

2.1. Szenario, Szene und Szenerie

Für die Beschreibung der Tests, die die Grundlage für diese Arbeit bilden, wird die verbreitete Ontologie nach Ulbrich *et al.*[7a] verwendet. Dabei werden folgende Begriffe, wie in Abbildung 2-1 zu sehen, definiert.

- **Szenerie**: Die physische Umgebung in einem Test wird als Szenerie bezeichnet. Darunter zählen alle festen Bestandteile der Umgebung. Für ein Rennstreckenszenario fällt darunter die Rennstrecke mit den zugehörigen Streckenbedingungen. Für die weitere Betrachtung spielen insbesondere auch die Kurven und deren Reihenfolge eine Rolle. Aus diesem Grund wird für die Szenerie ebenfalls der Begriff **Layout** verwendet.
- **Szene**: Die Erweiterung der Szenerie ist die Szene, die neben der Szenerie auch das Ego-Fahrzeug und alle anderen dynamischen Objekte mit einschließt.
- **Szenario**: Die oberste betrachtete Ebene ist das Szenario. Es beinhaltet neben einer Szene, die die physische Welt beschreibt, zusätzlich alle Aktionen und Ereignisse, die während eines Tests geschehen. Weiter beinhaltet das Szenario eine Zielvorgabe für das Ego-Fahrzeug und die dynamischen Elemente sowie weitere Werte, die die Entwicklung der Objekte beschreiben. Ein Szenario beinhaltet als Erweiterung der Szene eine zeitliche Abfolge.

2.2. Verhalten und Interaktion

Als **Verhalten** werden in der Verhaltensbiologie alle Änderungen eines Organismus bezeichnet, die von außen wahrnehmbar sind und zu einer Veränderung des Organismus führen. Dazu gehören bei Lebewesen nicht nur Bewegungen, sondern auch Laute, Gerüche oder sichtbare Veränderungen der Haut.[8] Im Kontext des autonomen Fahrens definieren Nolte *et al.*[9] das Verhalten als eine Aneinanderreihung von Zuständen. Bei den Zuständen wird zusätzlich die Unterscheidung nach äußeren und inneren Zuständen getroffen. Diese Definition ist konsistent mit anderen Quellen wie

[7] Ulbrich, S. et al.: Begriffsdefinitionen für das automatisierte Fahren (2015).

[8] Minton, E. A. et al.: Belief systems, religion, and behavioral economics (2014).

[9] Nolte, M. et al.: Towards a skill- and ability-based development process for road vehicles (2017).

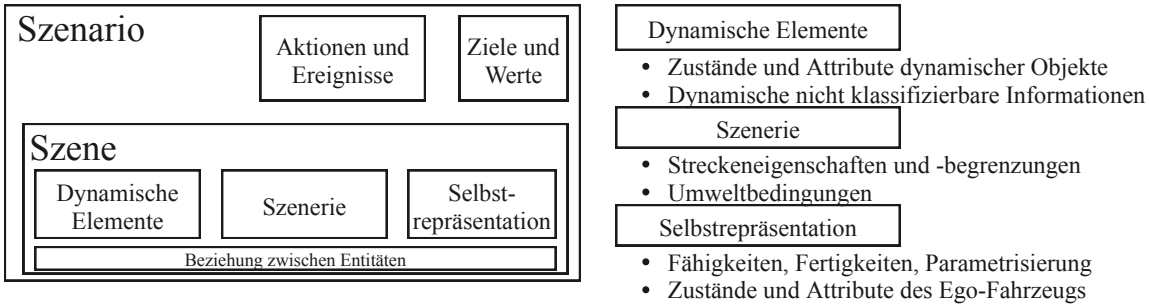


Abbildung 2-1.: Definition der Begriffe Szene, Szenario, Szenerie angelehnt an Ulbrich[7b]

der Ontologie für automatisierte Fahrzeuge von Czarnecki[10] und wird auch in dieser Arbeit übernommen.

Übertragen auf ein autonomes Fahrzeug umfasst das Verhalten zum einen die Bewegung, die das Fahrzeug ausführt, aber auch alle Kommunikationssignale, die es aussendet und die von anderen Verkehrsteilnehmern wahrnehmbar sind. Im Rahmen dieser Arbeit wird das Verhalten und dessen Bewertung im Kontext des automatisierten Fahrens auf die Bewegung des Fahrzeugs beschränkt. Verhalten bezeichnet im Kontext dieser Arbeit jede Form der Bewegung eines Fahrzeugs. Führt diese Bewegung bei einem anderen Verkehrsteilnehmer zu einer Änderung dessen Verhaltens, so wird dieser Vorgang als **Interaktion** verstanden.

2.3. Maschinelles Lernen

Da für die Generierung von Verhalten zunehmend Algorithmen des Maschinellen Lernens (ML) zum Einsatz kommen[11], spielen diese auch in dieser Arbeit eine entscheidende Rolle für die Generierung der Fahrzeuginteraktion. Folgende Begriffe sind daher gegeneinander abzugrenzen.

Als **Agent** bezeichnet sich das Neuronales Netzwerk (NN), das durch den Lernalgorithmus trainiert wird und das Verhalten erzeugt. **Reinforcement Learning** (RL) beschreibt eine Form des ML, bei dem ein Agent eine Umgebung selbstständig erkundet. Die Aktionen des Agenten werden zusammen mit dem Zustand der Umgebung gespeichert und bewertet. Diese Bewertung wird als **Reward** bezeichnet. Der **Lernalgorithmus** ist dabei der Algorithmus, der das NN auf Basis von Rewards, der Aktion des Agenten und dem zugehörigen Zustand der Umgebung optimiert.[12] Im Gegensatz zum Supervised Learning, bei dem ein NN angelernt wird, bspw. um die Klassifizierung

[10] Czarnecki, K.: Operational World Model Ontology for Automated Driving Systems - Part 2 (2018).

[11] Vgl. Mirchevska, B. et al.: High-level Decision Making for Safe Lane Changing using RL (2018); vgl. Zhou, M. et al.: Scalable Multi-Agent RL Training School for Autonomous Driving (2020); vgl. Chen, J. et al.: Interpretable End-to-end Urban Autonomous Driving with Latent Deep RL (2020); vgl. Loiacono, D. et al.: Learning to overtake in TORCS using simple reinforcement learning (2010).

[12] Dong, H. et al.: Deep RL: Fundamentals, Research and Applications (2020), S. 48ff.

eines bestimmten Datensatzes zu reproduzieren, existieren beim RL keine korrekten oder falschen Aktionen. Die Aktionen, die der Agent trifft, unterscheiden sich lediglich durch den Reward, der durch die definierte Umgebung für eine bestimmte Aktion gegeben wird.[13] Damit verbunden ist auch, dass, wenn ein Algorithmus innerhalb einer Umgebung trainiert wird, das erzielte Ergebnis nur unter der Bedingung eines idealen Lernprozesses und einer passend definierten Umgebung und Reward-Funktion ein globales Optimum erreicht.

Für die Beschreibung von ML-Verfahren werden sogenannte **Hyperparameter** verwendet. Diese beschreiben die für ein Lernverfahren oder NN zu wählenden Parameter wie die Anzahl von Neuronen in einem NN oder die Lernrate für die Optimierung. Der Begriff ist aber in dieser Arbeit nicht auf den Bereich des ML beschränkt. Er beschreibt vielmehr allgemein einen Parameter für einen Prozess, ein Verfahren oder einen Test, der durch den Anwender festzulegen ist.

Als Abgrenzung zum Reward, der durch die Reward-Funktion definiert ist, beschreibt die **Kostenfunktion** die Funktion, die als Grundlage für einen Optimierungsalgorithmus dient.

2.4. Spieltheorie

In der Spieltheorie wird grundsätzlich unterschieden zwischen **Nullsummenspiel**en und Nicht-Nullsummenspielen. Dafür wird zunächst angenommen, dass ein Spieler n gegen einen anderen Spieler m in einem Spiel antritt. Ein Nullsummenspiel bezeichnet ein Spiel, in der der Gewinn, den ein Teilnehmer erhält, dem Verlust eines anderen Teilnehmers gleicht.[14] Bekannte Beispiele für Nullsummenspiele sind Brettspiele wie Schach oder Go, bei denen, wenn ein Spieler gewinnt, der andere verliert oder ein unentschiedener Zustand erreicht wird. Der Ausgang eines Spiels wird an dieser Stelle mit Auszahlung oder **Payoff** p_{nm} bezeichnet.

2.5. Performance

Der Begriff der **Performance** beschreibt allgemein wie "gut" eine bestimmte Aufgabe von einer Entität, relativ zu anderen Entitäten ausgeführt wird. Zhou *et al*.[15] fassen darunter bspw. die Häufigkeit von Kollisionen, die Häufigkeit von erfolgreichen Szenarien und die Robustheit eines Algorithmus gegenüber Variationen innerhalb eines definierten Szenarios zusammen. Sander[16] dagegen, der sich mit dem Training von NN in einem Rennszenario beschäftigt, bezieht sich mit dem Begriff der Performance auf

[13] Stone, P.: Encyclopedia of Machine Learning: Reinforcement Learning (2010).

[14] Owen, G.: Spieltheorie (1968), S. 11.

[15] Zhou, M. et al.: Scalable Multi-Agent RL Training School for Autonomous Driving (2020).

[16] Sander, R.: Emergent Autonomous Racing Via Multi-Agent Proximal Policy Optimization (2020).

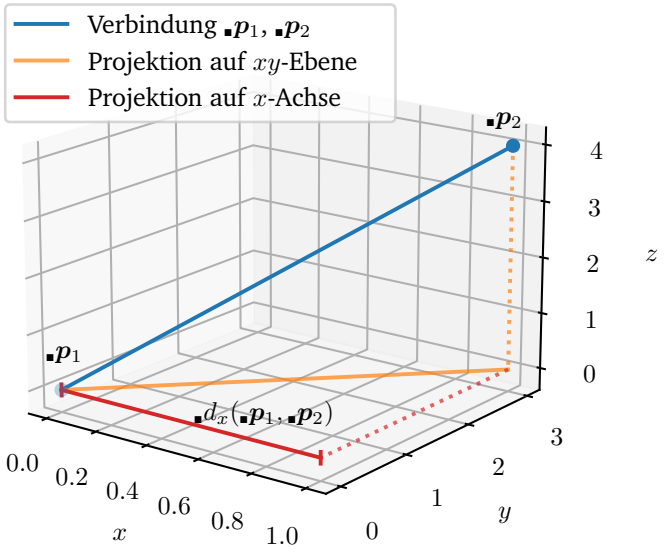


Abbildung 2-2.: Beispiel für die Verwendung der Notation für projizierte Abstände. Dargestellt ist die projizierte Distanz zwischen den Punkten ${}_{\blacksquare}\boldsymbol{p}_1$ und ${}_{\blacksquare}\boldsymbol{p}_2$ auf die x-Achse des ■-Koordinatensystems.

die Punktzahl, die ein NN während eines Szenarios erreicht. Ähnliche Verwendungen des Performance-Begriffs finden sich ebenfalls in anderen Studien mit thematischem Bezug zu ML.[17] Angelehnt an die Verwendung des Begriffs in der Literatur beschreibt die Performance in dieser Arbeit ebenfalls die Effektivität eines Verhaltensalgorithmus mit Bezug auf den Ausgang eines Szenarios. Faktoren, die dabei berücksichtigt werden, sind (a) die Häufigkeit, mit der ein Szenario erfolgreich abgeschlossen wird, (b) die Häufigkeit, mit der eine Kollision mit einem anderen Fahrzeug stattfindet und (c) die erreichte Geschwindigkeit im Vergleich mit einem Szenario, bei dem sich kein anderes Fahrzeug auf der Strecke befindet.

2.6. Verwendete Koordinatensysteme und Notation

Es werden drei Koordinatensysteme für die Modellierung verwendet. Deren Notation erfolgt als Index vor der jeweiligen Variable.

- ${}_{E}\square$ bezeichnet Werte in einem erdgebundenen Koordinatensystem, das die Basis für alle Gleichungen des Optimierungsproblems bildet.
- ${}_{\mathbb{N}}\square$ bezeichnet Werte in einem natürlichen Koordinatensystem, das auf der errechneten Bewegung des Fahrzeugs basiert.
- ${}_{F}\square$ bezeichnet Werte in einem Frenet-Koordinatensystem, das sich auf die Mittellinie der Strecke bezieht.

[17] Vgl. Likmeta, A. et al.: Combining RL with rule-based controllers for autonomous driving (2020); vgl. Niu, J. et al.: Two-Stage Safe Reinforcement Learning for High-Speed Autonomous Racing (2020).

Alle normalisierten Vektoren werden mit $\boldsymbol{e}$ bezeichnet und die Einheitsvektoren entlang der ▲-Achse im ●-Koordinatensystem mit ${}_{\bullet}\boldsymbol{e}_{\blacktriangle}$. Um die Notation für projizierte Distanzen abzukürzen, bezeichnet ${}_{\bullet}d_{\blacktriangle}({}_{\blacksquare}\boldsymbol{p}_1, {}_{\blacksquare}\boldsymbol{p}_2)$ eine projizierte Distanz zwischen dem Punkt ${}_{\blacksquare}\boldsymbol{p}_1$ und dem Punkt ${}_{\blacksquare}\boldsymbol{p}_2$ im ■-Koordinatensystem entlang der ▲-Achse des ●-Koordinatensystems. Ein Beispiel für die Verwendung der Notation ist in Abbildung 2-2 dargestellt.

3. Stand der Forschung

Grundlage der Untersuchungen dieser Arbeit bilden der aktuelle Stand der Forschung im Bereich der Evaluierung von Planungsalgorithmen in Abschnitt 3.1, um der initialen Frage dieser Arbeit nachzugehen, auf welche Weise möglich ist, Vergleichbarkeit zwischen verschiedenen Verhaltensalgorithmen für den autonomen Motorsport herzustellen. Im Anschluss daran wird durch die Analyse des Standes der Forschung auf dem Gebiet von Ranking-Systemen in Abschnitt 3.2 die Grundlage für die Beantwortung der zweiten initialen Frage dieser Arbeit geschaffen, auf welche Weise möglich ist, eine Vielzahl von Verhaltensalgorithmen im speziellen Szenario des autonomen Motorsports in eine Rangfolge zu bringen, um die relative Güte der Algorithmen zu evaluieren.

3.1. Verhaltensbewertung für Fahrzeuge

Die Bewertung des Verhaltens von technischen Systemen und insbesondere autonomen Fahrzeugen ist ein Thema, das bereits in einigen Forschungsarbeiten thematisiert wurde. Die Hauptfragestellung ist, wie möglich ist, die Interaktionen eines Systems mit seiner Umwelt und anderen dort befindlichen Systemen zu bewerten und zu quantifizieren. Von Interesse sind an dieser Stelle zum einen die verwendeten Metriken, aber auch Methoden für die Einordnung der Performance in Relation zu anderen Systemen. Im Folgenden werden einige repräsentative Ansätze kurz vorgestellt und abschließend bewertet.

3.1.1. Bestehende Ansätze

Huang *et al.*[18] stellen ein NN vor, das in einem Rennszenario in The Open Racing Car Simulator (TORCS)[19] sowohl Überholen als auch Verteidigen lernt. Sie evaluieren ihren Ansatz in einem ersten Schritt auf einer geraden Strecke und werten für verschiedene Agenten-Strecken-Kombinationen die Überholhäufigkeit aus. Dabei wird immer derselbe Gegner verwendet, der ein von TORCS übernommener Fahralgorithmus ist. Dieser ist in der Geschwindigkeit begrenzt, um eine hohe Zahl an Überholmanövern zu erzeugen. Für die Überholfähigkeit in Kurven argumentieren die Autoren, dass einzelne Szenarien keine große Aussagekraft besitzen, da in der Realität diese sich immer unterschiedlich entwickeln und lassen deshalb die Fahrzeuge viele Runden gegeneinander fahren. Als Auswertungsmetrik wird danach zum einen die Anzahl an

[18] Huang, H.-H. et al.: Learning overtaking and blocking skills in simulated car racing (2015).

[19] TORCS ist eine Open-Source-Rennsimulation mit offenen Schnittstellen mit dem Ziel, die Entwicklung von ML-basierten Ansätzen für die Fahrzeugführung voranzutreiben.
Vgl. Wymann, B.: Torcs FAQs (2012).

Überrundungen verwendet und zum anderen der TORCS-interne ermittelte Schaden am Fahrzeug.[20]

Das vorgestellte Benchmarking von Hermansdorfer *et al.*[21] bezieht sich weniger auf die Verhaltensebene des Fahrzeugs und mehr auf die darunter liegende Reglerebene. Für die Regelung schlagen die Autoren einige Key Performance Indikatoren (KPI) für die Bewertung des Fahrstils eines autonomen Rennfahrzeugs vor. Die Hauptkategorien der Bewertungskriterien stammen dabei von Segers[22] und Trzesniowski[23]:

- Performance (u. a. wie lange gibt der Algorithmus Vollgas?)
- Glattheit (Welchen Wert habe die höheren Ableitungen der Eingabesignale?)
- Ansprechverhalten (Wie schnell wird Bremsdruck aufgebaut?)
- Effizienz (z. B. Verhältnis longitudinale Beschleunigung zu Gaspedalstellung)
- Konstante Ergebnisse über verschiedene Kurven, Runden und Strecken

Die Autoren schlagen zusätzlich noch die Nutzung der Fahrzeugstabilität und die Linienwahl als KPI vor und vergleichen ihren Softwarestack anhand dieser Metriken mit einem professionellen Rennfahrer. Die Linienwahl wird hier allerdings nicht auf eine gute Eignung oder die erreichbare Rundenzeit analysiert, sondern sie wird genutzt, um zu ermitteln, ob der Fahrregler in der Lage ist, der vorgegebenen Trajektorie zu folgen. Die Metriken zeigen dabei deutlich die Schwächen des Algorithmus auf. Andere Fahrzeuge werden in der Studie nicht mit berücksichtigt. Zudem zeigt sich in der Studie, dass der Vergleich mit einem Menschen den Nachteil der schwierigen Datengewinnung hat. Aufgrund der hohen Kosten für die realen Tests ist nur möglich, einzelne Runden zu vergleichen, was zwar für einen Regelalgorithmus bereits Anhaltspunkte für Stärken und Schwächen liefert, aber mit steigender Komplexität der Szenarien nur eine geringe Aussagekraft besitzt, da nur unter hohem finanziellen Aufwand möglich wäre, eine große Zahl an Versuchen zu fahren.

Die Veröffentlichung von Zhou *et al.*[24] stellt ein Framework vor, dass die Interaktion zwischen Fahrzeugen im Straßenverkehr in den Vordergrund stellt. Dabei konzentrieren sich die Autoren auf die Verwendung von RL-Algorithmen. Das Framework implementiert eine Vielzahl verschiedener Szenarien des Straßenverkehrs. Ein wichtiger Aspekt des Frameworks ist aber nicht nur die Vielfalt der Szenarien, sondern

[20] Schaden bedeutet in diesem Zusammenhang ein akkumulierter Wert, der sich bei einer Kollision mit anderen Fahrzeugen oder dem Rand der Strecke erhöht und den Widerstand des Fahrzeugs erhöht. Vgl. Wymann, B.: Forenbeitrag zum Schadensmodell in TORCS (2012).

[21] Hermansdorfer, L. et al.: Benchmarking of a software stack for autonomous racing (2020).

[22] Segers, J.: Analysis techniques for racecar data acquisition (2014).

[23] Trzesniowski, M. et al.: Rennwagentechnik - Datenanalyse, Abstimmung (2017).

[24] Zhou, M. et al.: Scalable Multi-Agent RL Training School for Autonomous Driving (2020).

die Möglichkeit, Interaktionspartner verschiedenster Funktionsweisen in Form von anderen Agenten in der Simulation als Fahrzeuge fahren zu lassen. Das ermöglicht, dass die Algorithmen stark unterschiedliche Beobachtungen sammeln und ein Lernen daraus ermöglicht wird. Auf diese Weise ist möglich, dass ein Algorithmus auch generalisierbares Verhalten erlernt. Die Evaluation der Algorithmen lässt sich damit ebenfalls mit verschiedenen Interaktionspartnern durchführen. Es lässt sich bspw. aber auch die Generalisierbarkeit des Verhaltens von einem Agenten überprüfen. Für die Evaluation teilen sich die Kriterien in folgende drei Kategorien:

- Performance: Anzahl Kollisionen, erfolgreich absolvierte Szenarien, Generalisierbarkeit/Robustheit über Szenariovariationen
- Verhalten: Sicherheit, Agilität, Stabilität, Verhältnisse verschiedener Inputs, Determiniertheit des Verhaltens, Häufigkeit von Einschermanövern
- Spiel-Theorie: Kooperatives Verhalten, Kompetitives Verhalten

Die Auswertung der Verhaltens-Metriken für verschiedene Lern-Algorithmen zeigt eine steigende Zahl an Interaktionen und eine steigende Komplexität der Interaktionen mit zunehmender Komplexität der Szenarien. Die Autoren gehen an dieser Stelle nicht weiter auf die Gründe dieser Unterschiede ein.

Die Veröffentlichung von Christiano *et al*.[25] zeigt beispielhaft, wie in vielen Fällen bei der Bewertung von RL-Algorithmen und -Lernverfahren vorgegangen wird. Der dort vorgestellte Algorithmus lernt ohne eine zuvor definierte Kostenfunktion ein NN an. Dabei wird auf das Feedback eines Menschen zurückgegriffen. Die Kostenfunktion wird also ebenfalls optimiert, um das Feedback des Menschen so genau wie möglich abzubilden. Auf diese Weise ist nicht notwendig, die Kostenfunktion für komplexe Szenarien händisch zu formulieren und zu parametrieren. Die Autoren nutzen das Verfahren, um dem Agenten Verhaltensweisen beizubringen, für die bislang keine Gütefunktionen existieren. Durch die gewählte Netzarchitektur liefert die Methode nicht nur die Güte des momentan beobachteten Zustands, sie zeigt ebenfalls an, mit welcher Konfidenz das Netz den Zustand bewertet hat. Auf diese Weise ist es lediglich notwendig, die Zustände im fortschreitenden Verlauf von einem Menschen zu bewerten bzw. zu vergleichen, bei denen die Entscheidung des Netzes keine hohe Konfidenz besitzt. Die Güte des Verfahrens wird daran gemessen, wie hoch der erreichte Reward ist, den der Agent während einer Episode im Mittel sammelt. Dieser Wert wird über dem Lernprozess beobachtet und auf diese Weise sowohl die Performance des Agenten als auch des Lernverfahrens beurteilt.

Niu *et al*.[26] nutzen zwei Ereignisse zur Evaluation ihrer Algorithmen. Sie zählen, wie

[25] Christiano, P. et al.: Deep reinforcement learning from human preferences (2017).

[26] Niu, J. et al.: Two-Stage Safe Reinforcement Learning for High-Speed Autonomous Racing (2020).

häufig der gelernte Algorithmus bzw. der Vergleichsmaßstab das Szenario erfolgreich absolviert und wie häufig eine unsichere Situation auftritt. Eine Definition, was unter einer unsicheren Situation verstanden wird, geben die Autoren nicht. Als weiteres Kriterium wird die Geschwindigkeit des Fahrzeugs während des Manövers herangezogen.

Der Ansatz von Likmeta *et al*.[27] kombiniert den Einsatz von ML mit einem Zustandsautomaten. Für die Evaluierung greifen die Autoren auf mehrere Metriken zurück. Sie vergleichen ihren Algorithmus, der angelernt wird, zwischen verschiedenen Regelmodi zu wechseln, mit einem ML-Algorithmus, der sich frei im Szenario bewegen darf. In einem ersten Schritt werden die beiden Varianten im nicht angelernten Zustand verglichen. Beide Algorithmen sind in diesem Zustand so, dass eine zufällige Aktion ausgewählt wird. Als Vergleich wird der erreichte Reward herangezogen, der auch für den Lernprozess der Algorithmen genutzt wird. Nach dem Lernprozess findet eine erneute Auswertung der Metrik statt. Um die Auflösung der Evaluierung zu erhöhen, erweitern die Autoren die Bewertung um Metriken wie die maximal gefahrene Distanz im Szenario, die Zeit, die der Agent auf dem rechten Fahrstreifen verbringt, wie oft ein anderes Fahrzeug blockiert wird und wie oft Fahrstreifenwechsel durchgeführt werden. Diese Metriken werden ebenfalls für nicht angelernte Algorithmen ausgewertet.

Ein eigenständiger Ansatz für die Bewertung wird von Sander *et al*.[28] verfolgt. In der Studie werden mehrere RL-Trainingsvarianten verglichen für die Aufgabe, ein Rennfahrzeug in einer Rennumgebung zu trainieren. Für die Evaluation wird ein Elo-Ratingsystem verwendet. Dieses System, das vor allem durch den Einsatz im Schach bekannt ist, basiert darauf, dass für ein Spiel mit zwei Spielern für jeden Spieler ein Erwartungswert in Abhängigkeit der Gesamtpunktzahl beider Spieler berechnet wird. Abhängig vom Erwartungswert für jeden Spieler und dem Ausgang des Spiels werden jedem Spieler Punkte gutgeschrieben oder abgezogen. Eine Beschreibung des Testszenarios, das für den Vergleich herangezogen wird, wird in der Veröffentlichung allerdings nicht gegeben.

Wang *et al*.[29] nutzen für ihren Zustandsautomaten-Ansatz eine Kostenfunktion. Diese Kostenfunktion wird zum einen verwendet, um zwischen verschiedenen vorgegebenen Verhaltensvarianten (bspw. Fahrstreifenwechsel nach rechts) hin und her zu wechseln. Gleichzeitig nutzen die Autoren die Kostenfunktion, um die Güte ihres Fahralgorithmus zu bestimmen, indem für jeden Zustand, wie bei einem RL-Algorithmus, ein Reward bestimmt wird. Der Algorithmus schneidet bei dieser Art der Bewertung also besser ab, wenn er mehr Punkte sammelt. Die verwendete Kostenfunktion setzt sich dabei aus den folgenden fünf Teilen zusammen:

[27] Likmeta, A. et al.: Combining RL with rule-based controllers for autonomous driving (2020).

[28] Sander, R.: Emergent Autonomous Racing Via Multi-Agent Proximal Policy Optimization (2020).

[29] Wang, P. et al.: Driving behavior decision making based on a benefit evaluation model (2020).

- Platzbedarf: Der longitudinale verfügbare Platz wird über die Time-to-collision[30] zu vorausfahrenden und hinterherfahrenden Fahrzeugen bestimmt.
- Sicherheit: Die gefühlte Sicherheit eines Fahrzeuginsassen wird anhand der Klasse des vorausfahrenden Fahrzeuges bewertet. Größere Fahrzeuge gehen mit einem höheren Risiko in diese Metrik ein.
- Effizienz: Der Quotient aus der erlaubten Geschwindigkeit und der gefahrenen Geschwindigkeit des eigenen Fahrzeugs beschreibt, wie effizient das Fahrzeug bedingt durch die eigene Wahl des Fahrstreifens durch den Verkehr fährt.
- Ökonomie: Die Behinderung von nachfolgenden Fahrzeugen wird als negativ für die Ökonomie angesehen, da auf diese Weise kinetische Energie verloren geht. Daher geht die Geschwindigkeitsdifferenz des eigenen Fahrzeugs und des dahinter fahrenden Fahrzeugs als Ökonomiefaktor in die Bewertung ein.
- Verkehrsregeln: Zeigt das Fahrzeug ein Verhalten, dass gegen die geltenden Verkehrsregeln verstößt, resultiert eine hohe Strafe.

Die Gewichtung der einzelnen Faktoren wird durch die Autoren angepasst. Eine genauere Evaluation des Vorgehens erfolgt nicht.

3.1.2. Fazit

Die Literatur zeigt, dass bereits Ansätze aus dem Rennsport existieren, die zeigen, wie es möglich ist, Rennfahrer zu beurteilen. Diese Beurteilung wird als Grundlage genutzt, um die Abstimmung eines Fahrzeugs an den jeweiligen Fahrer anzupassen. Allerdings wird selbst in einschlägiger Literatur für Rennfahrzeuge außer Acht gelassen, dass auch auf der Rennstrecke Fahrzeuge interagieren. Die Beurteilung und Datenauswertung beziehen sich alleinig auf den Fahrer selbst.

Für die Verhaltensbewertung zeigt sich, dass vorrangig statistische Methoden verwendet werden. Einzelne Szenarien sind zwar gut geeignet, um einzelne Stärken bzw. Schwächen zu veranschaulichen. Eine qualitative Aussage mit hoher Aussagekraft ist allerdings nur über eine große Zahl an getesteten Szenarien sinnvoll.

Die Bewertung von Verhalten baut darauf auf, dass die Interaktion zwischen mehreren Objekten bewertet wird. Ziel ist, eine nach Möglichkeit allgemeingültige Aussage zu erhalten, ob ein Algorithmus in der Lage ist, eine Interaktion mit einem anderen Objekt erfolgreich durchzuführen. Ein zentraler Punkt ist daher die Qualität und Vielfältigkeit des verwendeten Maßstabs bzw. des verwendeten Interaktionspartners.[31] Wird bspw. versucht, die Performance beim Überholen von anderen Fahrzeugen festzustellen,

[30] Vgl. Winner, H. et al.: Handbuch Fahrerassistenzsysteme (2015), S. 200.

[31] Vgl. Zhou, M. et al.: Scalable Multi-Agent RL Training School for Autonomous Driving (2020).

schmälert ein Interaktionspartner, der deutlich langsamer als das eigene Fahrzeug fährt, die Aussagekraft eines Tests deutlich.[32]

Der bei NN häufig verwendete Ansatz, über den erreichten Reward während des Lernprozesses die Güte des resultierenden Algorithmus festzustellen[33], ist stark von der verwendeten Referenzfunktion abhängig. Nur Aspekte, die auch in der Reward-Funktion für ein Szenario implementiert sind, gehen in diesem Fall in die Bewertung ein. Die verwendete Reward-Funktion beinhaltet damit implizit auch die Art und Weise, wie das Szenario bewertet wird. Ebenfalls geht dieser Ansatz davon aus, dass das Anwendungsszenario gleich dem Trainingsszenario ist und das eine gute Performance im Training gleichzusetzen ist mit einer guten Performance in der Anwendung selbst. Dadurch ist es bspw. nicht möglich, Algorithmen zu bewerten, die anhand einer Serie verschiedener Szenarien trainiert werden, wie es in der Studie von Song *et al.*[34] demonstriert wird. Hier zeigt sich, dass durch die Art, wie NN trainiert werden, das gewünschte Verhalten, der modellierte Reward und das erlernte Verhalten nicht immer deckungsgleich sind, denn gerade wenn ein Agent aus mehreren verschiedenen Lernszenarien trainiert wird, hat jedes Szenario einen speziellen Fokus und dementsprechend auch eine Reward-Funktion, die diesen Fokus widerspiegelt. Auch die verschiedenen Strategien, einen Reward zu modellieren, erschweren die Bewertung alleinig anhand des Rewards. Oft wird nur für ein bestimmtes Ergebnis ein Reward vergeben (sparse rewards), um zu verhindern, dass durch einen kontinuierlichen Reward (dense rewards), das erlernte Verhalten durch die Reward-Funktion negativ beeinflusst wird.[35] Die Studie von Knox *et al.*[36] befasst sich ausschließlich mit dem Design von Reward-Funktionen und den Problemen, die dabei auftreten, mit Bezug auf das automatisierte Fahren. Die Autoren gehen insbesondere darauf ein, dass durch die Berechnungsweise der Rewards bei RL eine Priorisierung verschiedener möglicher Resultate in einem Szenario erreicht wird. Sie formulieren die Forderung, dass die Priorisierung als Grundlage für die Definition der Reward-Funktion genutzt und auch beim Design der Szenarien berücksichtigt wird.

Der Lernprozess von NN beeinflusst die Eignung von Reward-Funktionen zur Evaluation aber ebenfalls. Bekannte Probleme beim Training von NN sind bspw. das "Catastropic Forgetting"[37], da das Lernen von neuem Wissen immer auch Einfluss auf

[32] Vgl. Huang, H.-H. et al.: Learning overtaking and blocking skills in simulated car racing (2015).

[33] Vgl. Gu, S. et al.: Continuous Deep Q-Learning with Model-based Acceleration (2016);
vgl. Likmeta, A. et al.: Combining RL with rule-based controllers for autonomous driving (2020);
vgl. Cai, P. et al.: High-speed Autonomous Drifting with Deep Reinforcement Learning (2020);
vgl. Li, D. et al.: RL and Deep Learning based Lateral Control for Autonomous Driving (2018).

[34] Song, Y. et al.: Autonomous Overtaking in Gran Turismo Sport Using Curriculum RL (2021).

[35] Vgl. Osiński, B. et al.: CARLA Real Traffic Scenarios - novel training ground and benchmark (2020).

[36] Knox, W. B. et al.: Reward (Mis)design for Autonomous Driving (2021).

[37] Vgl. McCloskey, M. et al.: Catastrophic Interference in Connectionist Networks (1989).

bereits erlernte Verhaltensweisen hat.[38] Ein weiteres Problem, insbesondere bei RL, ist der Umgang mit seltenen Ereignissen[39], die durch die geringe Exposition auch nur einen geringen Einfluss auf den Lernprozess bzw. den Datenpool nehmen, der zum Optimieren des NN verwendet wird.[40] Dasselbe gilt in diesem Zusammenhang auch für die Auswertung. Ereignisse, die nur sehr selten auftauchen, haben auch keinen signifikanten Einfluss auf eine statistische Auswertung der erreichten Rewards. Auch haben NN die Eigenschaft, dass Zusammenhänge erkannt werden, die so nicht vom Ersteller des Szenarios beabsichtigt sind[41], oder aber, dass der Algorithmus durch die berücksichtigten Faktoren in der Reward-Funktion Strategien entwickelt, die nicht das tatsächliche Problem lösen, aber trotzdem zu einem besonders hohen Reward führen.[42] In diesem Fall ist die Reward-Funktion ebenfalls nur bedingt geeignet, um die Güte eines Algorithmus festzustellen.

Der Ansatz von Christiano[43] zeigt allerdings, dass auch möglich ist, die Reward-Funktionen erfolgreich zu erlernen, sodass eine händische Definition nicht mehr nötig ist. Aber auch bei diesem Ansatz gibt es keine Garantie, dass die Reward-Funktion tatsächliche Aussagekraft bezüglich des ursprünglich gewünschten Verhaltens besitzt.

Die Vorschläge für die Bewertung von autonomen Fahrzeugen bspw. von Zhou *et al*. bringen mit der Agilität, Stabilität oder Sicherheit erste Metriken ins Spiel, die sich auf die gewünschte Fahrfunktion und nicht auf die zu formulierte Reward-Funktion stützen.[44] Likmeta *et al*. schlagen in diesem Zusammenhang eine Auswahl von Metriken vor, die sich durch Relevanz für den Kunden eines autonomen Straßenfahrzeugs auszeichnet.[45] Die zusätzliche Auswertung nach Metriken von Zhou *et al*., die das Verhalten angelehnt an die Spieltheorie in kooperativ und kompetitiv unterteilen, zeigt einen ersten Ansatz für die Bewertung der Interaktion von Straßenfahrzeugen.

3.2. Ratingsysteme

Bei der Bewertung von Nullsummenspielen werden vielen Fällen Ranking-Systeme eingesetzt, die jedem Spieler einen Rang oder Punktzahl zuweisen, die den Spieler als gut oder schlecht im Vergleich mit anderen Spielern klassifizieren. Das Ziel, das dabei verfolgt wird, ist, alle vorhandenen Spieler in eine Rangfolge zu bringen und damit auf einer Ordinalskala einzuordnen. Der Vergleich von Spielern ist aber nicht die

[38] Vgl. Ratcliff, R.: Connectionist models of recognition memory (1990).

[39] Vgl. Frank, J. et al.: Reinforcement learning in the presence of rare events (2008).

[40] Vgl. Kaiser, Ł. et al.: Learning to Remember Rare Events (2017).

[41] Vgl. Jacobsen, J.-H. et al.: Shortcuts: Neural Networks Love To Cheat (2020).

[42] Vgl. Everitt, T. et al.: Reinforcement Learning with a Corrupted Reward Channel (2017).

[43] Christiano, P. et al.: Deep reinforcement learning from human preferences (2017).

[44] Vgl. Zhou, M. et al.: Scalable Multi-Agent RL Training School for Autonomous Driving (2020).

[45] Vgl. Likmeta, A. et al.: Combining RL with rule-based controllers for autonomous driving (2020).

einzige Funktion eines Ratingsystems. Es dient ebenfalls dazu, möglichst spannende Spieler-Paarungen herauszufinden, also Paarungen, bei denen ein eindeutiger Sieg eines einzelnen Spielers unwahrscheinlich ist, um die Spiele für Spieler und Zuschauer möglichst anziehend zu gestalten.

Es ist also ebenfalls die Aufgabe eines Ratingsystems, die Wahrscheinlichkeit für einen Sieg eines Spielers gegen einen anderen Spieler vorauszusagen. Diese Wahrscheinlichkeit, dass ein Spieler $n \in [1, .., \mathcal{N}]$ gegen einen anderen Spieler $m \in [1, .., \mathcal{N}]$ gewinnt, wird im Folgenden als p_{nm} bezeichnet, wobei $\mathcal{N}$ die Gesamtanzahl an verfügbaren Spielern bezeichnet. Die Wahrscheinlichkeiten p_{nm} für alle Spielerkombinationen sind in der sogenannten Payoff-Matrix $\boldsymbol{p}$ gesammelt.

Die Payoff-Matrix $\boldsymbol{p}$ wird für ein eindimensionales Bewertungssystem als transitiv angenommen. Das bedeutet, dass in einem fiktiven Spiel mit drei Spielern A, B und C, wenn für die Paarung A vs. B die Gewinnchance bei 2/1 und ebenfalls für die Paarung B vs. C die Gewinnchance bei 2/1 liegt, dass die Gewinnchance für die Paarung A vs. C mit 4/1 angenommen werden darf. Ein eindeutiges Beispiel für eine solche Beziehung sind Rennwettbewerbe, bei denen der schnellste Teilnehmer gewinnt. Die Spielstärke ist somit auf einer eindimensionalen Achse verteilt. Dieser Art der Beziehung wird als **transitiv** bezeichnet.

Im Gegensatz dazu steht eine zyklische Payoff-Matrix, wie es bei dem Spiel Schere-Stein-Papier (SSP) bekannt ist. Alle möglichen reinen Strategien sind bei dem Spiel gleichwertig, da sie gegen jeweils eine andere Strategie gewinnen und gegen die andere verlieren. In diesem Fall wird von **Intransitivität** oder einer **zyklischen Payoff-Matrix** gesprochen.[46]

Der folgende Abschnitt stellt bestehende Ansätze von Rankingmethoden vor und geht auf die jeweiligen Fokuspunkte ein.

3.2.1. Bestehende Ansätze

Eines der bekanntesten Ratingsysteme ist das Elo-System[47], das heutzutage vor allem im Schach und Go in Abwandlung eingesetzt wird, um die Fähigkeiten von verschiedenen Spielern gegeneinander zu bewerten.[48] Das System berechnet für jede Partie einen erwarteten Ausgang p_{nm} anhand der Punktzahlen der jeweiligen Gegenspieler. Nach der Partie wird, je nach Ausgang dem Gewinner eine bestimmte Anzahl an Punkten vom Punktekonto des Verlierers gutgeschrieben. Die Anzahl der übertragenen Punkte hängt dabei von der Differenz der Punktzahlen der Spieler vor dem Spiel ab. Dem Elo-System liegt die Annahme zugrunde, dass die Fähigkeiten der Spieler als

[46] Vgl. Duan, J. et al.: A Generalized Model for Multidimensional Intransitivity (2017).

[47] Elo, A. E.: The Rating of Chessplayers, Past and Present (1978).

[48] Vgl. Ebtekar, A. et al.: An Elo-like System for Massive Multiplayer Competitions (2021).

transitiv angenommen werden und dass das Fähigkeiten-Niveau eines Spielers sich innerhalb einer bestimmten Normalverteilung befindet. Die Elo-Punktzahl beschreibt den Mittelwert dieser Normalverteilung, für die eine konstante Standardabweichung vorausgesetzt wird.

Als Weiterentwicklung des Elo-Systems wurde 2007 von Microsoft das TrueSkill-Ratingsystem vorgestellt.[49] Es erweitert das Elo-Rating von einem reinen Mittelwert auf einen Mittelwert mit einer spielerindividuellen Standardabweichung. Der wahre Fähigkeitswert eines Spielers wird also zu Anfang nur mit einer großen Standardabweichung als nicht genau bestimmt angesehen. Der angegebene Punktewert eines Spielers ist dabei nicht der Mittelwert der Verteilung, sondern der Mittelwert abzüglich des dreifachen Wertes der Standardabweichung. Die Punktzahl ist damit eine konservative Schätzung des wahren Wertes. Nach jedem Spiel wird die Standardabweichung und der Mittelwert eines jeden Spielers aktualisiert. Das System hat damit im Gegensatz zu Elo den Vorteil, dass das Fähigkeiten-Niveau eines Spielers schneller gegen den wahren Wert konvergiert, da die Updates des Mittelwertes bei einer hohen Standardabweichung entsprechend größer sind, als wenn ein Spieler bereits viele Spiele gespielt und damit eine geringe Standardabweichung hat. Zudem erweitert TrueSkill das Rating von einem reinen 2-Spieler-Spiel auf die Möglichkeit, auch Teamspiele zu bewerten, da jeder Spieler individuelle Updates seines Fähigkeiten-Niveaus erhält, wodurch es möglich ist, die Punkteinflation[50] des Elo-Systems zu stoppen. Ein ähnlicher Ansatz wie TrueSkill wird vom Glicko-System und Glicko2-System verfolgt.[51] In Glicko2 werden Mittelwert und Standardabweichung zusätzlich noch durch einen Schwankungswert ergänzt, der die Leistungskonstanz von Spielern zusätzlich honoriert.

Der Ansatz eines Ranking-Systems von Prisco *et al*.[52] zielt nicht auf das Ranking von einzelnen Spielern ab, sondern verfolgt das Ziel, Lerninhalte mit angepasstem Schwierigkeitsgrad für Studierende zu finden, um den Lernprozess dieser zu optimieren, indem herausfordernde, aber nicht überfordernde Lerninhalte präsentiert werden. Daher wird das Elo-System auf eine zusätzliche Dimension erweitert, sodass nicht nur den Studierenden ein Fähigkeiten-Niveau zugewiesen wird, sondern auch den zu lösenden Aufgaben. Das System wird zusätzlich erweitert, da die Autoren davon ausgehen, dass die zu lernenden Probleme nicht über ein einzelnes Fähigkeiten-Niveau abgebildet werden, sondern dass die Aufgaben erfordern, Wissen in mehreren Bereichen zu erlernen, die nicht komplementär sind und unter Umständen sogar orthogonal zueinander erlernt werden. Den verschiedenen Aufgaben werden jeweils Gewichte für die einzelnen möglichen Fähigkeiten zugewiesen, sodass das Fähigkeiten-Niveau entsprechend ange-

[49] Vgl. Herbrich, R. et al.: TrueSkill : A Bayesian Skill Rating System (2007).

[50] Vgl. Veček, N. et al.: A Comparison between Ratings for Ranking Evolutionary Algorithms (2014).

[51] Glickman, M.: Example of the Glicko-2 system (2013).

[52] Prisco, A. et al.: A multidimensional ELO model for matching learning objects (2018).

passt wird. Damit erhält jeder Spieler eine Punktzahl für eine Vielzahl an Fähigkeiten, die in Abhängigkeit der Gewichtung von bearbeiteten Aufgaben upgedatet werden. Der Ansatz ist damit nicht in der Lage, Spieler gegen Spieler miteinander zu evaluieren, sondern die Performance der Spieler in Bezug auf verschiedene Aufgaben. Gleichzeitig wird durch die Multidimensionalität der Fähigkeiten die Annahme der Transitivität aufgeweicht und es ist möglich, auch Probleme zyklischer Natur zu analysieren.

Ein Ansatz auf Basis von Spieltheorie wird von Balduzzi *et al*.[53] vorgestellt. Der Ansatz zielt auf eine allgemeine Verbesserung des Elo-Systems ab, ohne einen dedizierten Anwendungsfall. Die getroffene Grundannahme des Ansatzes ist, dass das Rating von mehreren Spielern als Meta-Spiel formuliert wird, in dem zwei Spieler gegeneinander antreten. Die Aktion, die die beiden Spieler treffen, ist, jeweils einen Spieler oder Spielstrategie auszuwählen, um das Spiel zu gewinnen. Als Informationsquelle dient dabei die Payoff-Matrix $\boldsymbol{p}$. Das Nash-Gleichgewicht maximaler Entropie (maxent NG)[54] zeigt an, welche Verteilung an Agenten gewählt wird für einen Sieg bei einer gleichzeitig maximal zufälligen Strategie.[55] Es dient damit als Indikator für das Ranking der einzelnen Algorithmen untereinander. Dieser Ansatz ist geeignet, sowohl Spieler-vs.-Spieler-Performance als auch Spieler-vs.-Aufgaben-Performance zu evaluieren, da es möglich ist, die Aufgaben wie auch bei der Methode von Prisco als Spieler zu modellieren. Der Ansatz ergibt zusätzlich die Möglichkeit, die Schwierigkeit der einzelnen Aufgaben anhand der Performance der einzelnen Agenten zu evaluieren. Mit dem Ansatz der Evaluierung über maxent NG entfällt die Schwäche der bereits präsentierten Methoden, dass sie beeinflussbar sind gegenüber (a) redundanten Agenten und (b) redundanten Aufgaben. Der Ansatz erfordert allerdings im Gegensatz zu Elo eine Information über die jeweilige Wahrscheinlichkeit für jeden Agenten gegenüber jedem anderen Agenten zu gewinnen und ist damit nicht für große Ranking-Systeme mit menschlichen Spielern geeignet, da (a) die Menge der notwendigen Einträge in $\boldsymbol{p}$ quadratisch mit der Anzahl an Spielern wächst, (b) es notwendig ist, für jede Spieler-vs.-Spieler-Kombination eine Gewinnwahrscheinlichkeit zu ermitteln und damit auch eine statistisch signifikante Anzahl an Spielen für jede Spieler-vs.-Spieler-Kombination durchzuführen. Zudem ist die Berechnung eines maxent NG für viele Agenten/Aufgaben mit hoher Rechenintensität verbunden. Das sich bildenden maxent NG beinhaltet mit einem großen Anteil die Algorithmen mit einem hohen Ranking. Algorithmen mit einem niedrigen Ranking sind dagegen in der Verteilung mitunter nicht vertreten, da eine optimierte Strategie für das Meta-Spiel die Verwendung von Algorithmen mit

[53] Balduzzi, D. et al.: Re-evaluating Evaluation (2018).

[54] Das maxent NG zeigt den Zustand an, in dem ein Spieler keinen Vorteil daraus erhält, wenn er alleine von seiner Strategie abweicht, ohne eine entsprechende Reaktion des Gegners.

[55] Für den einfachen Fall des Spiels Schere-Stein-Papier ist diese Verteilung $\left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$, da jede Strategie den anderen gegenüber gleichwertig ist. Die ideale Strategie ist daher eine zufällig gewählte Strategie, die jede Strategie gleichermaßen gewichtet.

schlechter Performance ausschließt. Algorithmen mit schlechter Performance lassen sich daher in dem Rating nicht voneinander unterscheiden.[56]

Zusätzlich stellen Balduzzi *et al*.in derselben Veröffentlichung multidimensionales Elo (mElo) vor, eine Erweiterung des Elo-Systems. Das einfache Ranking von Elo wird dabei um eine Interaktionsmatrix erweitert. Damit ist das Verfahren in der Lage, auch intransitive Interaktionen korrekt abzubilden.

α Rank bezeichnet ein evolutionäres Verfahren für die Evaluation von Agenten bzw. Spielstrategien für Nullsummen- und Nicht-Nullsummenspiele, das von Omidshafiei *et al*.[57] veröffentlicht wurde. Als Basis dient wie auch für Balduzzi's Methode ein Meta-Spiel, bei dem zwei oder, wie im Falle von α Rank auch mehrere Spieler eine Strategie oder einen speziellen Agenten wählen. Die Kernidee ist dabei, dass mittels einer Markov-Kette die Evolution von einer Spielstrategie zu einer anderen Spielstrategie zulässig ist. Damit wird nicht nur ein reines maxent NG ermittelt, sondern es ist möglich, auch dynamische Strategiewechsel zu modellieren, indem in jeder reinen Spielstrategie zufällige, seltene Änderungen auftreten, denen möglich ist, wenn sie der ursprünglichen Strategie überlegen sind, auch einen Strategiepool zu übernehmen. Der Parameter α beschreibt für den Ansatz den einzigen Hyperparameter, der als Verstärkungsfaktor der Wahrscheinlichkeitsfunktion dient, die die Reproduktion von einer zufälligen Strategieänderung der Strategiepopulation in einem Pool gleicher Strategien beschreibt. Für jeden Parameter α ist möglich, eine Verteilung zu berechnen, wie lange ein bestimmte Strategie im Vergleich zu anderen Strategien aktiv ist. Durch einen Parameter-Sweep von α wird der Punkt der Konvergenz dieser Verteilung gefunden. Die Autoren nennen als wesentliche Vorteile der Methode, dass anders als für maxent NG mit α Rank möglich ist, auch nicht Nullsummenspiele zu evaluieren sowie eine beliebige Anzahl an Spielern. Auf diese Weise ist auch der Einfluss mehrerer Spielstrategien untereinander bewertbar. Die Methode ist laut der Veröffentlichung von Yang *et al*.[58] nur unter hohem Rechenaufwand auf eine hohe Anzahl an Strategien und Spielern aufgrund von exponentiell wachsendem Rechenleistungsbedarf skalierbar, verursacht durch die exponentiell ansteigende Anzahl an Strategiekombinationen. Sie schlagen die adaptierte Methode α^α Rank vor, die eine anfänglich zufällige Verteilung mehrerer Strategiepools nutzt und die Ermittlung der statistischen Verteilungen bei einem bestimmten Wert von α durch numerische Optimierung durchführt. Auf diese Weise ist den Autoren möglich, die Anforderungen an Rechenleistung und Speicher für die Ermittlung der jeweiligen Rangfolgen für einzelne Strategien um drei Größenordnungen zu senken.

[56] Vgl. Vinyals, O. et al.: AlphaStar: Mastering the Real-Time Strategy Game StarCraft II (2019); vgl. Balduzzi, D. et al.: Re-evaluating Evaluation (2018).

[57] Omidshafiei, S. et al.: α-Rank: Multi-Agent Evaluation by Evolution (2019).

[58] Yang, Y. et al.: α^α-Rank: Practically Scaling α-Rank through Stochastic Optimisation (2020).

3.2.2. Fazit

Die Übersicht über den Stand der Technik macht die bisherigen Fokuspunkte bereits existierender Ranking-Systeme deutlich. Etablierte Bewertungsschemen wie Glicko, Elo oder TrueSkill haben einen eindeutigen Fokus auf die Anwendung mit menschlichen Spielern. Es wird nur eine einzelne Punktzahl für jeden Spieler berechnet und die Systeme basieren alle auf der Annahme der Transitivität der Fähigkeiten der Spieler. Für menschliche Spieler ist diese Annahme zwar nicht ausnahmslos gültig, da menschliche Spieler einem ständigen Lernprozess unterliegen und ihre Fähigkeiten nicht statisch sind, so erscheint die Annahme der Transitivität trotzdem in der Praxis valide. Dies stützt sich auf die Annahme, dass eine langsame Anpassung an Strategien, die im jeweiligen Fähigkeiten-Niveau eines menschlichen Spielers etabliert sind, möglich ist.

Es wird ebenfalls deutlich, dass die Ranking-Systeme für Menschen darauf aus sind, die Payoff-Matrix $\boldsymbol{p}$ unter Annahme der Transitivität zu schätzen. Der Gedanke dahinter ist, dass durch die Transitivität die Dimension von $\boldsymbol{p}$ auf eine einzelne Ordinalskala reduziert wird. Nur dadurch ist ein groß angelegtes Ranking-System wie Elo umsetzbar, da bei mehreren Millionen menschlichen Spielern die Ermittlung einer voll-besetzten Payoff-Matrix nicht möglich ist.

Algorithmen für die Generierung von Verhalten und Interaktion besitzen in der heutigen Zeit nur in seltenen Fällen die Fähigkeit, sich an eine Vielzahl an Strategien anzupassen. Durch den Lernprozess, der für NN hauptsächlich auf Stochastik basiert, werden die Algorithmen angelernt, eine möglichst hohe Punktzahl in der maximalen Anzahl an Fällen zu erreichen. Nachdem die Algorithmen angelernt wurden, findet in den meisten Fällen kein Lernprozess mehr statt. Damit wird ein NN ebenso wie deterministische Algorithmen als eine feststehende Strategie verstanden. Die tatsächlich verfolgte Strategie des Agenten hängt damit zur Gänze von den Bedingungen während des Lernprozesses ab[59] oder im Falle eines deterministischen Algorithmus von den Fähigkeiten des Entwicklers. Die Transitivität ist damit nur noch eingeschränkt gegeben, da eine Strategie, die in der Entwicklungsumgebung funktioniert, nicht notwendigerweise gegen andere Agenten oder Menschen gleichermaßen effektiv ist.

Ein weiterer Unterschied zwischen Systemen für die Bewertung von Menschen und Systemen für den Vergleich von ML-Algorithmen oder Spielstrategien ist die Verfügbarkeit. Während möglich ist, mit Algorithmen in schnell rechnenden Umgebungen mehrere Tausend Spiele täglich zu rechnen, sind Menschen dazu nicht in der Lage. Daher erstellen Systeme wie Elo iterativ ein Ranking eines Spielers unter Berücksichtigung der Fähigkeiten der jeweils anderen Spieler. Bedingt durch die Intransitivität von ML-Algorithmen und deren Verfügbarkeit kommen hier für die Bewertung multidimensionale Verfahren wie das maxent NG von Balduzzi oder α Rank zum Einsatz.

[59] Vgl. Lee, C.-S. et al.: Human vs. Computer Go: Review and Prospect (2016).

Hier ist allerdings notwendig, dass die Gewinnwahrscheinlichkeiten für jede mögliche Spielerkombination und jede Spieleranzahl bekannt ist. Die Information über ein einzelnes Spiel genügt in diesem Fall nicht. Diese Verfahren berücksichtigen ebenfalls keine Wiederholung einer Interaktion, da für ein Spiel zweier Algorithmen keine unterschiedlichen Ergebnisse erwartet werden, im Gegensatz zu einer Mensch-gegen-Mensch-Interaktion. Der Zweck der Methoden wechselt damit von einer Approximation von $\boldsymbol{p}$ auf die Bestimmung einer Rangfolge von Algorithmen, obwohl die Annahme der Transitivität nicht gültig ist. Das Ziel ist damit die Reduzierung der Komplexität der mehrdimensionalen Payoff-Matrix auf eine ordinale Skala, um die Interpretierbarkeit einfacher zu gestalten.

Ein Evaluation von mehreren Fähigkeiten in Bezug auf verschiedene Aufgaben wird nur mittels des Verfahrens von Prisco und vergleichbaren Lernsystemen[60] über mehrere Elo-Skalen und das mElo-Verfahren von Balduzzi möglich. Hier wird die Rangfolge aber nicht notwendigerweise durch Spieler-gegen-Spieler-Szenarien gebildet, sondern es ist möglich, jeden Spieler zusätzlich gegen verschiedene Aufgaben zu messen. Damit ist der erwartete Erfolg p_{nm} nicht mehr beschränkt auf die Wahrscheinlichkeit, gegen einen anderen Spieler zu gewinnen, sondern wird als erwartete Güte verstanden, mit der eine gegebene Aufgabe gelöst wird.

Zunächst beschränkt sich die Spieltheorie auf die Anwendung der Methoden auf einfache Nullsummenspiele. Erste Ansätze der Methoden auf komplexe Spiele wie StarCraft zeigen aber das Potenzial der Methoden.[61] Die Stärken und Schwächen der einzelnen Methoden werden in Tabelle 3-1 zusammengefasst.

[60] Vgl. Abdi, S. et al.: Modelling Learners in Adaptive Educational Systems (2021).

[61] Vgl. Vinyals, O. et al.: AlphaStar: Mastering the Real-Time Strategy Game StarCraft II (2019).

Tabelle 3-1.: Zusammenfassung der Stärken und Schwächen der vorgestellten Ranking-Systeme.
Legende:

+ : Methode ist geeignet die Eigenschaft oder Aufgabe abzubilden

○ : Methode ist eingeschränkt geeignet die Eigenschaft oder Aufgabe abzubilden

- : Methode ist nicht geeignet die Eigenschaft oder Aufgabe abzubilden

	ELO	Glicko, TrueSkill	mElo	Nash Gleichgewicht	α Rank
Payoff Matrix berechnen	+	+	+	-	-
Konvergenz	-	○	+	+	+
In-Transitivität	-	-	+	+	+
Spieler vs. Spieler	+	+	+	+	+
Spieler vs. Aufgabe	+	+	+	+	-
Redundante Spieler	-	+	+	+	+
Redundante Aufgaben	-	-	+	+	-
Sehr viele Spieler	+	+	+	-	-

4. Vorgehen des Benchmarkings und das System Motorsport

Neben der relativen Evaluation in Form von Ranking-Systemen wird die Evaluierung eines technischen Systems üblicherweise in Form einer Validierung durchgeführt. Die Validierung nutzt eine individuelle Menge an Tests, die genutzt werden, um die zuvor definierten Anforderungen zu erfüllen.[62] Ziel der Validierung ist festzustellen, ob ein System den Anforderungen entspricht und ob ein System eine ausreichende Güte besitzt, unter der Annahme, dass die Erfüllung der Anforderung in einem "guten" System resultiert.

Im Folgenden werden in Abschnitt 4.1 diese Bedingungen für eine Systembeurteilung diskutiert und führen auf die in dieser Arbeit verfolgte Variante der Systembeurteilung, dem Benchmarking. Im Anschluss daran komplettiert die Analyse des Systems Motorsport in Abschnitt 4.2 mit Blick auf die verfolgten Ziele und Rahmenbedingungen die Grundlagen dieser Arbeit.

4.1. Systembeurteilung und Einführung in den Prozess des Benchmarkings

Die Durchführung einer Validierung eines technischen Systems setzt voraus, dass im Vorhinein eine bestimmte Menge an Anforderungen definiert wurde, die das System hinreichend genau beschreibt. Mit der Validierung ist möglich, diese Anforderungen abzutesten und damit die Qualität des Systems zu überprüfen. Voraussetzung für dieses Vorgehen ist, dass die formulierten Anforderungen quantifizierbar und damit auch bewertbar sind.

Im Unterschied zu dieser Vorgehensweise ist in einem Rennen das Verhalten eines Rennteilnehmers immer im Kontext eines Nullsummenspiels zu betrachten. Die absolute Performance ist daher ein wichtiger Indikator, ob ein System "gut" ist. Je schneller ein Fahrzeug auf der Strecke bewegt wird, desto besser. Die tatsächliche Performance entsteht aber tatsächlich nur, wenn ein Algorithmus mit einer Referenz verglichen wird. Es ist nicht entscheidend, ob ein Fahrzeug schnell ist, es ist nur entscheidend, dass es vor den gegnerischen Fahrzeugen so oft wie möglich die Ziellinie überquert. Zusätzlich gibt es für die Interaktion von Rennfahrzeugen anders als im Straßenverkehr nur folgende Regeln.

[62] Rabe, M. et al.: Verifikation und Validierung - Definitionen (2008), S. 19ff.

- Es darf nur auf eine Art und Weise gefahren werden, sodass keine anderen Fahrer oder Personen in Gefahr geraten.[63]
- Ein Fahrzeug, das sich aus Streckenperspektive weiter vorne befindet, hat höhere Priorität als Fahrzeuge dahinter und damit Vorfahrt auf der Strecke.[64]
- Auf der Geraden darf ein schnelleres Fahrzeug nicht durch wiederholtes Wechseln der Fahrbahnseite am Überholen gehindert werden.[65]

Weiter sind zunächst keine quantifizierbaren Anforderungen bekannt, die eine objektive Validierung ermöglichen. Im Folgenden wird daher ein Benchmarking entwickelt, der die Performance von mehreren Algorithmen sowohl relativ zueinander vergleicht und gleichzeitig eine objektive Referenz bietet. Der folgende Abschnitt befasst sich zunächst mit den theoretischen Grundlagen des Benchmarkings.

4.1.1. Was ist Benchmarking?

Benchmarking ist ein Prozess mit dem Ziel, systematisch Informationen bezüglich der Performance von Prozessen, Programmen oder Systemen zu sammeln. Zusätzlich dazu werden Benchmarks dazu genutzt, Bereiche zu identifizieren, die einer Verbesserung im Rahmen einer Entwicklung bedürfen.[66] Das Benchmarking eines Systems zeigt daher nicht nur die Performance eines Systems in Relation zu anderen Systemen auf, sondern unterteilt gleichzeitig in unterschiedliche Bereiche, die das Gesamtergebnis beeinflussen. Auf diese Weise wird ermöglicht, etwaige Vor- oder Nachteile eines bestimmten Systems zu identifizieren und Rückschlüsse auf deren Herkunft zuzulassen.[67]

Nach Drew[68] besteht die Entwicklung und Anwendung eines Benchmarking-Prozesses aus fünf wesentlichen Phasen.[69]

- Identifizieren des zu benchmarkenden Objektes
- Auswählen eines Vergleichsmaßstabes bzw. einer Referenz
- Sammeln und Analysieren von Daten
- Setzen von Zielen für die Verbesserung
- Umsetzen des Plans und Analysieren der Ergebnisse

[63] FIA: Formula 1 Sporting Regulations Issue 5 (2021), S. 27.

[64] Auch, wenn diese Regel i. A. von allen Fahrern eingehalten wird, ist diese Regel nicht in den Regularien der FIA festgehalten.

[65] FIA: Appendix L - Driver's licenses, equipment and conduct (2021), S. 47f.

[66] Vgl. Carpinetti, L. C. et al.: What to benchmark? (2002), S. 245.

[67] Vgl. Saavedra, R. H. et al.: Analysis of benchmark characteristics and -performance (1996), S. 357.

[68] Drew, S. A. W.: From knowledge to action (1997).

[69] Vgl. Carpinetti, L. C. et al.: What to benchmark? (2002), S. 245.

Im ersten Schritt erfolgt die Auswahl des Untersuchungsobjektes. Dieses wird beschrieben als die Dimensionen, Aktivitäten und Abläufe, die essenziell für den Erfolg eines Systems sind.[70] Danach erfolgt die Auswahl des Maßstabs, an dem die Performance gemessen wird. Dieser Schritt ist ebenfalls von großer Bedeutung, da der Nachweis einer systematischen Überlegenheit nur anhand einer performanten Referenz glaubwürdig nachweisbar ist. Daran anschließend werden die notwendigen Tests durchgeführt und die Daten analysiert. Die letzten beiden Schritte beschreiben einen Verbesserungsprozess, der durch die Anwendung des Benchmarkings entsteht. Die Verbesserung und die Analyse des weiterentwickelten Produktes werden im Folgenden nicht weiter behandelt.

Damit definieren sich die Hauptziele des Benchmarkings für die Themenstellung dieser Arbeit, dass Aspekte untersucht werden, die wesentlich für den Sieg in (a) einzelnen Rennen und (b) Rennserien sind.

Innerhalb eines Benchmarkings besteht die Forderung, derart zu testen, dass nicht nur das Potenzial für das Gewinnen von Rennen aufgezeigt wird. Von den Tests, die während des Benchmarkings durchgeführt werden, ist gefordert, dass sie sich für die Identifikation sowohl von Stärken als auch Schwächen einzelner Algorithmen eignen, um die Weiterentwicklung von Algorithmen oder die Anpassung von einzelnen Hyperparametern zu ermöglichen.

4.1.2. Was sind Kriterien für einen guten Benchmark?

Das Ziel des Benchmarkings ist, unterschiedlich performante Systeme durch Tests voneinander abzugrenzen. Dabei wird das System unter verschiedenen Bedingungen Tests unterzogen, die im Anschluss anhand bestimmter Metriken evaluiert werden. Das Ergebnis davon ist eine Rangfolge bzw. eine Bewertung bezüglich verschiedener Kriterien und eine Aussage über die Stärken und Schwächen des untersuchten Systems bzw. deren Ursache.

Bezogen auf den gesamten Benchmarking-Prozess beschreibt Huppler folgende fünf Kriterien als besonders entscheidend für die Qualität eines Benchmarkings.[71a]

(1) Die Relevanz des Vergleichs wird gewährleistet durch die Auswahl von aussagekräftigen und gleichzeitig gut verständlichen Metriken. Daneben ist die einfache und unmissverständliche Interpretation der Ergebnisse essenziell. Dafür ist im Vorhinein eindeutig zu definieren, was vom Benchmarking getestet wird und was nicht. Ebenso wichtig ist, dass das Testobjekt sowohl in Bezug auf die Software als auch auf die Hardware möglichst nahe an der vorgesehenen Verwendung evaluiert wird.[71b]

[70] Vgl. Carpinetti, L. C. et al.: What to benchmark? (2002), S. 250.

[71] Huppler, K.: The Art of Building a Good Benchmark (2009), a: S. 19, b: S.20.

(2) Die Wiederholbarkeit eines Tests, dass auch nach mehrmaliger Wiederholung dieselben Ergebnisse resultieren.

(3) Fairness bezüglich verschiedener Umsetzungen. Es besteht die Forderung, dass keine Umsetzungen in verschiedenen Programmiersprachen oder systematischen Unterschieden bevorzugt oder benachteiligt werden, was durch eine gemeinsame Plattform für die Tests erreicht werden kann.

(4) Verifizierbarkeit der Tests, sodass auch bei komplexen Tests die Ergebnisse als vertrauenswürdig angesehen werden. Aus diesem Grund ist eine möglichst einfache Struktur zu wählen.

(5) Die Ökonomie des Benchmarkings beschreibt die Kosten, die durch die Durchführung entstehen. Diese Kosten sollten in einem akzeptablen Rahmen bleiben, sodass für eine Evaluierung vorhandenen Ressourcen nicht unverhältnismäßig stark beansprucht werden.

Weiter wird von einem Benchmarking gefordert, dass es an die Gesamtheit der Lösungen angepasst ist, da ansonsten die Gefahr besteht, dass eine "banale" Lösung besser abschneidet als eine höher entwickelte, da zusätzliche Fähigkeiten nicht abgetestet werden.[72a] In der Studie von Karakovskiy *et al*.[73], die sich mit der Performance von Agenten im Spiel Super Mario befasst, wurden die Tests, in denen die Algorithmen untersucht wurden, dementsprechend erweitert, sodass eine hohe Punktzahl für einfache Algorithmen weniger wahrscheinlich und eine Optimierung auf einige wenige Tests erschwert wird. Zusätzlich wurden Tests hinzugefügt, die mit Methoden, die auf einfachen Suchalgorithmen basieren, nicht zu lösen sind. Die untersuchten Kriterien und Metriken sind im Idealfall orthogonal zueinander, sodass die Aussagekraft und Trennschärfe der Metriken maximal ist.[74] Die Trennschärfe bezieht sich aber nicht ausschließlich darauf, die "guten" Algorithmen voneinander zu trennen, sondern auch die mit einer geringeren Performance[72b]. Dafür ist es notwendig, dass die Metriken/Tests so ausgelegt sind, dass nicht nur eine binäre Aussage generiert wird, sondern immer auch eine quantitative.

4.2. Analyse des Systems Motorsport und dessen Ebenen

Um eine umfängliche Repräsentation des Motorsport-Kosmos im entwickelten Benchmarking zu erhalten, analysiert der folgende Abschnitt die Ziele, die im Motorsport verfolgt werden, im Details. Dabei gliedert sich die Analyse auf die folgenden drei Ebenen. Die Rennsaison, das einzelne Rennen und ein einzelnes Überholszenario als kleinste Einheit. Das System ist schematisch in Abbildung 4-1

[72] Vgl. Karakovskiy, S. et al.: The Mario AI Benchmark and Competitions (2012).

[73] Karakovskiy, S. et al.: The Mario AI Benchmark and Competitions (2012).

[74] Vgl. Saavedra, R. H. et al.: Analysis of benchmark characteristics and -performance (1996), S. 374f.

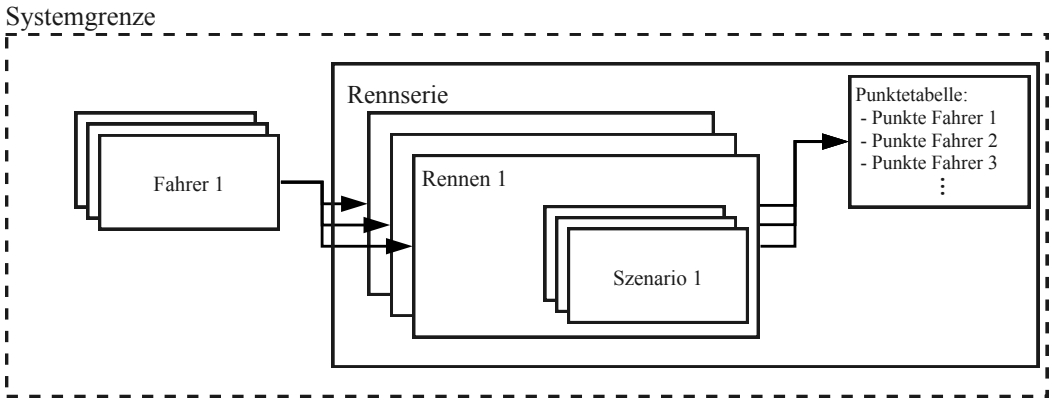


Abbildung 4-1.: Systemgrenzen des in dieser Arbeit betrachteten Motorsport-Systems

Die Rennsaison ist die oberste Einheit im Motorsport, wie auch in anderen Sportarten. Das Hauptziel besteht darin, eine Saison als Sieger abzuschließen. Dafür existiert sowohl im Rennsport als auch in anderen Sportarten üblicherweise ein Punktesystem, das je nach Ausgang eines einzelnen Rennens oder Veranstaltung eine definierte Anzahl an Punkten gewährt. Der Teilnehmer, der am Ende der Saison die meisten Punkte besitzt, ist der Sieger. Daraus leitet sich entsprechend auch das erste und wichtigste Ziel her. In einem Rennen versucht jeder Teilnehmer die maximal mögliche Anzahl an Punkten zu sammeln mit dem Ziel, am Ende einer Saison im Vergleich zu den anderen Teilnehmern insgesamt die meisten Punkte zu besitzen.

Es ist aber keineswegs notwendig, jedes einzelne Rennen zu gewinnen. Abbildung 4-2 zeigt eine Häufigkeitsverteilung (engl. Cumulative Distribution Function (CDF)) der erreichten Punktzahlen der ersten drei Platzierten der jeweiligen Formel-1-Saisons zwischen 1961 und 2020. Abbildung 4-2a zeigt die CDF der erreichten Punkte für den Erst-, Zweit- und Drittplatzierten der Saison. Daneben ist in Abbildung 4-2b die notwendige durchschnittliche Platzierung in Abhängigkeit des in der jeweiligen Saison geltenden Punkteschemas abgebildet. Beide Abbildungen verdeutlichen, dass eine konstant gute Leistung für die Rennserie der Formel-1 ebenfalls zu einem Sieg der Saison bzw. eine hohe Gesamtplatzierung in der Serie führen.

Dabei ist anzumerken, dass diese Aussage sich auf eine gesamte Saison bezieht. Diese Gewichtung verschiebt sich mit einer fortschreitenden Saison und damit verbunden einer sinkenden Anzahl zu gewinnender Punkte. Je weniger Punkte in einer Saison noch zu gewinnen sind, desto stärker polarisiert sich die Notwendigkeit, in jedem Rennen zu gewinnen, wenn das eigene Fahrzeug nur wenige Punkte im Vergleich mit der Konkurrenz erzielen konnte. Im Gegensatz dazu, wenn am Anfang einer Saison bereits deutlich mehr Punkte als die Konkurrenz eingefahren werden konnten, ist möglich, eine deutlich sicherheitsbetontere Strategie zu verfolgen.

In einem Rennen ist wie auch in der Rennsaison, das Ziel mit einer Platzierung abzuschließen, die so weit vorne im Teilnehmerfeld wie möglich liegt. Daraus ergeben

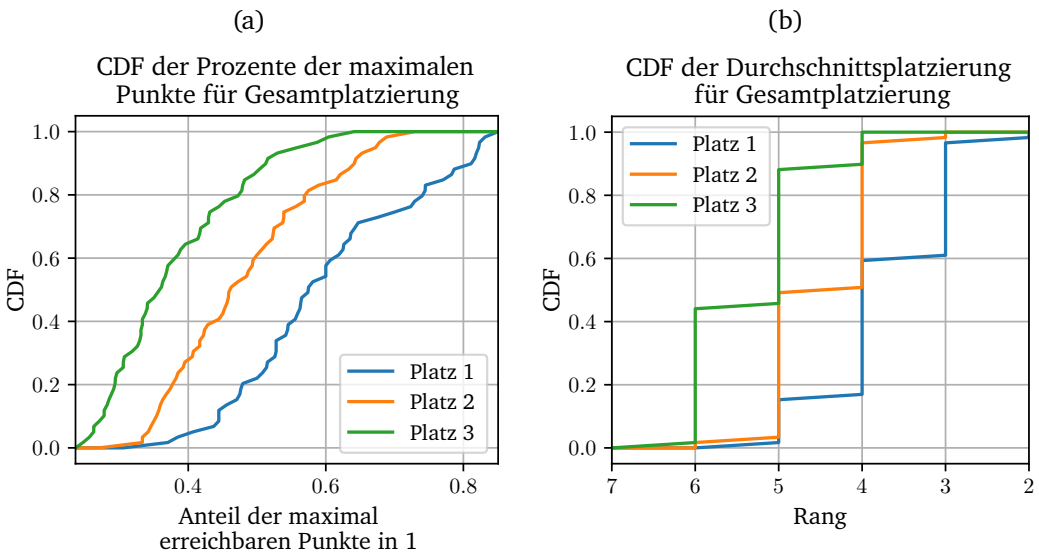


Abbildung 4-2.: Auswertung der Punkteverteilungen der Formel-1 von 1961-2020. Links: CDF der maximal erreichbaren Punkte, die in einer Saison erreicht wurden mit der jeweiligen Platzierung. Rechts: CDF der notwendigen Durchschnittsplatzierung, um eine bestimmen Rang in einer Saison zu erreichen.

sich zwei unmittelbare Ziele. Zum einen ist notwendig, dass das Fahrzeug dafür so schnell wie möglich fährt, um die Fahrzeit für eine vordefinierte Anzahl an Rennrunden zu minimalisieren. Zum anderen erhält das Fahrzeug nur Punkte, wenn das Ziel tatsächlich erreicht wird. Diese beiden Metriken werden als gegenläufig zueinander betrachtet. Je schneller ein Fahrzeug auf der Strecke fährt, desto mehr nähert es sich auch seinen eigenen physikalischen Grenzen an und es erhöht sich damit auch die Wahrscheinlichkeit für einen Unfall. Die Auslegung eines Verhaltensalgorithmus ist damit ein Optimierungsproblem, das in der Entwicklung zu lösen ist, und daraus lässt sich die erste einfache Möglichkeit ableiten, Algorithmen qualitativ zu clustern.

Die dritte zu betrachtende Komponente, die im Motorsport eine Rolle spielen kann, ist die Strategie, die ein Team für das Fahrzeug festlegt, die wie z. B. in der Formel-1, einen entscheidenden Beitrag zur Möglichkeit leistet, ein Rennen zu gewinnen.[76] Dieser Beitrag hängt im Besonderen ab von (a) dem Reglement der Rennserie, ob und auf welche Weise taktische Entscheidungen für ein Fahrzeug notwendig sind und (b) den gesammelten taktischen Entscheidungen der gegnerischen Fahrzeuge. Da es nicht möglich ist, den Einfluss taktischer Entscheidungen aus Mangel an nutzbaren Daten im Rahmen dieser Arbeit auf statistisch relevante Weise zu untersuchen, wird er an dieser Stelle nicht weiter betrachtet.

Die Zielsetzung für jedes Rennen, dass eine möglichst gute Position im Ziel erreicht

[76] Vgl. Limited, M. R.: Formula One Race Strategy (2021);
vgl. Maiza, A.: Reinforcement Learning for Formula 1 Race Strategy (2020).

wird, spiegelt sich auch auf der Ebene eines Überholszenarios wider. Zunächst wird ein Zustand im Rennen betrachtet, bei dem kein anderes Fahrzeug in der Nähe ist und eine Interaktion ermöglicht. Das zu erreichende Ziel ist daher schlicht, einen gegebenen Abschnitt in so kurzer Zeit wie möglich abzufahren. Die verwendete Metrik ist dabei der relative Rundenzeitunterschied zu einer vorher definierten Referenzmethode. Je schneller ein Algorithmus im Vergleich mit der Rundenzeitreferenz abschneidet, desto besser.

Sobald ein anderes Fahrzeug an einem Szenario beteiligt ist, kommen einige zusätzliche Punkte hinzu. Zunächst gilt auch hier, dass das oberste Ziel ist, so schnell wie möglich die Strecke zurückzulegen. Da das Gesamtergebnis allerdings nicht nach der Zeit, sondern nach der erreichten Position relativ zu den anderen Teilnehmern beurteilt wird, ist auch die bereits gefahrene Strecke bzw. die noch zu fahrende Strecke von Bedeutung. Je weiter fortgeschritten ein Rennen ist, desto mehr gewinnt die Position relativ zum anderen Fahrzeug an Bedeutung.

Ein Fahrzeug, das sich mit einer Differenzgeschwindigkeit von hinten nähert, hat per Definition eine höhere Geschwindigkeit als das eigene Fahrzeug und sollte daher in der Lage sein, das eigene Fahrzeug zu überholen. Sind im Rennen noch viele Runden zu fahren, ist möglich, dass das andere Fahrzeug eine andere Strategie verfolgt und daher eine Berücksichtigung für das eigene Rennen zunächst nicht notwendig ist. Die eigene Strategie hat in diesem Fall noch einen langen Hebelarm, um den Ausgang des Rennens zu beeinflussen. Die Zeit, die in diesem Fall verloren geht, wenn versucht wird, das andere Fahrzeug am Überholen zu hindern, wirkt sich möglicherweise gegen Ende des Rennens negativ auf das eigene Gesamtergebnis aus. Zusätzlich ist zu bedenken, dass die Wahrscheinlichkeit, dass das andere Fahrzeug im Laufe des Rennens tatsächlich überholt höher ist, je länger das Rennen noch andauert.

Ist das Rennen aber bereits in einem fortgeschrittenen Stadium, ändert sich dieser Blickwinkel. Der Hebelarm der eigenen Strategie im Vergleich zur Strategie des anderen Fahrzeugs sinkt. Damit sinkt auch der Wert der Zeit, die das eigene Fahrzeug verliert, und der Wert für den möglichen Verlust der Position auf der Strecke steigt im Verhältnis zum Wert der verlorenen Zeit an. In diesem Fall lohnt sich, das andere Fahrzeug am Überholen zu hindern, auch auf die Gefahr hin, selbst Zeit dabei zu verlieren.

Damit ergeben sich zwei zu betrachtende Grenzfälle. Im Ersten wird ein Überholen des schnelleren Fahrzeugs als unausweichlich angesehen. In dem Fall ist im Interesse beider Fahrzeuge, dass ein Überholmanöver so schnell wie möglich stattfindet. Der zweite Grenzfall ist, dass das vordere Fahrzeug versucht, das hintere am Überholen zu hindern. Für die Bewertung dieser beiden Szenarien ergeben sich zwei Betrachtungsweisen.

Der Überholende will in beiden Fällen so schnell wie möglich am anderen Fahrzeug vorbeifahren, das ihn an der schnellen Weiterfahrt hindert. Jede Zeit, die er also verliert

im Vergleich zu einer Fahrt ohne ein anderes Objekt auf der Strecke, ist in diesem Fall als negativ zu bewerten. Die untersuchte Metrik ist in diesem Fall die Gesamtzeit des Manövers abzüglich der Fahrzeit der identischen Strecke ohne das andere Fahrzeug.

Für das überholte Fahrzeug ist die Betrachtung für beide Grenzfälle unterschiedlich. Wenn versucht wird, das andere Fahrzeug am Überholen zu hindern, gilt, je mehr Zeit das überholende Fahrzeug beim Überholversuch verliert, desto besser. Ist es nicht notwendig, den Überholer am Überholen zu hindern, ist die gleiche Metrik wie für den Überholer anzusetzen, wie viel Zeit verliert das überholte Fahrzeug, um den Überholer vorbeizulassen.

5. Identifikation des Forschungsbedarfs

Die Bewertung von technischen Systemen richtet sich im allgemeinen Fall nach Anforderungslisten. Ziel ist, dass das System in den Anforderungen präzise definiert ist, sodass, wenn alle Anforderungen abgeprüft sind, das System als “den Anforderungen entsprechend“ klassifiziert wird. Allerdings gibt die Validierung damit zunächst keine Aussage darüber, ob ein System auch als “gut“ zu klassifizieren ist. Das Benchmarking, das aufgrund der fehlenden Anforderungen und der Interaktion der Rennteilnehmer im Rahmen dieser Arbeit entwickelt wird, teilt sich gemäß der Funktion eines Benchmarkings in zwei wesentliche Teile, die qualitative Bewertung mit dem Ziel, Stärken und Schwächen an einem Algorithmus zu identifizieren und den quantitativen Vergleich von mehreren Algorithmen.

Für die Analyse der grundsätzlichen Funktion eines Verhaltensalgorithmus ergeben sich folgende zwei Aufgaben mit steigender Komplexität.

5.1. Verhaltensbewertung

Der Verhaltensalgorithmus ist dafür verantwortlich, das Fahrzeug unabhängig von anderen Fahrzeugen so schnell wie möglich auf der Rennstrecke vorwärts zu bewegen. Die zweite Stufe ist, dass der Algorithmus in der Lage ist, auch in Anwesenheit von statischen bzw. dynamischen Objekten eine möglichst kurze Rundenzeit zu gewährleisten und damit auch langsamere Fahrzeuge zu überholen.

Die Analyse des momentanen Stands der Forschung auf dem Gebiet der Analyse von Fahralgorithmen zeigt, dass der Fokus vorrangig auf der ersten Aufgabe liegt, die Geschwindigkeit zu beurteilen. Metriken, die hier aus der Analyse von menschlichen Fahrern abgeleiteten wurden, bieten die Möglichkeit, diese Aufgabe umfassend zu analysieren. Die Interaktion zwischen verschiedenen Rennteilnehmern wird bislang nur in wenigen Veröffentlichungen thematisiert. Die vorhandenen Veröffentlichungen geben häufig ein definiertes Szenario vor, ohne aber konkret auf die Motivation hinter dem Szenario einzugehen. In den meisten Fällen sind die Randbedingungen der Tests weder für andere Leser/Entwickler reproduzierbar, noch werden die präsentierten Ergebnisse mit dem Zweck des Szenarios in Relation gesetzt und deren Bedeutung diskutiert, da die Implementierungen häufig auf dedizierte Entwicklungsumgebungen optimiert wurden. Die Ergebnisse haben damit häufig nur eine sehr eingeschränkte Aussagekraft für den eigentlichen Verwendungszweck. Eine quantitative Vergleichbarkeit ist somit nicht gegeben.

Die häufige Betrachtung von synthetischen Reward-Funktionen bei der Analyse von ML-Algorithmen ist an dieser Stelle ähnlich einzuordnen, da das Design der Reward-

Funktion auf die Funktion der zugrunde liegenden Algorithmen und nicht für eine Bewertung von Menschen optimiert ist.[77] Die Aussagekraft von erzielten Rewards besitzt daher nur in Ausnahmefällen Aussagekraft für die Performance in einem Zielszenario. Dies gilt insbesondere dann, wenn ein Problem nicht in einem einzelnen Szenario, sondern in einem Curriculum erlernt wird, um das Lernen komplexer Vorgänge für einen Algorithmus zu erleichtern.[78] Die Ergebnisse, die aus einer solchen Evaluierung resultieren, sind daher ebenfalls nur wenig aussagekräftig für einen allgemeinen Anwendungsfall und meist beschränkt auf die spezifischen Bedingungen der Untersuchung. Eine genaue Analyse der eigenen Algorithmen wird dadurch auch für die Entwickler erschwert, da die Algorithmen unter eingeschränkten Bedingungen getestet werden.

In Bezug auf die Analyse von Verhaltensalgorithmen stellt sich daher die Forschungsfrage:

FF 1: Welchen Einfluss besitzen die Randbedingungen eines Szenarios auf die Aussagekraft der Evaluation von Verhaltensalgorithmen für autonome Rennfahrzeuge in Bezug auf die qualitative Analyse der Algorithmen und die quantitative Vergleichbarkeit?

5.2. Rankingmethoden

Rankingmethoden, die bisher in der Literatur präsentiert wurden, beschäftigen sich mit der Problemstellung, einen bestimmten Pool von verschiedenen Spielern in eine leicht verständliche Rangfolge zu bringen. Dabei lösen sie zwei grundsätzlich verschiedene Problemstellungen. Das eine ist die Berechnung oder Approximation der Payoff-Matrix $\boldsymbol{p}$, der Wahrscheinlichkeit von jedem Spieler n gegen jeden beliebigen anderen Spieler m zu gewinnen. An dieser Stelle existieren bereits erfolgreiche Ansätze wie Elo, Glicko oder TrueSkill, die diese Aufgabe für sehr große Pools an menschlichen Spielern ($>$ 1.000.000) lösen, indem die Transitivität der Spielerfähigkeiten angenommen wird. Eine Behandlung intransitiver Interaktionen ist an dieser Stelle ebenfalls durch den Einsatz von mehrdimensionalen Interferenz-Matrizen wie bei mElo möglich, aber bislang nicht für eine große Anzahl Spieler evaluiert.[79] Dieser Ansatz bietet auch die Möglichkeit, nicht nur Spieler-vs.-Spieler-Interaktionen, sondern auch Spieler-vs.-Aufgabe-Szenarien zu modellieren und so die Schwierigkeit von Aufgaben bezogen auf verschiedene Fähigkeiten zu ermitteln. Das Ranking wird zu diesem Zweck erweitert, sodass neben den Spielern ebenfalls jeder Aufgabe einen Rang zuweist und einen Payoff berechnet. In diesem Fall beschreibt die Payoff-Matrix den erwarteten Erfolg eines Spielers in einem bestimmten Szenario.

[77] Vgl. Knox, W. B. et al.: Reward (Mis)design for Autonomous Driving (2021).

[78] Vgl. Song, Y. et al.: Autonomous Overtaking in Gran Turismo Sport Using Curriculum RL (2021).

[79] Vgl. Klimenko, A. Y.: Intransitivity in Theory and in the Real World (2015).

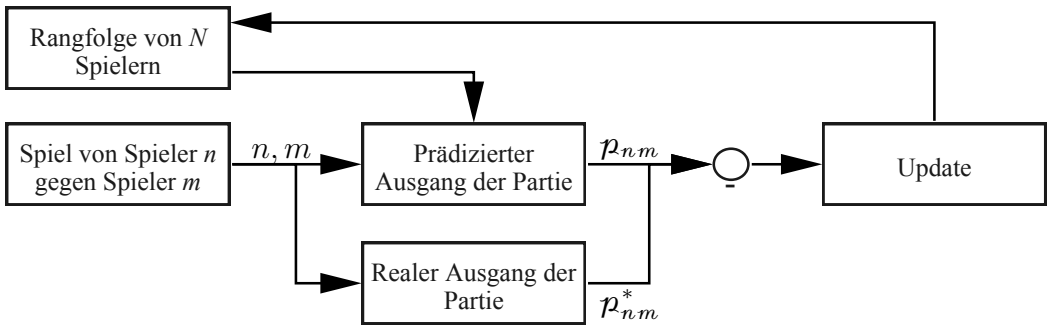


Abbildung 5-1.: Prinzip der Berechnung einer Rangfolge von Elo, mElo, Glicko und TrueSkill

Die Payoff-Matrix ist dabei die Basis für alle Methoden, um das Ranking upzudaten. Für die Erstellung des Rankings wird für die beschriebenen Methoden der in Abbildung 5-1 dargestellte abstrahierte Prozess angewandt. Spielt ein Spieler n gegen einen anderen Spieler oder eine Aufgabe m wird mithilfe des momentanen Rankings der erwartete Erfolg p_{nm} von Spieler n berechnet. Handelt es sich dabei um eine Partie gegen einen Spieler, entspricht p_{nm} der Gewinnwahrscheinlichkeit von Spieler n. Entspricht die Partie einem Test im Rahmen einer Aufgabe, repräsentiert p_{nm} das erwartete Testresultat. Das Update erfolgt auf Basis der Differenz des erwarteten Payoffs p_{nm} und dem realen Ergebnis der Partie p^*_{nm}. Ein Ranking von intransitiven Relationen konvergiert daher nur korrekt, wenn das Ranking-Verfahren auch in der Lage ist, die nicht transitiven Relationen darzustellen. Die Folgen davon sind in Abbildung 5-1 abgebildet. Dort wurde der Rang für die Payoff-Matrix des bekannten Spiels Schere-Stein-Papier (SSP) berechnet. Die Abbildung zeigt den berechneten Rang über einer Anzahl von 250 Spielen für ein Elo-Ranking und ein mElo-Ranking. Die genaue Funktionsweise der Algorithmen wird in Abschnitt 6.6 im Details vorgestellt. Das einfache Elo-Verfahren ist nicht in der Lage, die intransitiven Relationen abzubilden, wodurch die errechneten Ränge nicht konvergieren, sondern zufällig schwingen. Für mElo konvergieren die Werte dagegen und bewegen sich ebenfalls in einem kleineren Intervall. Dabei ist anzumerken, dass für mElo die berechneten Ränge noch durch eine Interaktionsmatrix ergänzt werden. Dadurch ergibt sich, dass, obwohl die Gewinnwahrscheinlichkeit für alle drei Strategien korrekt berechnet wird, die Ränge nicht vollständig identisch sind. Die resultierenden Payoff-Matrizen für das Problem sind in Tabelle 5-1 abgebildet. Während mElo die Dynamik der Spielstrategien korrekt abbildet, zeigt Elo eine Schätzung der Matrix, die ohne intransitive Relation stark von der echten Matrix abweicht. Im Falle von intransitiven Spieler-Spieler- oder Spieler-Aufgabe-Relationen ist daher zwingend ein Ranking-Verfahren notwendig, das in der Lage ist, diese Relationen korrekt abzubilden.

Die zweite wesentliche Herausforderung ist die Ermittlung einer Rangfolge unter Berücksichtigung der Payoff-Matrix $\boldsymbol{p}$. Wird die Transitivität der Spielerfähigkeiten angenommen, ist diese Aufgabe als einfach anzusehen, da es lediglich notwendig ist,

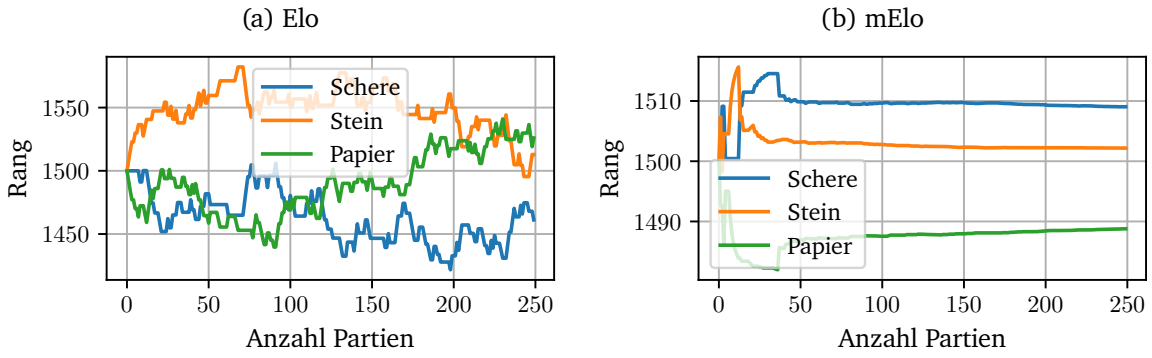


Abbildung 5-2.: Ermittelte Rangfolge für Schere-Stein-Papier Szenario über Zeit (a) mittels Elo Verfahren, (b) mittels mElo Verfahren

Tabelle 5-1.: (a) Echte Payoff-Matrix für SSP, (b) über Elo-Verfahren errechnete Payoff-Matrix , (c) über mELO-Verfahren errechnete Payoff-Matrix

	Schere	Stein	Papier
Schere	0,5	0	1
Stein	1	0,5	0
Papier	0	1	0,5

(a)

	Schere	Stein	Papier
Schere	0,5	0,55	0,58
Stein	0,45	0,5	0,53
Papier	0,42	0,47	0,5

(b)

	Schere	Stein	Papier
Schere	0,5	0	1
Stein	1	0,5	0
Papier	0	1	0,5

(c)

die einzelnen Wahrscheinlichkeiten in $\boldsymbol{p}$ zu sortieren. Im Falle von Elo, Glicko oder TrueSkill wird daher von der Methode selbst nur ein Ranking errechnet und die Payoff-Matrix daraus ermittelt. Für die Annahme von intransitiven Interaktionen zwischen verschiedenen Spielern oder Spielstrategien sind bislang entwickelte Methoden wie α Rank oder das maxent NG in der Lage, aus einer voll besetzten Payoff-Matrix eine Rangfolge zu bestimmen, indem eine optimale statistische Verteilung der Spielstrategien für ein Metaspiel gebildet wird.

Der Algorithmus versucht, eine ideale Spielstrategie für das abstrahierte Spiel zu generieren, bei dem seine Aufgabe darin besteht, aus der Anzahl vorhandener Spieler bzw. Aufgaben die richtige auszuwählen, sodass Spieler n bzw. m gewinnt. Übertragen auf das einfache SSP Beispiel bedeutet das, dass der Algorithmus errechnet, wie häufig er welche Strategie (Schere, Stein oder Papier) wählt, um eine beliebige Strategie des

Gegners zu schlagen. Für das einfache SSP Beispiel ist diese Verteilung $\left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$, da alle Strategien gleichwertig sind.

Auf diese Weise ist möglich, sowohl die Stärke einzelner Spieler als auch die Schwierigkeit von Aufgaben zu evaluieren und zu berechnen. Eine Spielstrategie, ein Agent oder eine Aufgabe, die bzw. der in der Verteilung stärker gewichtet ist, ist dementsprechend als stärker anzusehen, da sie häufiger zum Sieg führt. Das mElo-Verfahren bildet zwischen den beiden Ansätzen das Bindeglied. Es ermöglicht die Berechnung eines Rankings auch auf Basis von unvollständigen Informationen mit dem Risiko, dass ein fehlender Eintrag in der Payoff-Matrix das Ranking-Ergebnis entscheidend beeinflusst.

Daher lässt sich der Grundzusammenhang bestimmen, dass je stärker ausgeprägt die intransitiven Relationen für Spiele oder Tests sind, desto höher der Aufwand, der in die Bestimmung der Payoff-Matrix fließt. Daher besteht zunächst die Frage:

FF 2: Ist die Annahme der Transitivität für die Interaktion von autonomen Rennfahrzeugen valide?

Für ein Benchmarking, das für die Bewertung des Verhaltens von mehreren Systemen und deren Interaktion untereinander erstellt wird, ergibt sich dabei folgendes Problem. Klassische Benchmarkings evaluieren die absolute Performance eines Systems und benötigen keine Interaktion verschiedenen Vergleichspartner. Ist eine Interaktion notwendig, benötigt die Instanz, die versucht, ein System im Rahmen des Benchmarkings zu testen, unter Annahme der Intransitivität auch Zugriff auf alle Algorithmen, die als Testobjekte in den Szenarien verwendet werden. Davon ist in der Praxis allerdings in einem kompetitiven Umfeld nicht auszugehen. Die Verfügbarkeit einer voll besetzten Payoff-Matrix ist aus diesem Grund nur unter der Bedingung eine valide Annahme, dass ein fest vorgegebener Satz an Tests gegen frei verfügbare Referenzalgorithmen der Maßstab für das Benchmarking ist und nicht von anderen Instanzen entwickelte Algorithmen. Dadurch ergibt sich folgende weitere Forschungsfrage:

FF 3: Wie beeinflusst eine nicht voll-besetzte Payoff-Matrix den Vergleich von mehreren Agenten unter der Annahme von intransitiver Interaktion der Agenten und damit die Möglichkeit, ein Ranking mehrerer Agenten zu bilden?

Darunter verstehen sich nicht nur die Auswirkungen auf die Payoff-Matrix $\boldsymbol{p}$ und den Ranking-Prozess, wenn einzelne Einträge von $\boldsymbol{p}$ fehlen oder ein Test nicht möglich ist. Eine Schätzung von einzelnen Einträgen in der Payoff-Matrix würde an dieser Stelle ermöglichen, dass ein Benchmarking einen Agenten nicht nur gegen ein bestimmtes Set von Aufgaben testet. Damit würde ebenfalls möglich, ein Ergebnis gegen einen unbekannten Agenten zu schätzen. Hier besteht allerdings die Voraussetzung, dass

eine Menge an jedem zugänglichen Vergleichstests vorhandenen ist, die eine Übertragbarkeit auf andere Agenten zulässt. Je höher der Grad der Übertragbarkeit der Referenz-Szenarien ist, desto größer ist auch die Wahrscheinlichkeit, dass die nicht zu testenden Einträge in p anhand der vorhandenen Einträge angenähert werden können, indem die Interaktionen der Agenten in den Referenzszenarien analysiert werden.

Die Erstellung einer Rangfolge von Algorithmen beinhaltet eine Reduzierung eines komplexen Sachverhaltes auf eine einfache, verständliche Metrik. Das verfolgte Ziel ist zu ermöglichen, dass die Performance oder Güte eines Objektes auf einer ordinalen Skala sortierbar ist. Wie die Voraussetzungen für ein aussagekräftiges Benchmarking definieren, spielen die Randbedingungen, unter denen getestet wird, eine große Rolle. Wenn einzelne Randbedingungen einen bestimmten Algorithmus bevorzugen, ist ein ebenbürtiger Vergleich nicht mehr möglich. Mit Blick auf die Vielfältigkeit der Szenarien und die Diskussion intransitiver Relationen stellt sich zusätzlich folgende Frage:

FF 4: Welchen Einfluss hat die Annahme von intransitiven Relationen auf die Aussagekraft eines quantitativen Rankings von Verhaltensalgorithmen?

5.3. Anforderungen an die Benchmarking-Tests

Nach den vorangegangenen Abschnitten lassen sich drei wesentliche Anforderungen an das Benchmarking formulieren. Es wird gefordert, dass durch das Benchmarking folgende Punkte ermöglicht werden.

- Qualitative Aussage über Stärken und Schwächen
- Quantitative Aussage über verlorene Zeit und Ereignis-Häufigkeiten auf der Strecke bei Überholmanövern
- Direkter Vergleich von zwei oder mehr Algorithmen in Form einer Rangfolge

Eine **qualitative Aussage über Stärken und Schwächen** ist die niedrigste Form der Aussage des Benchmarkings. Sie ermöglicht, systematische Schwachstellen und Verbesserungspotenziale zu identifizieren, aber auch Stärken einer bestimmten Funktionsweise genau zu erkennen. Ein Zusammenhang zu einer konkreten Strecke ist an dieser Stelle zwar wünschenswert, aber nicht notwendig, um diese Information zur Verfügung zu stellen.

Im nächsten Schritt ist Aufgabe des Benchmarkings, eine **quantitative Aussage über verlorene Zeit und Ereignis-Häufigkeiten auf der Strecke** zu liefern. Hier geht es darum, zu bestimmen, wie viel Zeit ein Algorithmus für ein Überholmanöver auf einem bestimmten Streckenabschnitt benötigt, ob bzw. wie häufig auf den Streckenabschnitten Unfälle passieren durch den Algorithmus oder ob bzw. wie häufig ein

Überholen überhaupt gelingt. Dabei wird eine hohe Aussagekraft von den generierten Informationen in Bezug auf die eigentliche Aufgabe gefordert, autonom Rennen zu fahren. Auf diese Weise ist möglich, die Überholfähigkeit der Algorithmen objektiv zu bewerten.

Für beide bisher genannten Punkte geht es maßgeblich um das Überholen von Fahrzeugen. Daher wird zur Erfüllung dieser beiden Anforderungen ein Referenzalgorithmus definiert, der eine so geringe Wechselwirkung wie möglich zwischen überholendem Fahrzeug und überholtem Fahrzeug gestattet. Ein **direkter Vergleich von zwei Algorithmen** ist allerdings nur gegen einen voll funktionsfähigen Vergleichspartner möglich, da das Verhindern eines Überholmanövers nur unter der Betrachtung von Wechselwirkungen möglich ist, da (a) notwendig ist, dass das hintere Fahrzeug auf das vorne fahrende Fahrzeug reagiert, und (b) dem vorne fahrenden Fahrzeug nur möglich ist, ein Überholen zu verhindern, wenn es auf die Aktionen des hinten fahrenden Fahrzeugs reagiert. Im direkten Vergleich ist möglich, sowohl das Überholverhalten als auch die Fähigkeit zum Abwehren eines anderen Fahrzeugs zu beurteilen. Ebenfalls ist bei einem direkten Vergleich die Ermittlung eines überlegenen Algorithmus möglich, da die verwendete Referenz der jeweils andere Algorithmus ist. Das gewünschte Resultat des direkten Vergleichs ist eine Rangfolge der getesteten Algorithmen.

Die Ermittlung der Rangfolge bedarf eines speziellen Versuchsdesigns. Den Grundwerten eines guten Benchmarkings in Abschnitt 4.1.2 folgend, besitzt ein aussagekräftiges Benchmarking die Eigenschaft, dass durch das Versuchsdesign kein Algorithmus einen grundlegenden Vorteil erhält. Eine zusätzliche Anforderung ist daher, dass die Hyperparameter der Tests keine Beeinflussung der ermittelten Rangfolge verursachen.

6. Benchmarking

Die Grundfragestellung dieser Arbeit nach der Vergleichbarkeit zwischen einer Vielzahl von verschiedenen Verhaltensalgorithmen ist nach Forschungsfrage 1 zunächst eine Frage der Randbedingungen, unter denen ein Algorithmus getestet wird. Im Folgenden werden daher die verschiedenen Einflussparameter eines Szenarios im Details diskutiert. Besonderer Fokus liegt an dieser Stelle auf der Aussagekraft auf zum einen eine qualitative Bewertung bzw. eine Analyse der Stärken und Schwächen von Verhaltensalgorithmen, zum anderen auf die Möglichkeit, eine quantitative Aussage bezüglich verlorener Zeit oder anderer Algorithmen in Form eines Rankings zu treffen.

Die Analyse bestehender Bewertungsansätze von Verhaltensalgorithmen in Abschnitt 3.1 zeigt an dieser Stelle die Wichtigkeit des Vergleichsmaßstabs auf. Die Auswahl von passenden Vergleichsobjekten für die Analyse wird daher in Abschnitt 6.1 näher beleuchtet. Der Begriffsdefinition des Szenarios von Ulbrich *et al.*[80] folgend ist die nächste Komponente eines Szenarios die Szenerie und damit die Strecke, auf der die Algorithmen getestet werden. Abschnitt 6.2 gibt einen Überblick über möglichen Varianten für Teststrecken und deren Einfluss auf die zu erwartenden Testergebnisse. Abschnitt 6.3 befasst sich mit der Parametrierung sowohl des Ego-Fahrzeugs als auch der dynamischen Objekte und komplettiert damit die Beschreibung der Szene. Die auszuwertenden Metriken in Bezug auf Überhol- und Abwehrmanöver werden im Anschluss daran in Abschnitt 6.4 diskutiert.

Abschnitt 6.5 stellt das in dieser Arbeit entwickelte Verfahren zum Ranking mehrerer Algorithmen vor. Zu diesem Zweck wird zuerst Forschungsfrage 2 nach der Transitivität der Interaktion von autonomen Rennfahrzeugen beantwortet. Daran anknüpfend beschreibt Abschnitt 6.6 das Verfahren zur Berechnung des Rankings und geht insbesondere auf den Umgang mit fehlenden Einträgen in der Payoff-Matrix und den damit verbundenen Implikationen für die definierten Tests ein.

6.1. Testreferenz

Das Vergleichsobjekt, gegen das der Algorithmus getestet wird, bildet den ersten Bestandteil der Szene und bestimmt für das in dieser Arbeit entwickelte Benchmarking die Aussagekraft, die durch die durchgeführten Tests generiert wird, indem es den Maßstab für den Vergleich bildet. Dieser ist ein Algorithmus mit unterschiedlich ausgeprägtem Funktionsumfang und Fähigkeit für intelligentes Verhalten, gegen den getestet wird. Beide Algorithmen in einem Szenario beeinflussen sich gegenseitig und damit entwickeln sich auch die Szenarien, in denen gefahren wird, nicht identisch für

[80] Ulbrich, S. et al.: Begriffsdefinitionen für das automatisierte Fahren (2015).

unterschiedliche zu untersuchende Algorithmen. Um einen objektiven Vergleich zu ermöglichen, werden daher Objekte genutzt, die eine möglichst geringe Komplexität und Interaktion mit dem Fahrzeug aufweisen. Gleichzeitig bestimmt der Funktionsumfang des Vergleichsobjektes die Übertragbarkeit der Testergebnisse auf einen direkten Vergleich zweier Algorithmen, da das Verhalten des Maßstabs als umso repräsentativer betrachtet wird, je ähnlicher Funktionsumfang und Verhalten denen eines anderen Agenten sind. Die Vergleichsbasis als zu überholendes Objekt wird daher ein Algorithmus verwendet, der möglichst wenig Interaktion mit dem Fahrzeug hinter sich erlaubt. Erst für den direkten Vergleich von zwei Algorithmen wird direkt der andere Algorithmus verwendet, der eine komplexe Interaktion zwischen beiden Algorithmen erzeugt.

Für den Test der Verteidigung gegen ein anderes Fahrzeug ist eine Betrachtung mit minimaler Rückkopplung allerdings nicht möglich. Das liegt u. a. daran, dass auf der Rennstrecke das vordere Fahrzeug stets Vorfahrt besitzt. Der Agent geht demnach davon aus, dass, solange das zu überholende Fahrzeug vorne fährt, es nicht notwendig ist, auf das dahinter fahrende Fahrzeug zu reagieren. Ist es aber die Aufgabe des Agenten, zu versuchen, das dynamische Objekt am Überholen zu hindern, ist das Objekt gezwungen, auf die Bewegung des vorne fahrenden Fahrzeugs zu reagieren. Als Vergleichsobjekt ist daher nur bedingt ein Ersatzobjekt für den Test verwendbar. Ein Test besitzt nur gegen einen bestimmten Algorithmus die volle Aussagekraft.

Zur Verdeutlichung des Einflusses des Referenzobjektes auf die erzielten Ergebnisse wurde ein Test mit drei unterschiedlichen Fahralgorithmen durchgeführt. Der erste Agent fährt dabei völlig ohne Interaktion zu dem anderen Objekt. Die beiden anderen Agenten wurden im Rahmen dieser Arbeit entwickelt und werden in Kapitel 7 im Details vorgestellt. Der aus der Literatur nach implementierte Referenzalgorithmus berücksichtigt zu jedem Zeitpunkt die Prädiktion des vorausfahrenden Fahrzeugs und versucht Kollisionen zu vermeiden. Der im Rahmen dieser Arbeit entwickelte LCADM-Algorithmus (engl. Linear Constraints with Advanced Decision Making) reagiert nur auf das gegnerische Fahrzeug, wenn es durch eine laterale Bewegung zu einer Kollision kommen würde, weil beide Fahrzeuge sich bezogen auf die Strecke in longitudinaler Richtung näher als eine Fahrzeuglänge sind. Als Testbedingung wird die Grand-Prix-Strecke des Hockenheimrings gewählt. Die beiden Fahrzeuge starten mit jeweils $20\,\mathrm{m}$ Abstand und einer Geschwindigkeit von $10\,\frac{\mathrm{m}}{\mathrm{s}}$. Das hintere Fahrzeug ist so parametriert, dass es eine geringere Rundenzeit besitzt als das vorne fahrende Fahrzeug. Eine detaillierte Auflistung der Randbedingungen findet sich in Anhang D.2. Jeder der drei Agenten wurde bei dem Test jeweils einmal als hinteres Fahrzeug und einmal als vorderes Fahrzeug eingesetzt. Als Vorgriff auf die Modellbeschreibung in Abschnitt 7.4 wird an dieser Stelle darauf hingewiesen, dass das in dieser Arbeit verwendete Punktmassenmodell durch drei wesentliche Parameter charakterisiert wird.

Die maximale laterale Beschleunigung $a_{\text{lat,max}}$, die maximal mögliche Bremsverzögerung D_{max} und die maximale Vortriebsleistung P_{max}. Das überholende Fahrzeug ist für alle Versuche konstant parametrisiert zu

$$P_{\text{max}} = 400\,\text{kW}, a_{\text{lat,max}} = 20\,\frac{\text{m}}{\text{s}^2}, D_{\text{max}} = 20\,\frac{\text{m}}{\text{s}^2}.$$

Die Erfolgs-, Unfall und Misserfolgsraten dieses Versuchs sind in Abbildung 6-1 dargestellt. Zur Ermittlung der Raten wurden jeweils 500 Versuche mit Startpositionen durchgeführt, die zufällig[81] entlang der Strecke verteilt sind. Auf diese Weise wird der Einfluss einer einzelnen Startbedingung gemindert. Die dargestellten Raten sind das arithmetische Mittel über alle Versuche. Die Ergebnisse zeigen deutlich, dass besonders die Verwendung des Agenten ohne Interaktion ein stark polarisiertes Ergebnis hervorruft. Versucht dieser Agent zu überholen, ohne dabei das vorausfahrende Fahrzeug zu berücksichtigen, kommt es mit dem LCADM-Algorithmus und sich selbst zu über $95\,\%$ zu einer Kollision, da der Agent lediglich schneller auf der Ideallinie fährt, aber keine Anpassung seiner geplanten Trajektorie vornimmt, um eine Kollision zu vermeiden. Gegen den Referenzalgorithmus hingegen überholt der Agent zu $84\,\%$ erfolgreich, da dieser durch die Annäherung des schnelleren Agenten von hinten sich aus dem Weg bewegt, um die Kollision zu verhindern. Die Abbildung zeigt aber auch, dass die Verwendung des Agenten ohne Rückkopplung als zu überholendes Objekt nicht für einen ebenbürtigen Vergleich nutzbar ist, da sich vor allem für den Referenzalgorithmus eine enorme Steigerung der Unfallrate zeigt. Ein genauer Blick in die Daten zeigt, dass es durch die Nicht-Beachtung des anderen Fahrzeugs häufig zu Kollisionen kommt, wenn das Fahrzeug sich während des Überholmanövers auf der Ideallinie aufhält.

Für die Bewertung in dieser Arbeit wird als dynamisches Objekt der LCADM-Algorithmus als passives zu überholendes Fahrzeug verwendet. Dieser wird bei einem schnelleren, sich von hinten nähernden Fahrzeug nicht beeinflusst, solange das andere Fahrzeug sich hinter ihm selbst befindet. Erst sobald beide Fahrzeuge sich in longitudinaler Richtung so nahe sind, dass eine Kollision möglich ist, versucht der Algorithmus entsprechend dem anderen Fahrzeug ausreichend Platz zu lassen, ohne dabei Rücksicht auf den errechneten Plan des anderen Algorithmus zu nehmen. Auf diese Weise wird die Interaktion zwischen beiden Teilnehmern minimal gehalten.

Für den Fall des Verteidigens gegen einen Gegner ist nur ein Algorithmus mit einer entsprechenden Interaktion als Vergleichsobjekt nutzbar. Ein einzelner Test gegen ein

[81] Zufällig bedeutet in diesem Kontext, dass vor jedem Szenario die Startposition entlang der Strecke und die laterale Position auf der Strecke (vgl. Abbildung 6-2 rechte Seite) zufällig gewählt werden. Es besteht keine Verbindung zwischen den Startpositionen der unterschiedlichen Variationen.

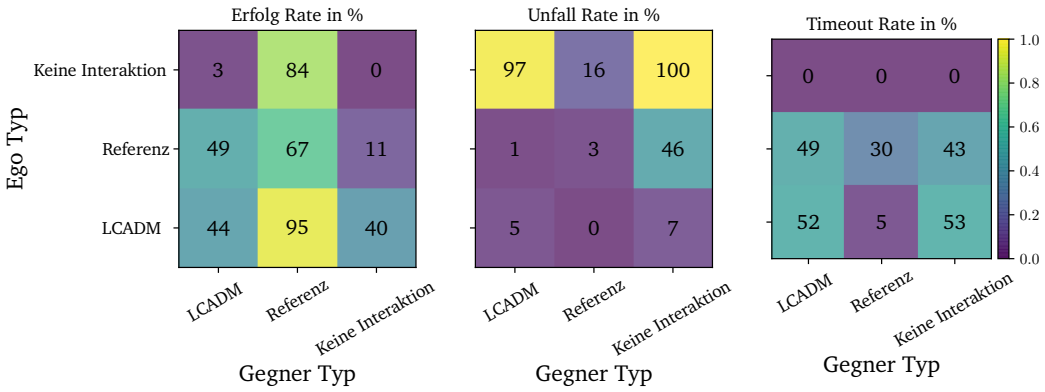


Abbildung 6-1.: Ergebnisse für Ego- und Gegner-Algorithmus-Variation auf Streckenlayout Hockenheim, Parametrisierung für: Gegner-Fahrzeug $P_{\text{max}} = 350\,\text{kW}, a_{\text{lat,max}} = 17\,\frac{\text{m}}{\text{s}^2}, D_{\text{max}} = 17\,\frac{\text{m}}{\text{s}^2}$, Ego-Fahrzeug $P_{\text{max}} = 400\,\text{kW}, a_{\text{lat,max}} = 20\,\frac{\text{m}}{\text{s}^2}, D_{\text{max}} = 20\,\frac{\text{m}}{\text{s}^2}$. Ziel in dem Versuch ist, dass das Ego-Fahrzeug das Gegner-Fahrzeug überholt. Die Abbildung zeigt, wie viele der 500 durchgeführten Tests pro Variation erfolgreich (Ego-Fahrzeug überholt Gegner-Fahrzeug), nicht erfolgreich (Gegner-Fahrzeug bleibt vor Ego-Fahrzeug) oder mit einer Kollision der beiden Fahrzeuge enden.

anderes Vergleichsobjekt besitzt demnach nur eine begrenzte Aussagekraft und Übertragbarkeit auf den Test mit einem anderen Algorithmus. Das Ziel des hier vorgestellten Benchmarks ist daher, mehrere Algorithmen als Vergleichsobjekte zur Verfügung zu stellen, um auf diese Weise die Aussagekraft der Tests als Ganzes durch die Verwendung eines Testkollektivs zu erhöhen.

6.2. Verwendete Strecken

Die Szenerie und damit das Streckenlayout ist für Szenarien die nächste entscheidende Komponente. Je nach Layout ist möglich, dass verschiedene Situationen zwischen den Rennteilnehmern entstehen, da verschiedene Kurven jeweils eine unterschiedliche Ideallinie besitzen. Aber auch die Kombination der Kurven spielt in diesem Kontext eine Rolle, da bspw. eine Haarnadelkurve nach einer langen Geraden mit einem starken Bremsmanöver angefahren wird. Ist auf der Strecke vor der Haarnadelkurve eine andere Haarnadelkurve, ist das Verhalten entsprechend unterschiedlich. Für die Layouts, die im Benchmark abgetestet werden, besteht für eine hohe Aussagekraft des Vergleichs die Forderung, dass sie so ähnlich wie möglich den Strecken sind, die auch in der vorgesehenen Rennserie des Algorithmus gefahren werden. Auf der anderen Seite, um eine möglichst hohe Aussagekraft durch eine hohe Parameterabdeckung verschiedener Szenerien zu erhalten, ist ein möglichst hoher Grad an Zufälligkeit im Testdesign notwendig.

Es ist demnach möglich, die getesteten Strecken komplett zufällig zu generieren, oder es werden die realen Streckenlayouts oder Streckenabschnitte genutzt, die in einer Rennserie befahren werden. Alternativ dazu besteht zusätzlich die Möglichkeit, mehrere Streckenabschnitte auszuwählen, die eine repräsentative Auswahl aus verschiedenen

Kurven und Kurvenkombinationen beinhalten, die für eine Evaluation genutzt werden. Dabei besitzen Rennstrecken grundsätzlich das Problem, dass sie von vornherein eine ungleiche Anzahl an Kurven nach links bzw. rechts besitzen. Zusätzlich ist die Richtung eins Großteil der Rennstrecken, die bspw. in der Formel 1 befahren werden, im Uhrzeigersinn (54/73), sodass reale Strecken einen deutlichen Bias hin zu Rechtskurven besitzen.[82]

Die möglichen Varianten der Streckenauswahl werden im Folgenden auf Eignung für die Erfüllung der genannten Anforderungen für die quantitative und qualitative Evaluierung der Algorithmen untersucht. Der Test auf einer einzelnen Strecke, der Test auf mehreren Streckenabschnitten, die eine Menge an repräsentativen Kurven darstellen und der Test auf komplett zufällig generierten Strecken.

Der Test auf zufällig generierten Strecken bieten den Vorteil, dass keine Bindung an vorgegebene Layouts besteht und dadurch eine Aufdeckung negativer Verhaltensweisen potenziell einfacher möglich ist, indem ein sehr weiter Parameterraum untersucht wird. Diese Art des Tests wird daher hinsichtlich des ausgenutzten Parameterraumes bevorzugt, da auf diese Weise das Risiko minimiert wird, dass ein Algorithmus als "gut" eingestuft wird, er aber tatsächlich nur auf eine einzelne Strecke optimiert wurde. Allerdings ist das zufällige Layout und auch die Reihenfolge der Kurven in diesem Fall nicht korreliert zu realen Strecken. Um quantitative Aussagen für die Algorithmen zu ermöglichen, ist daher der Nachweis zu erbringen, dass die Reihenfolge von Kurvenabschnitten keine Rolle spielt für (a) die Zeit, die durch ein Überholmanöver verloren wurde im Vergleich zur unbeeinflussten Fahrt auf der Strecke und (b) die Wahrscheinlichkeit für ein Überholmanöver per se.

Beim Test auf mehreren repräsentativen Strecken gilt derselbe Sachverhalt. Die Tests weisen in dem Fall Kurven auf, die potenziell repräsentativ für die Aufgabe der Fahrt auf einer Rennstrecke sind. Die Reihenfolge, in der die Kurven abgebildet sind, ist aber nicht notwendigerweise abgebildet bzw. es werden nicht die vollständigen Strecken genutzt.

Die Eignung von zufälligen Streckenlayouts und der Fahrt auf repräsentativen Strecken für quantitative Aussagen im Rahmen eines Benchmarkings wird im Folgenden untersucht. Dafür wird der Frage nachgegangen, ob das Streckenlayout und damit verbunden die Reihenfolge, in der Kurven in einer Strecke auftreten, Einfluss auf die quantitative und zeitbezogene Auswertung von Überholszenarien nimmt.

Zur Klärung dieser Frage wurden mehrere Versuche mit unterschiedlichen Variationen an Parametern für ein Gegner-Fahrzeug durchgeführt, das in einem zufällig parametrierten Trainingsszenario fährt. Als Algorithmus für das Ego-Fahrzeug wurden sowohl

[82] Tremayne, D.: The concise encyclopedia of Formula One (2000), S. 70ff.

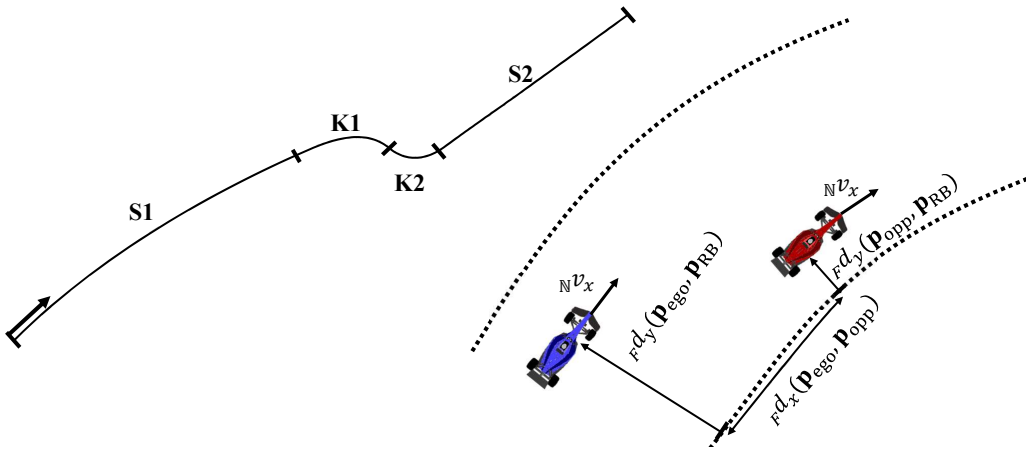


Abbildung 6-2.: Anfangsbedingungen für ein zufällig generiertes Szenario. Dabei werden die Längen und Krümmungen der Segmente unter Berücksichtigung der Parametergrenzen in Tabelle 6-1 zufällig ermittelt.
Links: Streckenlayout, Rechts: Gesetzte Anfangsbedingungen

der LCADM-Algorithmus genutzt als auch die implementierte Referenz. Das dafür erstellte Layout dient als Grundlage für eine Vielzahl an Tests im Rahmen dieser Arbeit und wird deshalb im Folgenden vorgestellt.

6.2.1. Zufällig generiertes Streckenlayout

Das genutzte Layout besteht aus vier wesentlichen Teilen, die in Abbildung 6-2 links visualisiert sind. Die detaillierte Parametrisierung der Segmente ist in Tabelle 6-1 zusammengefasst. Der Begriff **"zufällig generiertes Streckenlayout"** bezieht sich an dieser Stelle darauf, dass das in diesem Abschnitt vorgestellte Layout zufällig parametriert wird, wobei das Layout immer dem identischen Schema folgt. Die Strecke beginnt mit Segment S1 mit sehr geringer Krümmung und einer zufälligen Länge, gefolgt von einer Kombination aus zwei Kurven K1 und K2, die ebenfalls eine zufällige Krümmung und Mittelpunktswinkel im Bogenmaß besitzen. Abschließend folgt das Auslaufsegment S2 mit Krümmung 0 und einer festen Länge von $1000\,\mathrm{m}$.

Das leicht gekrümmte Streckensegmente S1 ist dabei definiert mit einem Kurvenradius größer als $1{,}5\,\mathrm{km}$. Damit ergibt sich die maximale Querbeschleunigung, wenn das Fahrzeug dem Segment folgt zu $5\,\frac{\mathrm{m}}{\mathrm{s}^2}$ bei einer angenommenen Maximalgeschwindigkeit von $300\,\frac{\mathrm{km}}{\mathrm{h}}$.

Alle Krümmungen der Segmente besitzen ein zufälliges Vorzeichen, die zusätzlich voneinander unabhängig sind. Damit werden von der Kombination der Kurven K1 und K2 alle üblichen Kurventypen abgebildet, da sowohl gleichsinnige als auch gegensinnige Kurvenkombinationen (Haarnadelkurven bzw. Schikanen und deren Abwandlungen) dadurch entstehen. Für die Kurven K1 und K2 wird ein Kreisbogen mit einem Mittelpunktswinkel zwischen $20\,^\circ$ und $90\,^\circ$ gewählt sowie ein Radius zwischen $12\,\mathrm{m}$ und $200\,\mathrm{m}$. Der minimale Radius begründet sich durch die gewählte Streckenbreite von

$10\,\mathrm{m}$. [83] Der maximale Radius wurde anhand realer Streckendaten ermittelt, der den Wert von $200\,\mathrm{m}$ nur in wenigen Ausnahmefällen wie dem Indianapolis Motorspeedway oder dem Autodromo Nazionale di Monza erreicht bzw. übersteigt.[84]

Die Anfangsbedingungen sind statisch definiert, wie in Abbildung 6-2 rechts dargestellt. Der longitudinale Abstand zwischen der Position des Ego-Fahrzeugs $\boldsymbol{p}_{\text{ego}}$ und des Gegner-Fahrzeugs $\boldsymbol{p}_{\text{opp}}$ projiziert entlang der x-Achse des Frenet-Koordinatensystems der Strecke ${}_{F}d_x(\boldsymbol{p}_{\text{ego}}, \boldsymbol{p}_{\text{opp}})$ ist zu $20\,\mathrm{m}$ definiert. Die Längsgeschwindigkeit beider Fahrzeuge ${}_{\mathbb{N}}v_x$ ist mit $20\,\frac{\mathrm{m}}{\mathrm{s}}$ so definiert, dass die Fahrzeuge noch Potenzial für Beschleunigung besitzen und damit die Möglichkeit auf dem ersten Geradensegment S1 zu überholen. Der laterale Abstand ${}_{F}d_x(\boldsymbol{p}_{\text{ego|opp}}, \boldsymbol{p}_{\text{RB}})$ zwischen der jeweiligen Position $\boldsymbol{p}_{\text{ego}}$ und $\boldsymbol{p}_{\text{opp}}$ der Fahrzeuge und der rechten Begrenzung $\boldsymbol{p}_{\text{RB}}$ ist für jede Wiederholung des Szenarios zufällig gewählt.

Tabelle 6-1.: Verwendete Parametrisierung für zufällig generierte Szenarien

	S1	K1	K2	S2
Länge bzw. Mittelpunktswinkel für Kreisbogen	$[100\,\mathrm{m}, 800\,\mathrm{m}]$	$[20\,^\circ, 90\,^\circ]$	$[20\,^\circ, 90\,^\circ]$	$1000\,\mathrm{m}$
Radius in m	$[1500, \infty]$	$[12, 200]$	$[12, 200]$	∞

Ein Szenario endet unter zwei verschiedenen Bedingungen:

- Wenn beide Fahrzeuge miteinander kollidieren.
- Wenn das Ego-Fahrzeug das Ende der Strecke erreicht.

Ausgewertet wird, wie viel Zeit das Ego-Fahrzeug ohne die Beeinflussung des Gegner-Fahrzeugs für dieselbe Strecke benötigt. Dazu fährt während jedes Tests ein Fahrzeug auf der Strecke mit denselben Startbedingungen und der identischen Parametrisierung wie denen des Ego-Fahrzeugs mit dem Unterschied, dass es keinerlei Beeinflussung erfährt.

6.2.2. Aussagekraft einzelner Streckenlayout-Typen

Das Ergebnis für eine Parametervariation für den Referenzalgorithmus ist in Abbildung 6-3a dargestellt. Die Grafik zeigt, welcher Ausgang des Szenarios mit welchem Zeitverlust τ_{loss} für diese Parametrisierung des Gegner-Fahrzeugs einhergeht. Der Ausgang eines Szenarios wird dabei unterteilt in (1) Erfolg, wenn das Gegner-Fahrzeug erfolgreich überholt wurde und damit sich im Frenet-Koordinatensystem hinter dem Ego-Fahrzeug befindet, (2) Kollision, wenn beide Fahrzeuge miteinander kollidiert

[83] Festgelegt anhand der Streckenbreiten realer Rennstrecken mit Zertifizierung des internationalen Dachverbands von Automobilclubs und Motorsportvereinen FIA.
Vgl. FIA: List of FIA licensed circuits (2020).

[84] Ermittelt im Rennspiel Projekt Cars 2 und verifiziert anhand von realem Kartenmaterial.

sind und (3) Abbruch, wenn das Überholmanöver nicht erfolgreich war. Der Parameter auf der Ordinate ist die Länge der ersten Geraden S1. Während der Versuche zeigte sich, dass insbesondere diese Länge einen großen Einfluss auf die Entwicklung der Szenarien besitzt, da durch eine lange Gerade bis zur Kurve auch die Wahrscheinlichkeit steigt, dass der Algorithmus bereits vor der Kurve in der Lage ist, zu überholen. Sowohl die Länge der Kurven als auch deren Krümmungen zeigen dagegen in den Daten nur einen geringen Einfluss sowohl auf die ausgewerteten Metriken als auch auf das Endereignis.

Die Verteilung des Zeitverlustes für den Fall des Abbruchs über der Länge S1 zeigt deutlich, dass die Länge der ersten Geraden einen eindeutigen Einfluss auf den Zeitverlust besitzt. Das ist vor allem damit zu erklären, dass mit einer längeren Startgeraden auch ein unterschiedlich langes Szenario einhergeht. Je nachdem, welche Parameter die Strecke besitzt, ist der mögliche Zeitverlust durch ein langsames vorausfahrendes Fahrzeug unterschiedlich. Die Verteilung der Punkte für Abbrüche ❶ zeigt aber auch, dass auch für eine einheitliche Länge der ersten Geraden die verlorene Zeit bei einem erfolgreichen Überholmanöver um $\pm\ 0{,}5\,\mathrm{s}$ variiert.

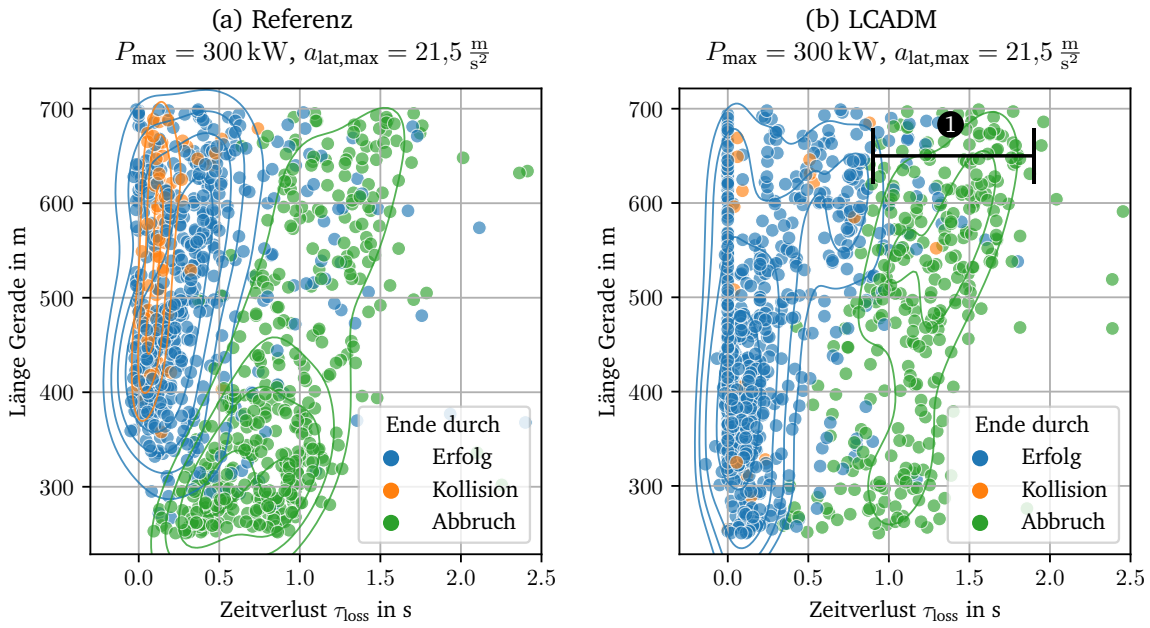


Abbildung 6-3.: Versuchsergebnisse Länge S1 über Zeitverlust für Referenzalgorithmus (links) und LCADM-Algorithmus (rechts) in zufällig generiertem Szenario unterschieden nach Ausgang des Szenarios. Die Linien geben die 2d-Kerndichteschätzung für jeden Szenarioausgang an.

In der Grafik zeigen sich auch bereits erste Abhängigkeiten der Ereignis-Häufigkeit von der Szenario-Parametrisierung. Abbildung 6-3b zeigt das Versuchsergebnis für den gleichen Versuch mit dem LCADM-Algorithmus. Der Vergleich mit Abbildung 6-3a zeigt sich, dass die Länge der ersten Geraden S1 insbesondere auch die Häufigkeit

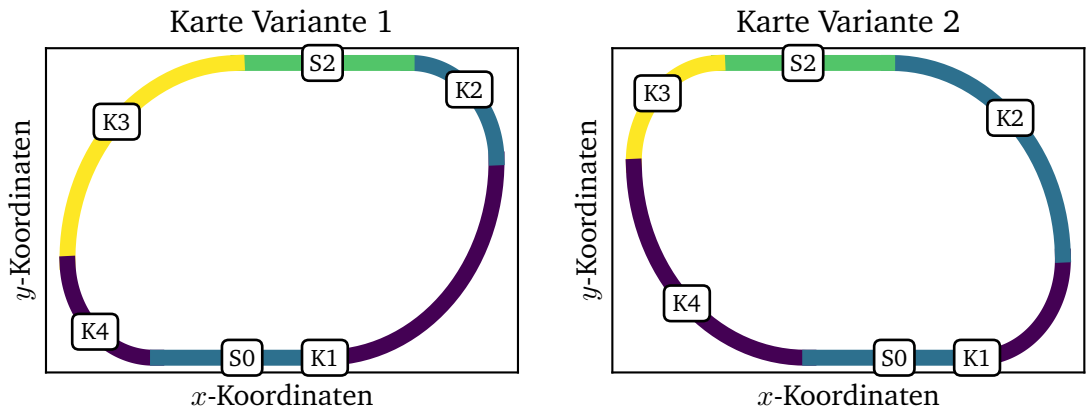


Abbildung 6-4.: Versuchslayout für Test der Relevanz der Reihenfolge von Kurven. Segmente sind eine Gerade mit $100\,\mathrm{m}$ Länge und zwei $90\,^\circ$ Kurven mit Radien von $50\,\mathrm{m}$ bzw. $100\,\mathrm{m}$. Dabei wird folgende Parametrisierung für das Gegner-Fahrzeug verwendet. Algorithmus: Referenz, Parameter: $P_{\mathrm{max}} = 335\,\mathrm{kW}$, $a_{\mathrm{lat,max}} = 17\,\frac{\mathrm{m}}{\mathrm{s}^2}$, Ergebnis: Überholrate Variante 1: $95\,\%$, Überholrate Variante 2: $0\,\%$

der erfolgreichen Überholmanöver beeinflusst und dass der Einfluss nicht identisch ist zu dem Einfluss auf den Referenzalgorithmus. Für den LCADM-Algorithmus sind die Überholmanöver für eine Länge von $400\,\mathrm{m}$ der Geraden S1 besonders häufig erfolgreich. Ein zufällig generiertes Streckenlayout nimmt damit Einfluss auf quantitative Auswertemetriken. Zeitabhängige Metriken besitzen zum einen eine Abhängigkeit von der Parametrisierung des Szenarios, zum anderen beeinflusst die Länge des Layouts das Ausgangsereignis für beide getesteten Agenten unterschiedlich. Eine quantitative Aussage bei Testdurchführung auf zufälligen Streckenlayouts besitzt damit keine direkte Aussagekraft für den Erfolg auf einer bestimmten Strecke.

Um den Einfluss der Streckenabschnitte mit repräsentativem Charakter auf einen quantitativen Vergleich zu untersuchen, wurde ein weiterer Versuch durchgeführt. Dazu wurde das in Abbildung 6-4 dargestellte, beispielhafte Streckenlayout erstellt. Die Strecke wurde jeweils 100-mal vorwärts (Variante 1) und rückwärts (Variante 2) vom Referenzalgorithmus durchfahren. Die Startbedingungen sind dabei identisch zu denen im Versuch zuvor beschriebenen Bedingungen. Durch das Versuchsdesign ergibt sich, dass obwohl dieselben Streckenelemente nur in einer anderen Reihenfolge verwendet wurden, sich die Rundenzeit sich um ca. $10\,\%$ unterscheidet. Dazu kommt, dass bei der Ausführung bei der Durchfahrt von Variante 1 der Algorithmus zu $95\,\%$ erfolgreich überholt. Bei der Durchfahrt von Variante 2 dagegen ist der Algorithmus nie erfolgreich mit der gegebenen identischen Parametrisierung. Die Reihenfolge von bestimmten Kurven ist daher ein wichtiger Einflussfaktor auf die Möglichkeit, ein Überholmanöver erfolgreich durchzuführen. Der Einfluss des Streckenlayouts auf eine quantitative Auswertung ist damit auch für die Verwendung eines repräsentativen Streckenlayouts gezeigt.

Die Auswahl der Strecke, auf der das Benchmarking durchgeführt wird und auch die Startbedingungen besitzen demnach einen großen Einfluss auf das Ergebnis. Aus diesem Grund besitzt eine quantitative Aussage nur Aussagekraft für den Vergleich von zwei Algorithmen in Bezug auf die Strecke, auf der der Test durchgeführt wurde. Für eine qualitative Evaluation wird dagegen das in diesem Abschnitt beschriebene, zufällig generierte Kurvenszenario verwendet, da dieses im Vergleich zu repräsentativ ausgewählten Kurven eine höhere Vielfalt besitzt und dadurch die Untersuchung des Einflusses von mehr Parametern möglich ist.

6.3. Parametervariationen

Die Parametrierung bzw. Fahrzeugauslegung von Ego- und Gegner-Fahrzeug komplettieren die Beschreibung der Szene des Tests. Sie bestimmen maßgeblich nicht nur die Rundenzeit, die ein Algorithmus in der Lage ist, mit einem bestimmten Fahrzeugmodell oder Fahrzeug zu fahren, sondern ebenfalls, an welchen Stellen ein Fahrzeug relativ zu einem anderen Fahrzeug in Lage ist, schneller zu fahren. Dadurch ergeben sich die Möglichkeiten für ein Überholmanöver, wenn eine fehlerfreie Fahrt der Algorithmen angenommen wird. Das gegnerische Fahrzeug erfüllt dabei notwendigerweise die Bedingung, dass die erreichte Rundenzeit höher ist als die des Ego-Fahrzeugs, da andernfalls ein Überholmanöver nicht möglich ist, da sich in diesem Fall die Entfernung zwischen den Fahrzeugen vergrößert.

In dieser Arbeit werden für die Trajektorienplanung Modellprädiktive Regler (MPC) in einer “perfekten“ Simulation genutzt (siehe Beschreibung in Abschnitt 7.4). Das Fahrzeug wird in der “perfekten“ Simulation bewegt, indem es von der momentanen Position auf den nächsten berechneten Punkt der Trajektorie gesetzt wird. Das Fahrzeug bewegt sich daher insofern “perfekt“, dass kein Regelalgorithmus die Position des Fahrzeugs bestimmt, sondern sich die Position allein aus den Berechnungen der Trajektorienplanung ergibt. Zwei Fahrzeuge mit identischer Parametrisierung fahren daher immer mit der identischen Geschwindigkeit und ohne Fahrfehler. Aus diesem Grund wird über die Parametrierung ebenfalls modelliert, dass ein Regelalgorithmus eines Fahrzeugs nicht immer den vollen Umfang der möglichen Quer- oder Längsbeschleunigung ausnutzt und das Fahrzeug nicht perfekt geregelt wird. Die Injektion von Fahrfehlern wird an dieser Stelle nicht betrachtet und ist eine mögliche Richtung für zukünftige Forschungen.

Grundsätzlich beeinflusst nahezu jeder Parameter eines Fahrzeugs auf die eine oder andere Weise sowohl die Rundenzeit und damit auch Stellen von relativem Geschwindigkeitspotenzial in Bezug auf ein anderes Fahrzeug. Eine Abstimmung eines Rennfahrzeugs ist dabei wesentlich vielschichtiger als die alleinige Suche nach der schnellsten Rundenzeit, da auch Gesichtspunkte wie der Reifenverschleiß oder die Fahrbarkeit

des Fahrzeugs eine Rolle spielen.[85a] Generell lässt sich aber sagen, dass sowohl lange Geraden als auch schnelle Kurven den größten Anteil für eine schnelle Rundenzeit liefern und daher als Grundlage für die Auslegung eines Rennfahrzeugs für eine Strecke genutzt werden.[85b] Die Auslegung ist dabei von der Aerodynamik des Fahrzeugs geprägt und in sich gegenläufig. Ein niedriger Luftwiderstand und damit verbunden ebenfalls niedriger Abtrieb, der maßgeblich für eine hohe Geschwindigkeit auf der Geraden verantwortlich ist, sorgt gleichzeitig dafür, dass die Durchfahrt einer Kurve nur mit geringer Querbeschleunigung möglich ist.[85c] Ein weiterer wichtiger Punkt, der an dieser Stelle anzumerken ist, ist, dass in einem Rennen Überholmanöver nur entweder auf einer Geraden oder in der Nähe oder direkt in einer Kurve möglich sind. Auf der Geraden ist dabei ein Vortrieb, der höher ist als der eines vorausfahrenden Fahrzeugs oder ein niedrigerer Luftwiderstand entscheidend. Das Überholen vor/in/hinter einer Kurve dagegen wird maßgeblich durch eine höhere mögliche Kurvengeschwindigkeit begünstigt, da die maximale Kurvengeschwindigkeit an dieser Stelle durch die maximale laterale Beschleunigung begrenzt ist. Die beiden Auslegungsparameter begünstigen also jeweils gegenläufig unterschiedliche Arten, ein anderes Fahrzeug zu überholen. Aus diesen Überlegungen heraus werden im Folgenden zwei Varianten für eine Parametervariation diskutiert.

- Zufällige Variation aller Parameter ohne zusätzliche Randbedingung
- Separate Variation der maximal übertragbaren Längs-/Querbeschleunigung bzw. der verfügbaren Vortriebsleistung

Werden alle Parameter zufällig variiert, hat das den Vorteil, dass sich die Anzahl an Gegnervarianten erhöht. Dagegen steht das Problem der Auswertbarkeit der Daten, die mit steigender Anzahl zufälliger Variablen komplexer wird. Dazu kommt, dass, wie von Trzesniowski beschrieben, die Parameter tatsächlich keine unabhängigen Variablen darstellen. Eine komplett zufällige Variation würde daher zwar sehr viele Varianten generieren, aber auch einen enormen Mehraufwand für Auswertung und Testdurchführung bedeuten für Varianten, die in dieser Form nicht realisierbar sind.[85d] Diese Form der Parametervariation wird aus den genannten Gründen nicht weiter verfolgt.

Die Unterscheidung nach zum einen der möglichen Querbeschleunigung bzw. Verzögerung und der Vortriebsleistung orientiert sich an dem Auslegungsproblem von Kurvengeschwindigkeit und Maximalgeschwindigkeit auf der Geraden. An dieser Stelle wird vereinfachend angenommen, dass die maximal mögliche Bremsbeschleunigung des Fahrzeugs und die maximal mögliche laterale Beschleunigung identisch sind, da

[85] Trzesniowski, M. et al.: Rennwagentechnik - Datenanalyse, Abstimmung (2017), a: S. 208ff, b: S.241f, c: S. 248, d: S. 208f.

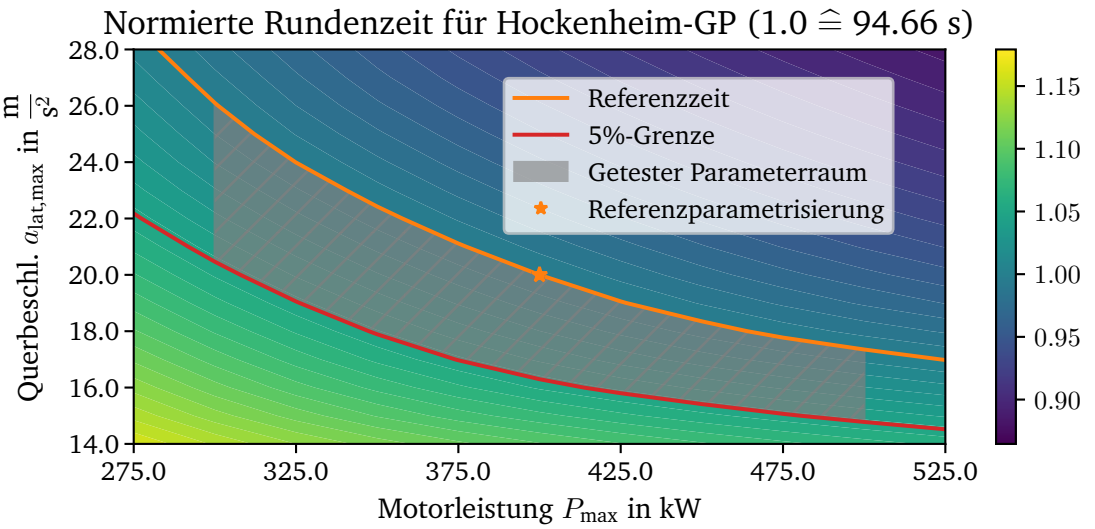


Abbildung 6-5.: Normierte Rundenzeit für die Hockenheim Grandprix Strecke bei Variation der Vortriebsleistung und der maximalen lateralen Beschleunigung. Die Rundenzeit ist auf die Rundenzeit des Fahrzeugs mit folgender Referenzparametrisierung normiert: $P_{\text{max}} = 400\,\text{kW}$, $a_{\text{lat,max}} = 20\,\frac{\text{m}}{\text{s}^2}$

sie beide von der Radaufstandskraft und dem Reifen beeinflusst werden.[86] Diese Unterscheidung bietet den Vorteil, dass die damit abgebildete relative Performance des gegnerischen Fahrzeugs näher an den Eigenschaften realer Fahrzeuge liegt als bei einer rein zufälligen Parametrierung des Fahrzeugs. Abbildung 6-5 zeigt beispielhaft die Rundenzeit für die Variation der zwei Parameter für die Hockenheim-Grand-Prix-Strecke. Die Rundenzeit des Fahrzeugs $\tau_{\text{lap,opp}}$ wird auf die Rundenzeit eines Ego-Fahrzeugs $\tau_{\text{lap,ego}}$ mit Referenzparametern normiert. Der erste variierte Parameter ist die vorhandene Vortriebsleistung und der zweite die kombinierte maximale laterale Beschleunigung mit der Bedingung $a_{\text{lat,max}} = D_{\text{max}}$. Die Rundenzeit wird ermittelt, indem der in Abschnitt 7.4 vorgestellte Trajektorienplaner eine fliegende Runde der Strecke fährt. Die benötigte Zeit wird anhand der ermittelten Positionen zu diskreten Zeitschritten interpoliert.

Die Abbildung zeigt, dass eine Variation der beiden Parameter sich nicht gleichmäßig auf die Rundenzeit auswirkt. Da die gefahrene Rundenzeit als einfachster Indikator für die Güte eines “allein fahrenden“ Algorithmus für eine Klassifikation nutzbar ist, wird daher im Folgenden eine Parametervariation für einzelne Rennstrecken mit Blick auf die entsprechende Rundenzeit durchgeführt.

Die minimale normierte Rundenzeit eines Gegner-Fahrzeugs, bei der es noch möglich ist, zu überholen, ist über den Quotienten aus der Rundenzeit des Gegner-Fahrzeugs $\tau_{\text{lap,opp}}$ und der Rundenzeit des Ego-Fahrzeugs $\tau_{\text{lap,ego}}$ zu $\tau_{\text{lap,opp}}/\tau_{\text{lap,ego}} > 1$ gegeben. Anhaltspunkte, wie eine entsprechende Untergrenze zu wählen ist, gibt es in der

[86] Vgl. Frömmig, L.: Grundkurs Rennwagentechnik (2019), S. 157f.

Formel 1. Die dort eingeführte $107\,\%$-Regel ist eine Hürde, dass die im Qualifying erreichte Rundenzeit eines Fahrzeugs nicht um mehr als $7\,\%$ höher sein darf als die des schnellsten Fahrzeugs, um am Rennen teilnehmen zu dürfen.[87] Bei einem so großen Unterschied ist aber schon nicht mehr an ein kompetitives Überholmanöver zu denken. Im Vergleich dazu in der Langstreckenserie WEC (World Endurance Championship), bei der verschiedene Klassen auf der Rennstrecke gleichzeitig fahren, ergaben sich 2020 prozentuale Rundenzeit-Unterschiede zwischen den verschiedenen Klassen wie in Tabelle 6-2 gezeigt. In einem WEC Rennen bewegen sich Fahrzeuge aus einzelnen Fahrzeug-Klassen gleichzeitig miteinander auf der Strecke, fahren aber nicht denselben Wettbewerb. Die Tabelle beschreibt, wie unterschiedlich die Rundenzeiten vom schnellsten Fahrzeug einer Klasse ↑ zum langsamsten Fahrzeug einer Klasse ↓ und zu den anderen Klassen im Jahr 2020 in Le Mans verhalten haben. Innerhalb einer Rennklasse zeigt sich ein maximaler Rundenzeit-Unterschied von $4{,}5\,\%$. Der Unterschied zwischen dem schnellsten Fahrzeug einer Klasse zum langsamsten Fahrzeug der nächst langsameren Klasse liegt bei etwa $10\,\%$. Der Unterschied innerhalb einer Klasse ist für die GTEPro Klasse am geringsten, da dort durch die Balance-of-Performance-Regeln dafür gesorgt wird, dass einzelne Fahrzeuge keinen Rundenzeit-Vorteil besitzen bzw. ein etwaiger Vorteil durch Maßnahmen wie Zusatzmassen relativiert wird.[88] Da es in dieser Arbeit zunächst um den Vergleich von gleichwertigen Fahrzeugen geht, wird ein maximaler Rundenzeit-Unterschied von $5\,\%$ angesetzt.

Tabelle 6-2.: Relative Rundenzeitunterschiede der Rennklassen in der WEC auf dem Circuit de la Sarthe in Les Mans 2020[89]. Die Zeiten basieren auf den Qualifyingzeiten der Fahrzeuge.

	LMP1 ↑	LMP1 ↓	LMP2 ↑	LMP2 ↓	GTEPro ↑	GTEPro ↓
LMP1 ↑	0.0%	4.0%	4.7%	9.5%	18.3%	19.1%
LMP1 ↓	-	0.0%	0.7%	5.3%	13.7%	14.5%
LMP2 ↑	-	-	0.0%	4.5%	12.9%	13.7%
LMP2 ↓	-	-	-	0.0%	8.0%	8.8%
GTEPro ↑	-	-	-	-	0.0%	0.7%
GTEPro ↓	-	-	-	-	-	0.0%

Die zweite Dimension, die für die Parametervariation genutzt wird, ist die Motorleistung. Grund dafür ist, dass während Änderungen, die die möglichen Quer- und Bremsbeschleunigungen des Fahrzeugs adaptieren, mit großen Änderungen am Fahrzeug im Bereich der Aerodynamik, des Fahrwerks oder der Reifen verbunden sind, sind Änderungen der verfügbaren Vortriebsleistung mit geringem Aufwand auch an einem realen Fahrzeug umsetzbar oder bereits durch technische Maßnahmen vorhanden. Ähnliche Vorrichtungen, die ein Überholen einzelner Teilnehmer erleichtern, werden seit Jahren

[87] Vgl. FIA: Formula 1 Sporting Regulations Issue 9 (2021), S. 33.

[88] Vgl. FIA: Technical Regulations for Grand Touring Cars (2020), S. 4.

[89] FIA: FIA WEC - Les Mans 2020 News (2021).

in der Formel 1 in Form des Drag Reduction Systems (DRS) eingesetzt, das den Luftwiderstand auf durch das Reglement festgelegten Geraden durch Öffnen einer Klappe am Heckflügel senkt[90a] und das Energy Recovery System (ERS), dass auf Kopfdruck einen elektrischen Motor als Überholhilfe freischaltet.[90b] Auch in der Formel e sind derartige Hilfen im Einsatz, die einem Fahrer ermöglichen, pro Rennen für einen begrenzten Zeitraum eine höhere elektrische Leistung aus der Batterie abzurufen.[91] Die maximale Querbeschleunigung $a_{\text{lat,max}}$ lässt sich für diesen Fall als Funktion der Vortriebsleistung und des Rundenzeit-Koeffizienten zu $a_{\text{lat,max}}(P_{\text{max}}, \tau_{\text{lap,opp}}/\tau_{\text{lap,ego}})$ beschreiben. Eine Parametrisierung wird demnach beschrieben durch die Vortriebsleistung P_{max} für einen Rundenzeit-Koeffizienten $\tau_{\text{lap,opp}}/\tau_{\text{lap,ego}} = 1$ und den Rundenzeit-Koeffizienten $\tau_{\text{lap,opp}}/\tau_{\text{lap,ego}}$ der Parametrisierung.

Um einen möglichst großen Unterschied zwischen einem schnellen Fahrzeug in der Kurve bzw. auf der Geraden sichtbar zu machen, wird ein Parameterbereich für den Vortrieb von $400\,\text{kW} \pm 100\,\text{kW}$ definiert. Zusammen mit der Begrenzung der Rundenzeiten ergibt sich der zu untersuchende Parameterraum in Abbildung 6-5 als grau schraffierter Bereich.

Für ein zufälliges Szenario nach Abschnitt 6.2.1 ist dagegen eine solche Berechnung der Rundenzeit nur unter hohem Simulationsaufwand durchführbar. Die offene Form der Strecke erschwert zusätzlich die Definition der Rundenzeit selbst. Da die zufälligen Szenarien nicht für eine quantitative Auswertung genutzt werden, wird stattdessen eine äquidistant verteilte Variation der Parameter Vortriebsleistung und laterale Beschleunigung durchgeführt.

6.4. Auswertbare Metriken

Die Metriken, die nach bzw. während der Testdurchführung untersucht werden, bestimmen, ob ein Test als erfolgreich klassifizierbar ist und dienen im Falle einer Validierung der Überprüfung definierter Anforderungen. Für das in dieser Arbeit vorgestellte Benchmarking, das sich in die zwei wesentlichen Bereiche der qualitativen und quantitativen Bewertung aufteilt, bilden die Metriken ebenfalls die Evaluationsgrundlage.

Die Generierung sowohl von qualitativen als auch von quantitativen Aussagen benötigt Metriken, die für den jeweiligen Untersuchungszweck eine möglichst hohe Aussagekraft besitzen. Hier bietet der Motorsport im Vergleich mit dem normalen Straßenverkehr andere Grundvoraussetzungen. Die Regeln, die im Motorsport gelten, sind einfach und besagen, dass ein Fahrzeug grundsätzlich Vorrang besitzt, wenn es sich vor einem anderen Fahrzeug befindet. Allerdings bedeutet diese Regel nicht, dass das Fahrzeug, das sich physisch weiter hinten auf der Strecke befindet, immer automatisch die

[90] Vgl. FIA: Formula 1 Technical Regulations Issue 9 (2021), a: S. 31ff, b: S. 10.

[91] Vgl. FIA: Formula e Sporting Regulations (2021), S. 64.

Schuld trägt, wenn eine Kollision auftritt. Bei Rennen im Motorsport werden dafür zusätzlich immer auch sogenannte Stewards eingesetzt, die über die Schuldfrage bei einer Kollision entscheiden.[92] Eine automatische Schuldzuweisung für Kollisionen wird im Rahmen dieser Arbeit daher nicht realisiert. Die Metrik, dass sich eine Kollision ereignet hat, ist an dieser Stelle die Einzige, die sich zuverlässig und definitiv auswerten lässt.

Andere häufig verwendete Maßstäbe, die dazu bestimmt sind, die Kritikalität von Szenarien zu beschreiben, wie bspw. die TTC (engl. Time-to-Collision)[93] oder die Kritikalitätsmetrik nach Junietz[94] basieren auf den Differenzgeschwindigkeiten der beiden untersuchten Fahrzeuge. In einem Motorsportkontext ist diese Herangehensweise der Evaluation nicht möglich, da durch die kompetitive Natur des Motorsports das Fahren mit sehr geringen Abständen üblich ist. Dazu kommt, dass bedingt durch Bremsphasen und hohe Kurvengeschwindigkeiten sich nahezu zu jedem Zeitpunkt das Fahrzeug unter hoher Längs- und/oder Querbeschleunigung befindet. Damit sind die getroffenen Grundannahmen solcher Metriken nicht erfüllt.

Die qualitative Bewertung zielt darauf ab, die Stärken und insbesondere Schwächen eines Algorithmus herauszufinden. Es geht also nicht um absolute Aussagen zu einem definierten Maßstab, sondern darum, einen relativen Vergleich mit anderen Algorithmen zu ermöglichen und durch die untersuchten Metriken gezielt Schwachstellen zu identifizieren.

Bislang wurde in dieser Arbeit davon ausgegangen, dass ein Algorithmus gegen einen anderen einzelnen Algorithmus oder ein dynamisches Objekt getestet wird. Damit ist möglich, im Falle einer qualitativen Systembewertung danach eine Aussage zu treffen, ob eine Kollision stattgefunden hat oder das Überholmanöver erfolgreich war. Wie der Test in Abschnitt 6.2 zeigt, ist eine Aussage über die Zeit, die ein Fahrzeug während eines Überholmanövers verliert, in diesem Test ebenfalls möglich. Der bereits vorgestellte Test zeigt auch, dass eine zeitliche Metrik in einem zufälligen Szenario auch für die qualitative Bewertung nur bedingt aussagekräftig und immer in Relation zu anderen Algorithmen für identische Testbedingungen zu sehen ist. Die verlorene Zeit sagt aber aus, an welchen Stellen in einem Szenario Zeit verloren geht und lässt damit im Vergleich mit anderen Algorithmen auch Rückschlüsse auf individuelle Stärken zu.

Eine Aussage, aus welchem Grund eine gezeigte Überlegenheit oder Schwäche auftritt, lässt sich daraus aber nicht ableiten. Die Diversität von Rennfahrzeugen, dass Fahrzeuge an unterschiedlichen Stellen auf Strecken überholen, da sie eine unterschiedliche

[92] FIA: Formula 1 Sporting Regulations Issue 9 (2021), S. 39.

[93] Winner, H. et al.: Handbuch Fahrerassistenzsysteme (2015), S. 187.

[94] Junietz, P. M.: Diss., Microscopic and Macroscopic Risk Metrics for Safety Validation (2019).

Abstimmung [95] besitzen, wird bei der verlorenen Zeit ebenfalls außer Acht gelassen. Es zeigt sich in den Daten und der Auswertung, dass die Variation der Parametrisierung und des Streckenlayouts oft für bestimmte Szenen dazu führt, dass Unfälle, die durch Fehler in der Programmierung der Algorithmen verursacht werden, gehäuft an ähnlichen Streckenabschnitten auftreten. Da sowohl die Programmierung eines Agenten als auch die Parametrisierung des Fahrzeugs wesentlichen Einfluss darauf nehmen, an welchen Stellen ein Überholmanöver möglich ist, wird die Position der verschiedenen Ereignisse als aussagekräftige Metrik identifiziert, um Schwachstellen eines Algorithmus zu finden. Diese Information ist zusätzlich nutzbar, um zu ermitteln, welche Fähigkeiten ein bestimmter Algorithmus besitzt, da ein Überholmanöver in einer Kurve andere planerische Fähigkeiten verlangt als ein Überholmanöver auf einer geraden Strecke, bedingt durch die unterschiedlichen Ideallinien.

Die Erweiterung der Parametervariation aus Abschnitt 6.3 ermöglicht, nicht nur eine binäre Aussage zu erhalten, ob und wie häufig ein Fahrzeug überholt oder am Überholen gehindert wird. Die Erweiterung der Parameter ergänzt zusätzlich die Möglichkeit für eine Aussage, wie schnell ein anderes Fahrzeug sein darf, sodass ein Überholmanöver bzw. das Verhindern eines Überholmanövers weiterhin möglich ist.

Die quantitative Evaluation verfolgt das Ziel, eine bestimmte Menge an Algorithmen in eine Rangfolge einzuordnen. In dieser Arbeit bereits vorgestellte Methoden der Spieltheorie basieren dabei in den meisten Fällen auf einem einfachen Gewinnschema. Gewinnt der Spieler, ist der Ausgang des Spiels zu 1 definiert, verliert er zu 0. In einigen Fällen wie dem Elo-System im Schach, wird ein unentschiedenes Spiel mit 0,5 gewertet. Dieses einfache Schema ist eine notwendige Grundannahme, auf der alle vorgestellten Ansätze basieren und ist für Nullsummenspiele gültig. Eine Erfolgsmetrik ist daher nur dann für ein solches Schema geeignet, wenn eine Begegnung zwischen zwei Spielern mit einem absolut skalierten Wert zwischen 0 und 1 bewertet wird. Über eine Vielzahl von Begegnungen errechnet sich dann der Erwartungswert einer bestimmten Begegnung p_{nm} in der Payoff-Matrix $\boldsymbol{p}$. Gleichzeitig besteht die Forderung, dass die Summe der Ergebnisse für alle Spieler innerhalb eines Szenarios 1 ergibt.

Ein einzelnes Szenario ist immer als Teil eines kompletten Rennens zu verstehen. Daher wird anders als für den Gewinn im Schach ein vereinfachtes Bewertungsschema für den Ausgang eines Szenarios verwendet. Der Ausgang eines Szenarios wird nur mit 1 bewertet, wenn es zu keiner Kollision gekommen ist, da ein Unfall je nach Rennserie, in der das Fahrzeug fährt, bereits dazu führt, dass eine kompetitive Weiterfahrt nicht mehr möglich ist. Da für diese Modellierung die Wahrscheinlichkeit für einen Unfall

[95] Ist ein Fahrzeug auf hohe Geschwindigkeit auf der Geraden ausgelegt, überholt es öfter auf der Geraden, während Fahrzeuge, die auf hohe Geschwindigkeiten in der Kurve ausgelegt sind, öfter in Kurven überholen. Die Gesamtrundenzeit ist bei beiden Varianten allerdings möglicherweise identisch.

p_{crash} nur implizit im Bewertungsschema enthalten ist, wird sie explizit als weiteres quantitativ auswertbares Ergebnis mit aufgeführt. Das Ergebnis eines Tests bezüglich der Unfälle ist damit definiert zu $1 - p_{\text{crash}}$ und eine Wahrscheinlichkeit von 0 für einen Unfall führt zum maximal erreichbaren Testergebnis von 1.

Die Forderung nach der Summe 1 für beide Spieler und die Auswertung eines Unfalls mit dem Ausgang 0 für beide Spieler machen an dieser Stelle notwendig, das Szenario aufzuteilen. Der Test gegen einen anderen Spieler wird als Aufgabe betrachtet. Wird erfolgreich überholt, ist das Ergebnis der Erfolg des Spielers. Im Falle einer Kollision oder des nicht erfolgreichen Überholens gewinnt die Aufgabe. Auf die gleiche Weise wird das Verteidigen gegen einen Gegner für den zu testenden Algorithmus als Aufgabe modelliert, die nur unterscheidet, nach Erfolg oder nicht Erfolg. Nur durch diese getrennte Betrachtungsweise in Überholen, Verteidigen und die jeweiligen Kollisionen ist sichergestellt, dass es möglich ist, ein Ranking-Verfahren anzuwenden.

Als Alternative zur binären Metrik, ob ein Szenario erfolgreich oder nicht erfolgreich war, bieten die bisher diskutierten Metriken das Potenzial, sowohl für das Überholen als auch für das Verteidigen absolut definierbare kontinuierliche Testergebnisse zu definieren, die die Forderung nach der Nullsumme erfüllen. Für das Verteidigen ist hier die Entfernung bzw. Zeit anzuführen, in der es einem Algorithmus gelingt, einen anderen Algorithmus am Überholen zu hindern. Da eine Zeit- bzw. Wegbeschränkung als Abbruchbedingung für den Test notwendig ist, ist ein Testergebnis als Erfüllungsgrad der Aufgabe in Prozent definierbar. Auf diese Weise ist möglich, zwischen einem Manöver zu unterscheiden, das bei $99\,\%$ der Zeit abgebrochen wurde und einem, bei dem das Ego-Fahrzeug bereits nach $5\,\%$ des Weges überholt wurde. Für eine binäre Auswertung sind diese beiden Fälle identisch mit 0 bewertet.

Ebenso gilt es für den Fall des Überholens, dass auch hier für jedes Manöver ein spezifizierter Zeit- bzw. Streckenrahmen notwendig ist. Das Testresultat lässt sich daher ebenso wie für den Fall des Verteidigens definieren als benötigten Anteil der Strecke für ein Überholmanöver in Relation mit dem verfügbaren Rahmen. Die Aussagekraft und deren möglicher Einfluss auf ein Ranking werden in der Evaluation in Abschnitt 8.2 genauer untersucht. Grundsätzlich gilt aber, dass der Streckenverlauf beeinflusst, wie lange es dauert, bis ein Überholmanöver stattfindet. Aus diesem Grund wird nicht nur ein einzelner Test auf einer Strecke durchgeführt, sondern für einen Test eines Algorithmus werden auf einer Strecke immer zufällige Startpunkte gewählt. Je höher die Anzahl der Testdurchführungen wird, desto näher kommt der aus den Testergebnissen berechnete Erwartungswert dem wahren Wert. Ebenso sinkt durch die zufällige Verteilung der Startpositionen der Einfluss einzelner Elemente im Streckenlayout, da die Ergebnisse aller Agenten durch die Streckenelemente gleichermaßen beeinflusst werden.

Die Präferenz der kontinuierlichen über der diskreten Metrik wird für das Überholen und für das Verteidigen zusammen mit den Ergebnissen der Tests in Abschnitt 8.2 diskutiert.

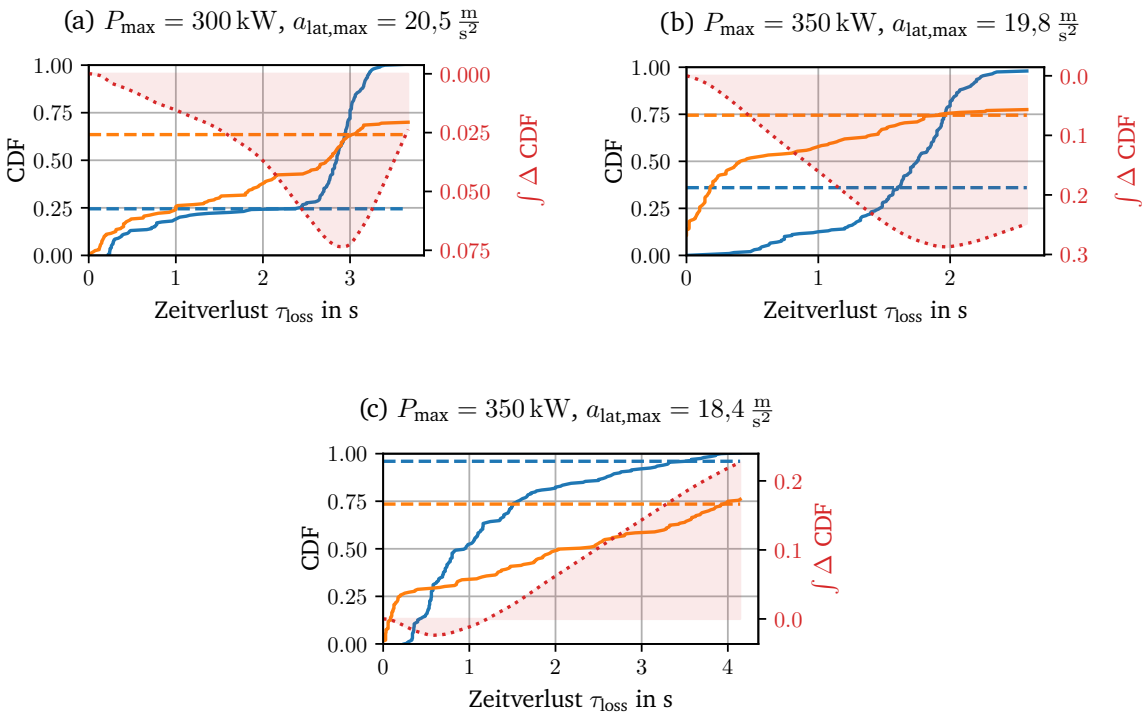


Abbildung 6-6.: Vergleich CDF des Zeitverlusts τ_{loss} an verschiedenen Arbeitspunkten auf der Hockenheim Grandprix Strecke. Blau Referenzalgorithmus, orange LCADM, horizontale gestrichelte Linien zeigen die Überholrate des jeweiligen Algorithmus. Gezählt werden nur Manöver ohne Kollision. Offset zum Maximalwert 1 zeigt die Unfallrate. Die rote gepunktete Linie zeigt, die Differenz des Integrals der beiden Kurven.

Die Verwendung der verlorenen Zeit τ_{loss} als Erfolgskriterium für ein Szenario und dem definierten Layout einer einzelnen Strecke wird an dieser Stelle ebenfalls untersucht. Um eine Eignung der Metrik festzustellen, wurde folgender Versuch durchgeführt. Verglichen wird die verlorene Zeit während eines Überholmanövers für den Referenzalgorithmus und den LCADM-Algorithmus. Dafür werden jeweils an 500 zufällig gewählten Startpunkten auf der Hockenheim-Grand-Prix-Strecke die beiden Algorithmen für verschiedene Arbeitspunkte des Gegner-Fahrzeugs getestet. Die Startbedingungen sind erneut identisch mit Abbildung 6-2. Ermittelt wurde dafür der Zeitverlust für beide Algorithmen in der Rolle des Ego-Fahrzeugs. Die CDF des Zeitverlusts ist in Abbildung 6-6 für drei ausgewählte Arbeitspunkte des Gegner-Fahrzeugs dargestellt. Die blauen Linien zeigen die CDF für den Referenzalgorithmus, die orangene, die für den LCADM-Algorithmus. In die Zählung des Diagramms gehen nur Szenarien ohne Kollision ein. Der Offset zu CDF $= 1$ zeigt damit die Anzahl der Unfälle. Die gestrichelten Linien zeigen die gesamte Anzahl an erfolgreichen Überholmanövern an. Um als Metrik für

den quantitativen Vergleich zwischen mehreren Algorithmen nutzbar zu sein, ist auch hier die Bedingung zu erfüllen, dass der Erfolg eindeutig zwischen 0 und 1 skalierbar ist. Für die Untersuchung und Auswahl einer Schwelle für die Skalierung der Metrik ist zusätzlich auf der rechten Ordinate als rote gepunktete Linie die Differenz der Integrale der beiden Kurven abgebildet. Dieses Integral beschreibt also die Anzahl der erfolgreichen Szenarien gewichtet mit der dabei verlorenen Zeit.

Der Verlauf der blauen Kurve in Abbildung 6-6a zeigt auf, dass die Metrik des Zeitverlustes nicht normalverteilt ist. Das liegt zum einen daran, dass sowohl die erfolgreichen Überholmanöver eingerechnet werden als auch die Manöver, in denen kein Überholen stattgefunden hat. Die zugrunde liegenden Verteilungen sind aber ebenfalls nicht normalverteilt. Die orangene Linie in Abbildung 6-6a zeigt auf, dass der Zeitverlust für den Arbeitspunkt für erfolgreiche Überholmanöver und nicht erfolgreiche Überholmanöver je einseitig verteilt ist. Eine Beschreibung der Parameter der resultierenden Verteilungen ist damit als einzelne Vergleichsmetrik nicht möglich.

Gleichzeitig wird beim Vergleich von Abbildung 6-6a und Abbildung 6-6b deutlich, dass die Diagramme eine wichtige Informationsquelle darstellen, da, obwohl die Verhältnisse an Überholereignissen zwischen den beiden Algorithmen sich ähneln, die Ausprägungen der Kurven sich voneinander deutlich unterscheiden. Eine Korrelation zwischen den Gesamtergebnissen des Tests und der verlorenen Zeit ist damit zwar gegeben, aber nicht durch einen einzelnen Zusammenhang quantifizierbar.

Um diese Schwäche zu kompensieren, wurde für die durchgeführten Tests ebenfalls das Integral der jeweiligen CDF berechnet, im Versuch, auf diese Weise eine absolut skalierte Metrik zu schaffen. Dieses Integral ist umso größer, je mehr Szenarien mit einem geringen Zeitverlust gefahren wurden. Es spiegelt damit eingeschränkt den Vergleich der beiden Algorithmen wider. Das Integral zeigt bei deutlichen Unterschieden zwischen den Algorithmen ebenfalls einen deutlichen Ausschlag und korreliert mit dem Unterschied der erfolgreich durchgeführten Manöver. Allerdings zeigt der Verlauf keine linearen Zusammenhänge. Die Schwelle, zu der das Integral damit ausgewertet wird, spielt damit eine entscheidende Rolle für die Auswertung. Wie in Abbildung 6-6c deutlich, weist das Integral in einigen Fällen (für die hier durchgeführten Tests in 21 von 50 getesteten Arbeitspunkten) sogar Nullstellen auf, sodass eine gesetzte Schwelle zur Normierung ebenfalls Einfluss auf einen Vergleich hätte. Gleichermaßen verhält es sich für mögliche Gewichtungen der Kurve, sodass ein eindeutiger Gewinner beim Vergleich von zwei Algorithmen in diesen Fällen nicht bestimmbar ist. Eine Betrachtung der absoluten Werte in Abbildung 6-6b und Abbildung 6-6c zeigt, dass auch durch das Integral die Unterschiede zwischen den beiden Algorithmen stark vereinfacht werden.

Im Folgenden wird daher lediglich die Verwendung der Häufigkeit für das Eintreten des binären Erfolgsereignisses und der gefahrene Weg normiert auf die maximale

Gesamtlänge des Szenarios als Metriken für den quantitativen Vergleich weiterverfolgt.

6.5. Ranking-Methode

Nachdem in den vorangegangenen Abschnitten die wichtigsten Einflussparameter auf eine qualitative und quantitative Evaluation hin untersucht wurden, definiert der folgende Abschnitt die Methode, die in dieser Arbeit genutzt wird, um eine Ranking verschiedener Algorithmen zu erstellen. Wie bereits in Abschnitt 5.2 beschrieben, setzt sich die Aufgabe eines Ranking-Verfahrens aus zwei wesentlichen Bestandteilen zusammen. Die Ermittlung der Payoff-Matrix $\boldsymbol{p}$, um für jede Spieler-Begegnung den Erwartungswert p_{nm} zu berechnen und die Ermittlung eines Rankings.

Unter der Annahme der Transitivität ist die Berechnung von $\boldsymbol{p}$ ohne einen Test jeder Spieler-Kombination mit den Methoden von Elo, Glicko oder TrueSkill möglich. Ist die Annahme der Transitivität nicht valide, ist die Berechnung nur mittels der Methoden mElo nach Balduzzi[96], dem maxent NG oder α-Rank möglich. Um Forschungsfrage 2 zu klären, ob das hier untersuchte Szenario tatsächlich intransitiv ist, werden die Ergebnisse aus Abbildung 6-1 erneut untersucht. Die Erfolgsraten sind zur einfacheren Betrachtung in Tabelle 6-3 gezeigt.

Tabelle 6-3.: Erfolgsrate für Ego- und Gegner-Algorithmus-Variation auf Streckenlayout Hockenheim. Das Testvorgehen ist in Abschnitt 6.2 beschrieben.

	LCADM	Referenz	Keine Interaktion
LCADM	0,44	0,95	0,40
Referenz	0,49	0,67	0,11
Keine Interaktion	0,03	0,84	0,00

Dazu wird mithilfe der Payoff-Matrix aus der Tabelle, die durch die Tests in Abschnitt 6.1 ermittelt wurde, jeweils ein Ranking über das Elo- bzw. das mElo Verfahren berechnet. Zur Evaluation, ob das Szenario transitiv ist, wird die errechnete Payoff-Matrix der beiden Verfahren mit der realen Payoff-Matrix aus Tabelle 6-3 verglichen. Der errechnete Unterschied zwischen der realen Payoff-Matrix $\boldsymbol{p}_{\text{real}}$ und der approximierten Payoff-Matrix $\boldsymbol{p}_{\text{Elo}\,|\,\text{mElo}}$ wird definiert zu

$$\frac{1}{N^2} \sum_{n=1}^{\mathcal{N}} \sum_{m=1}^{\mathcal{N}} \left| p_{nm,\text{real}} - p_{nm,\text{Elo}\,|\,\text{mElo}} \right| \tag{6-1}$$

[96] Balduzzi, D. et al.: Re-evaluating Evaluation (2018).

für die Gesamtanzahl Spieler $\mathcal{N}$. Der Vergleich zeigt deutlich, dass das mElo-Verfahren mit einer Gesamtabweichung von 0,001 die reale Payoff-Matrix mit nur geringer Genauigkeitseinbuße approximiert. Das einfache Elo-Verfahren dagegen erreicht mit einer Gesamtabweichung von 0,1 nur eine sehr grobe Abschätzung der Werte. Die Interpretation des Wertes des mElo Verfahrens ist so zu verstehen, dass die Summe der absoluten Abweichungen über alle Einträge der Payoff-Matrix etwa $1\,\%$ beträgt. Dementsprechend werden die einzelnen Einträge der Matrix im arithmetischen Mittel bis auf $0{,}1\,\%$ angenähert. Daher ist in diesem Fall von einer intransitiven Beziehung der Agenten untereinander auszugehen. Als Vergleichsmaßstab für die Werte wurde die Gesamtabweichung für eine komplett zufällig gewählte Payoff-Matrix zu einem Wert von 0,3 mit einer Standardabweichung von 0,08 für 1000 zufällig gewählte Matrizen ermittelt. Im Folgenden wird daher das mElo-Verfahren für die Bildung des Rankings verwendet. In der Evaluation in Abschnitt 8.2 wird als Vergleich ebenfalls das maxent NG herangezogen und mit mElo hinsichtlich Eignung für die Erstellung eines Rankings verglichen. Die Methode α Rank ist nicht nutzbar, da die Methode bislang keine Möglichkeit bietet, eine Rangfolge für eine Spieler-vs.-Aufgabe-Modellierung zu berechnen, sondern auf eine reine Spieler-vs.-Spieler-Payoff-Matrix ausgelegt ist.

Brettspiele wie Schach oder Go und komplexe Strategiespiele wie Star-Craft, die bislang in der Spieltheorie untersucht wurden, bieten nahezu gleiche Ausgangsbedingungen und auch das identische Ziel für alle Spieler. Die minimalen Abweichungen der Startbedingungen (beim Schach bspw. der weiße Spieler, der zuerst zieht) können entweder ignoriert oder durch einen kleine Bias im Erwartungswert berücksichtigt werden.[97] Die in dieser Arbeit untersuchte Aufgabe, Rennen zu fahren, teilt sich dagegen in zwei nahezu komplett unterschiedliche Bereiche, das Überholen und das Verteidigen gegen einen Überholer. Bisher umgesetzte Algorithmen, die im Details in Abschnitt 7.2 vorgestellt werden, beschäftigen sich hauptsächlich mit der Aufgabe zu überholen. Nur wenige Ansätze zeigen bislang Anstrengungen, eine Verteidigung gegen ein überholendes Fahrzeug umzusetzen. Auf Basis der Analyse der Untersuchungsmetriken in Abschnitt 6.4 wird zusätzlich zur Zweiteilung in Überholen und Verteidigen jeweils noch die Kategorie Kollisionen hinzugefügt. Die Bereiche werden jeweils einzeln ausgewertet, um sicherzustellen, dass die Grundannahme des Nullsummenspiels erfüllt ist.

Im Folgenden für die Evaluierung und ebenfalls für die Erstellung eines Rankings wird daher die Performance viergeteilt in die beiden Bereiche des Überholens und Verteidigens mit jeweils den Kollisionen für die beiden Bereiche. Die Unterteilung wird vorgenommen, da die beiden Bereiche (a) nach der Analyse der Literatur und eigenen Beobachtungen einen unterschiedlichen Fundus an Fähigkeiten erfordern und (b) für die jeweilige Aufgabe ermöglichen, separate Algorithmen zu implementieren, die sich als Aufgabe in das hier gezeigte Benchmark integrieren lassen. In beiden Fällen wird,

[97] Vgl. Lazaridis, D.: An R package for multidimensional Elo ratings (2021).

wie die Diskussion der Metriken in Abschnitt 6.4 ergeben hat, die Erfolgshäufigkeit als Auswertekriterium verwendet. Für ein überholendes Fahrzeug bedeutet das, dass ein Szenario als erfolgreich gilt, wenn das vordere Fahrzeug ohne Kollision erfolgreich überholt wird. Andersherum für einen Verteidiger ist ein Szenario erfolgreich, wenn weder eine Kollision noch ein Überholmanöver innerhalb eines gesetzten Zeitrahmens oder Streckenrahmens stattfinden. Diese binären Ergebnisse der Tests werden in Abschnitt 8.2 zusätzlich mit der Variante des kontinuierlichen Payoffs verglichen, der ein Überholen innerhalb eines möglichst kurzen Streckenabschnittes und ein Verteidigen über einen möglichst langen Streckenabschnitt widerspiegelt.

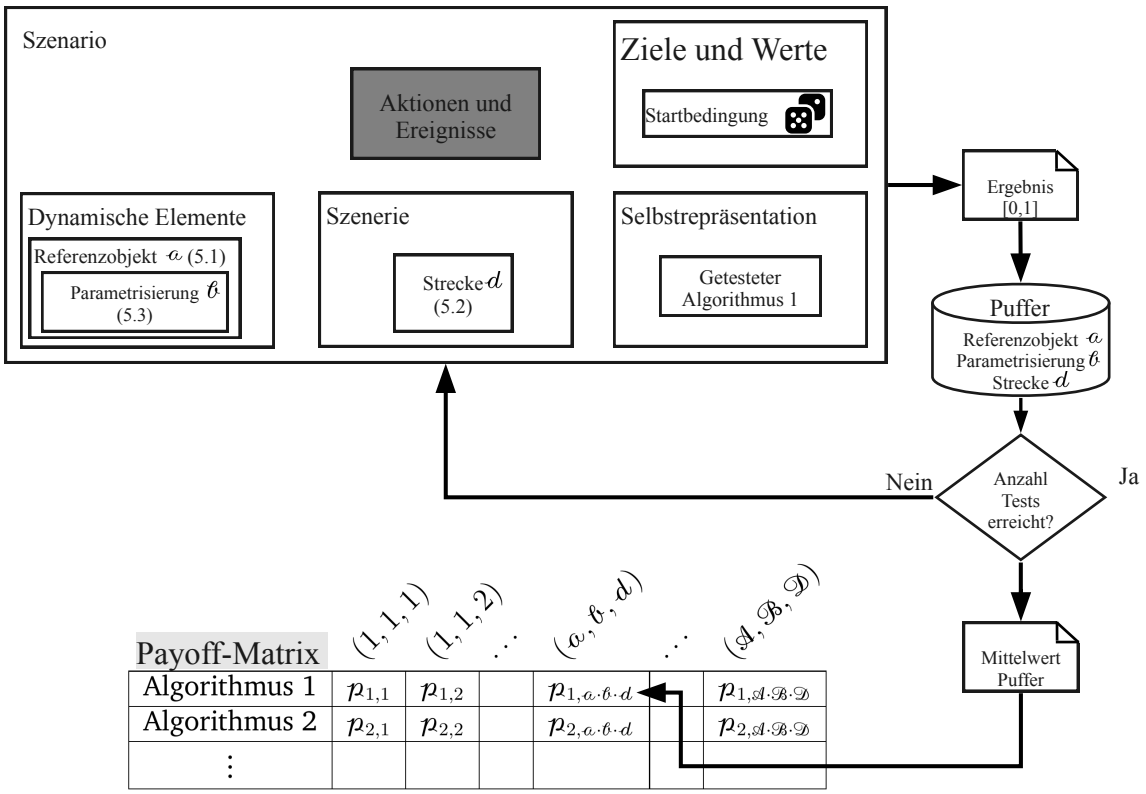


Abbildung 6-7.: Prinzip der Ermittlung der Payoff-Matrix

Die Ermittlung der Payoff-Matrix für das Ranking erfolgt dabei gemäß des in Abbildung 6-7 dargestellten Prozesses, der für jeden zu testenden Algorithmus durchgeführt wird. Aus den vorangegangenen Kapiteln wird der zu testende Algorithmus gegen ein Referenzobjekt $a \in [1, \mathcal{A}]$ mit der Parametrisierung $b \in [1, \mathcal{B}]$ auf der Strecke $d \in [1, \mathcal{D}]$ getestet. Die Startposition für den Algorithmus ist dabei zufällig auf der Strecke gewählt. Das Ergebnis des Tests wird in einem Puffer festgehalten und der Test so lange wiederholt, bis eine vorher festgelegte Anzahl Tests erreicht ist. Der Eintrag in der Payoff-Matrix entspricht dann dem Mittelwert aller getesteten Ergebnisse für das Szenario (a, b, d). Für eine vollständige Payoff-Matrix wird dieser Test für alle Variationen von $\mathcal{A}$ Referenzobjekten, $\mathcal{B}$ Parametrisierungen und $\mathcal{D}$ Strecken wiederholt.

Lässt sich die Payoff-Matrix $\mathbf{p}$ nicht ermitteln oder ist ein kompletter Testplan nicht gewünscht, besteht die weitere Option, Einträge in der Matrix zu schätzen. Andernfalls dürfen sich nur vollständige Reihen oder Spalten in der Payoff-Matrix befinden. In diesem Fall wäre ein Test zwischen mehreren Agenten nur möglich, wenn die Entwickler ihre Agenten für andere frei zugänglich machen. Alternativ dazu ist möglich, eine Agent-gegen-Aufgabe-Teststruktur zu wählen, die Agenten also nur gegen Referenzobjekte, nicht aber gegen andere Agenten zu testen. Mit der Option, Einträge in $\mathbf{p}$ zu schätzen, ließe sich das Verfahren flexibel erweitern, wenn eine größere Anzahl an Agenten frei zugänglich wird. Ebenfalls ist auf diese Weise möglich, eine Rangfolge zu errechnen, wenn von anderen Teilnehmern lediglich die Benchmark-Ergebnisse und nicht die Agenten selbst zur Verfügung stehen. Das Prinzip ist in Abbildung 6-8 dargestellt.

Abbildung 6-8.: Benchmark Prinzip zum Testen neuer Agenten.
bekannt, mglw. bekannt, testbar, unbekannt.

Für die Beantwortung von Forschungsfrage 3 wurde im Rahmen dieser Arbeit das mElo-Verfahren nach Balduzzi umgesetzt. Dieses ist in der Lage, eine Rangfolge anhand einer Payoff-Matrix zu berechnen auch unter der Annahme, dass die Payoff-Matrix intransitive Relationen enthält. Eine voll besetzte Payoff-Matrix ist dabei aber keine Grundvoraussetzung.

6.6. Umsetzung und Eigenschaften

Das von Balduzzi vorgestellte multidimensionale Elo-Verfahren[98] ist zunächst auf ein Ranking innerhalb einer Spieler-vs.-Spieler-Modellierung ausgelegt und wird erst in einem zweiten Schritt um die Möglichkeit des Spieler-vs.-Aufgabe-Rankings erweitert. Die Funktionsweise des Verfahrens wird in Abschnitt 6.6.1 vorgestellt. Abschnitt 6.6.2 erweitert das Verfahren gemäß Balduzzi, um auch Aufgaben in der Modellierung mit zu berücksichtigen. In Abschnitt 6.6.3 wird ein Prozess vorgestellt, der eine Schätzung von unbekannten Einträgen in der Payoff-Matrix erlaubt. Danach beleuchtet Abschnitt 6.6.4 die daraus resultierenden Eigenschaften für den hier präsentierten Anwendungsfall und deren Einfluss auf das Design des Benchmarkings.

[98] Vgl. Balduzzi, D. et al.: Re-evaluating Evaluation (2018).

6.6.1. Funktionsweise des multidimensionalen Elo-Rankings

Für das multidimensionale Elo-Verfahren wird davon ausgegangen, dass für eine Anzahl Spieler $\mathcal{N}$ ein Ranking-Vector $\boldsymbol{r}$ existiert, wobei $r_n \in \mathbb{R}$ das individuelle Ranking von Spieler n bezeichnet. Am Anfang des Verfahrens wird $\boldsymbol{r}$ für alle Spieler identisch initialisiert. Um neben den eindimensionalen transitiven Beziehungen auch die Abbildung von intransitiven Beziehungen zu ermöglichen, wird ebenfalls die Interaktionsmatrix $\mathcal{C}$ mit den Dimensionen $2k \times \mathcal{N}$ definiert, der Parameter k bezeichnet die Dimension des Verfahrens. Je höher die Dimension k, desto komplexere, intransitive Relationen sind durch das Verfahren abbildbar. Diese Matrix wird zum Anfang der Berechnungen mit zufälligen Werten initialisiert.[99] Dabei bezeichnet $\mathcal{C}_n$ den Reihenvektor der n-ten Reihe. Die Payoff-Matrix ergibt sich an dieser Stelle zu:

$$\boldsymbol{p} = \text{grad}(\boldsymbol{r}) + \mathcal{C}^\top \boldsymbol{\Omega} \mathcal{C} \tag{6-2}$$

$$\boldsymbol{\Omega} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & \ddots & \\ 0 & 0 & & \ddots \end{bmatrix} \text{ der Dimension } 2k \times 2k \tag{6-3}$$

$$\text{grad}(\boldsymbol{r}) = \boldsymbol{r}\mathbf{1}^\top - \mathbf{1}\boldsymbol{r}^\top \tag{6-4}$$

$$\mathbf{1} = [1, 1, ..., 1] \text{ mit Länge } \mathcal{N} \tag{6-5}$$

Die Matrix $\boldsymbol{\Omega}$ stellt eine mathematische Vereinfachung für Berechnung der Updates dar. Für die Dimension $k = 0$ ergibt sich an dieser Stelle das normale Elo-Verfahren. In diesem Fall fällt der Term $\mathcal{C}^\top \boldsymbol{\Omega} \mathcal{C}$ weg, da $\boldsymbol{\Omega}$ für $k = 0$ nicht definiert ist. Für ein Update bei einem Spiel von Spieler n gegen Spieler m ergibt sich der Erwartungswert für die Begegnung p_{nm} zu.

$$p_{nm} = \text{sigmoid}\left(\frac{\ln(10)}{400}\left(r_n - r_m + \mathcal{C}_n^T \cdot \boldsymbol{\Omega} \cdot \mathcal{C}_m\right)\right) \tag{6-6}$$

$$\text{mit sigmoid}(x) = \frac{1}{1 + \mathrm{e}^{-x}} \tag{6-7}$$

Ein Update für Spieler n ergibt sich nach der Begegnung mit dem Ausgang des Spiels p^*_{nm} zu

[99] An dieser Stelle ist anzumerken, dass in der Literatur bislang keine genaue Zahlenmenge definiert ist, aus der die zufälligen Werte für die Initialisierung ermittelt werden, sodass die Zeit für die Konvergenz des Verfahrens minimal wird. Die Initialisierung im Rahmen dieser Arbeit erfolgt daher durch zufällige Werte aus $[-10, 10]$.

$$\text{Update } \boldsymbol{r}_n = \eta_r(\boldsymbol{p}^*_{nm} - \boldsymbol{p}_{nm}) \tag{6-8}$$

$$\text{Update } \boldsymbol{\mathcal{C}}_n = \eta_{\mathcal{C}}(\boldsymbol{p}^*_{nm} - \boldsymbol{p}_{nm})\boldsymbol{\Omega}\boldsymbol{\mathcal{C}}_m \tag{6-9}$$

η_r repräsentiert dabei die Lernrate für das Ranking $\boldsymbol{r}$ und $\eta_{\mathcal{C}}$ die Lernrate der Interaktionsmatrix $\boldsymbol{\mathcal{C}}$.

6.6.2. Erweiterung multidimensionales Elo auf Spieler-vs.-Aufgabe-Modellierung

Um das mElo-Verfahren zu erweitern, wird zusätzlich zur Anzahl $\mathcal{N}$ der Spieler angenommen, dass $\mathcal{M}$ Aufgaben existieren. Diese Aufgaben werden ebenfalls als Spieler modelliert. Damit vergrößert sich die Payoff-Matrix von $\mathcal{N} \times \mathcal{N}$ auf die Dimension $(\mathcal{N} + \mathcal{M}) \times (\mathcal{N} + \mathcal{M})$. Alle weiteren Aspekte des Verfahrens bleiben unverändert. Das Update des Rankings $\boldsymbol{r}$ und der Interaktionsmatrix $\boldsymbol{\mathcal{C}}$ verändert sich lediglich insofern, dass die Spielpartner eingeschränkt werden, sodass $n \in [1, \mathcal{N}]$ und $m \in]\mathcal{N}, \mathcal{N} + \mathcal{M}]$ gilt.

6.6.3. Anwendung für die Prognose von Benchmark-Ergebnissen

Um mithilfe des Verfahrens einzelne Einträge von $\boldsymbol{p}$ zu schätzen, wird folgender Prozess angewendet. Die Basis bildet an dieser Stelle eine Payoff-Matrix $\boldsymbol{p}$. Dabei besteht die Annahme gemäß Abbildung 6-8, dass nicht alle Einträge dieser Payoff-Matrix bekannt sind. Mögliche Gründe dafür sind, dass bspw. ein Agent nicht frei zugänglich ist. In dem Fall ist es nicht möglich, einen bestimmten Test durchzuführen. Für jeden Agenten n beschreibt die Menge $\mathbb{K}_n$ alle Tests, für die der Erwartungswert p_{nm} für den Test unbekannt ist. In diesem Fall werden nur Updates ausgeführt, für die $m \notin \mathbb{K}_n$ gilt.

Grundvoraussetzung, dass es möglich ist, einen Eintrag in $\boldsymbol{p}$ zu schätzen, ist, dass die Relation durch andere Einträge in der Matrix abgebildet wird. Damit wird die Bedingung gestellt, dass die Information, die an dieser Stelle fehlt, in der Matrix verfügbar ist. Ist diese Bedingung erfüllt, liefert eine einzelne Wiederholung des mElo-Verfahrens einen Schätzwert, der allerdings nicht konsistent mit dem korrekten Wert übereinstimmt. Diese Tatsache ist bedingt durch (a) die zufällige Initialisierung von $\boldsymbol{\mathcal{C}}$ und (b) die einseitigen Updates für $\boldsymbol{\mathcal{C}}_n$ wenn $\mathbb{K}_n \neq \{\}$ gilt.

Für alle unbekannten Einträge der Payoff-Matrix wird aus diesem Grund zusätzlich ein geschätzter Wert $p_{nm,\text{est}}$ eingeführt. Die unbekannten Einträge von $\boldsymbol{p}$ werden geschätzt, indem das mElo-Verfahren mehrmals hintereinander durchgeführt wird. Während des Verfahrens wird wie für die ursprüngliche Methode immer zufällig eine Partie von Spieler n gegen Aufgabe m ausgewählt. Ist der Ausgang der Partie p^*_{nm}

bekannt, erfolgt das Update von $\boldsymbol{r}$ und $\mathcal{C}$ nach Gleichung (6-8) und (6-9) und dem Ausgang der Begegnung p^*_{nm} gemäß der Payoff-Matrix. Gilt aber $m \in \mathbb{K}_n$, wird der Ausgang der Partie zufällig ermittelt mit der angenommenen Wahrscheinlichkeit $p_{nm,\text{est}}$ für den Sieg.

Die Konvergenz wird gewährleistet durch ein Update des geschätzten Wertes $p_{nm,\text{est}}$ nach jeder Durchführung des mElo Verfahrens mit der Lernrate η_{est}.

$$\text{Update } p_{nm,\text{est}} = \eta_{\text{est}}(p_{nm} - p_{nm,\text{est}}) \tag{6-10}$$

Der Prozess ist in Algorithmus 1 zusammengefasst.

Algorithmus 1 Berechnungsprozess für fehlende Elemente der Payoff-Matrix

```
Initialisierung aller unbekannten Werte p_{nm,est} für alle n für die K_n ≠ {} gilt
for Anzahl Wiederholungen mElo do
    Initialisierung des Rankings r auf identische Werte
    Initialisierung der Interaktionsmatrix C auf zufällige Werte
    for Anzahl Wiederholungen Spiele pro mElo do
        Ermittlung einer zufälligen Begegnung zwischen Spieler n und Aufgabe m
        if m ∉ K_n then
            Ausgang der Begegnung p*_{nm} entspricht Eintrag in der Payoff-Matrix P
        else
            Ausgang der Begegnung p*_{nm} ist zufällig 0 oder 1 mit der Gewinnwahr-
            scheinlichkeit p_{nm,est}
        end if
        Updates von r und C gemäß Gleichungen (6-8) und (6-9)
    end for
    Update aller p_{nm,est} gemäß Gleichung (6-10)
end for
```

Die weitere Details zur algorithmischen Umsetzung und die notwendigen Hyperparameter finden sich in Anhang B.3.

6.6.4. Eigenschaften bezüglich Benchmarking und Intransitivität

Das vorgestellte Verfahren verfolgt das Ziel, fehlende Teile der Payoff-Matrix $\boldsymbol{P}$ anhand der anderen Einträge der Matrix zu ermitteln. Dieser Abschnitt untersucht daher, welche Genauigkeit mit diesem Verfahren erzielt wird. Die realen Ergebnisse dieser Arbeit beinhalten eine hohe Anzahl an Tests und deren Intransitivität ist daher nicht eindeutig quantifizierbar. Die Eigenschaften der Methode werden daher anhand eines

Tabelle 6-4.: Payoff-Matrix Schere-Stein-Papier-Feuer-Wasser.

	Schere	Stein	Papier	Feuer	Wasser
Schere	0,5	0	1	0	1
Stein	1	0,5	0	0	1
Papier	0	1	0,5	0	1
Feuer	1	1	1	0,5	0
Wasser	0	0	0	1	0,5

leicht analysierbaren theoretischen Beispiels diskutiert. Die Eigenschaften von mElo bezüglich der Vorhersagegüte bezüglich gewöhnlicher intransitiver Relationen und bekannter Payoff-Matrix $\boldsymbol{p}$ wurden bereits von Lazaridis[100] eingehend untersucht.

Er beschreibt dessen Eigenschaften für die intransitiven Spiele Schere-Stein-Papier (SSP) und Schere-Stein-Papier-Feuer-Wasser (SSPFW). SSPFW ist dabei eine Erweiterung des bekannten Spiels SSP, wobei Feuer alle anderen Strategien schlägt außer Wasser, Wasser dagegen verliert gegen alle anderen Strategien außer Feuer. Die Payoff-Matrix des Spiels ist in Tabelle 6-4 gezeigt.

Lazaridis zeigt, dass für eine steigende Anzahl an intransitiven Interaktionen auch eine höhere Dimension k notwendig ist. Das Verfahren ist aber in der Lage, die komplexe intransitive Interaktion der Strategien von SSPFW mit der Dimension $k = 2$ mit minimalen Abweichungen (maximale absolute Abweichung pro Eintrag $<1\,\%$) abzubilden.

Im Folgenden wird jeweils davon ausgegangen, dass je ein einzelner Eintrag aus $\boldsymbol{p}$ nicht bekannt ist. Wird das Schätzverfahren aus Algorithmus 1 auf die Payoff-Matrix $\boldsymbol{p}$ in Tabelle 6-4 mit einem fehlenden Eintrag angewendet, konvergiert das Verfahren zunächst zu zufälligen Werten, da bislang keine redundanten Informationen vorhanden sind, aus denen eine Schätzung der fehlenden Einträge möglich ist. Eine Voraussetzung für die Konvergenz des Verfahrens zu den realen Werten ist, dass die intransitive Relation in einem anderen Test oder Agenten bereits vorhanden ist.

Um die Genauigkeit des Verfahrens zu testen, wurde die Payoff-Matrix aus Tabelle 6-4 um jeweils einen bereits vorhandenen Agenten bzw. eine bereits vorhandene Aufgabe ergänzt, sodass das Verfahren Informationen aus diesen Einträgen nutzt, um maskierte Einträge zu schätzen. Danach wurden einzelne Einträge maskiert und die entsprechenden Werte mittels des vorgestellten Algorithmus 1 geschätzt. Eine beispielhaft modifizierte Payoffmatrix ist in Tabelle 6-5 dargestellt. Das Ergebnis ist, dass die fehlenden Einträge bis auf eine Genauigkeit von $\pm 2{,}5\,\%$ geschätzt werden. Eine volle Auflistung der Ergebnisse findet sich in Anhang B.3.

[100]Lazaridis, D.: An R package for multidimensional Elo ratings (2021).

Tabelle 6-5.: Beispielhaft erweiterte Payoff-Matrix Schere-Stein-Papier-Feuer-Wasser. Der markierte Wert wird als unbekannt angenommen

	Schere	Stein	Papier	Feuer	Wasser
Schere	0,5	0	1	0	1
Stein	1	0,5	0	0	1
Papier	0	1	0,5	0	1
Feuer	1	1	1	0,5	0
Wasser	0	0	0	1	0,5
Wasser	0	0	0	1	0,5

Um eine Abhängigkeit von den speziellen binären Zahlenwerten der präsentierten Matrix auszuschließen, wurden ebenfalls Varianten der Matrix getestet, die eine "schwache" Payoff-Struktur besitzen. Es wird also nicht davon ausgegangen, dass bei einer Begegnung Spieler n mit einer Wahrscheinlichkeit von 1 gewinnt oder 0 verliert, sondern es wird eine fixe Wahrscheinlichkeit für den Sieg $\in [0{,}5, 1]$ und für eine Niederlage $\in [0, 0{,}5]$ angenommen. Die Ergebnisse der Schätzung sind dabei von ähnlicher Güte.

Weitere Test mit dem Verfahren zeigten allerdings auch die Einschränkung der Methode. Das Verfahren verliert an Präzision, wenn der Anteil an Information/Einträgen in der Payoff-Matrix zunimmt, die auf den zu schätzenden Parameter keinen Einfluss besitzen. Dies liegt daran, dass in diesem Fall auch der Einfluss der benötigten Information in der Interaktionsmatrix $\mathscr{C}$ sinkt. Ebenso zeigt das Verfahren eine starke Abhängigkeit der Ergebnisqualität von den gewählten Hyperparametern, wie den Lernraten.

6.6.5. Fazit

Die vorgestellte Methode für die Approximation von einzelnen Einträgen in der Payoff-Matrix $\boldsymbol{p}$ zeigt sich darin erfolgreich für den abstrakten und einfachen Anwendungsfall des Spiels SSPFW einzelne Einträge in $\boldsymbol{p}$ zu schätzen. Das Verfahren hat allerdings limitierende Faktoren. Dazu zählt der Informationsgehalt in $\boldsymbol{p}$. Es ist notwendig, dass Informationen, die die unbekannte intransitive Relation beschreiben, in ausreichendem Maße vorhanden sind, damit die Methode in der Lage ist, unbekannte Werte zu schätzen. Dazu kommt, dass das Verfahren allein auf den Erwartungswerten der einzelnen Partien p_{nm} beruht und keinerlei Informationen über die Agenten an sich verwendet.

Für die Anwendung auf eine Payoff-Matrix mit einer Vielzahl an Tests und Agenten ist nicht möglich, sicherzustellen, dass (a) in der Payoff-Matrix die notwendigen Informationen vorhanden sind und (b) die vorhandenen Informationen die Relation wirklich korrekt abbilden.

Die Schätzung von Einträgen in der Payoff-Matrix wird daher im weiteren Verlauf dieser Arbeit nicht weiter verfolgt. Stattdessen wird als notwendige Voraussetzung für einen Algorithmen-übergreifenden Test die Anforderung formuliert, dass das Testframework an alle gängigen Methoden der Trajektorienberechnung anpassbar ist. Auf diese Weise wird sichergestellt, dass keine Inkompatibilitäten zwischen einzelnen Algorithmen das Testen einer Algorithmen-Kombination verhindern. Ebenfalls resultiert daraus die Forderung, dass alle Vergleichsobjekte, die im Rahmen der Tests genutzt werden, eine Trajektorie ausgeben, die möglicherweise von zu testenden Algorithmen für die Planung benötigt wird.

Das vorgestellte Benchmarking folgt damit einer Spieler-vs.-Aufgabe-Struktur. Die Tests, die im Benchmark durchgeführt werden, sind ein fester Satz an Aufgaben, für die jeweils möglich ist, eine Gewinnhäufigkeit oder mittleren Erfolg über Tests zu ermitteln. Damit ergibt sich, dass das Benchmarking zum einen unabhängig bezüglich der Anzahl an Agenten ist. Es ist frei skalierbar, da nicht notwendig ist, jeden Agenten gegen jeden anderen Agenten zu testen. Die Anzahl an Tests sind für jeden Agenten identisch und der gesamte Testaufwand steigt linear mit der Anzahl an Agenten.

Die Rangfolge, die gebildet wird, gibt allerdings nicht den absoluten Rang eines Spielers n wieder, da sich der Payoff sowohl aus dessen Ranking r_n als auch aus dem Anteil der Interaktionsmatrix $\mathcal{C}_n$ zusammensetzt. Dies ist eine direkte Folge der Intransitivität der Interaktion und der Einfluss wächst mit einer steigenden Anzahl an Agenten/Tests.

6.7. Weitere Fähigkeiten

Neben der messbaren Performance auf der Strecke gibt es weitere Fähigkeiten, die ein Agent möglicherweise besitzt und die bei einem Überholmanöver oder gegen einen bestimmten anderen Agenten einen Vorteil bringen, die für einen Algorithmus ermittelbar sind. Diese Eigenschaften werden im Folgenden vorgestellt und entsprechende Tests vorgeschlagen, die eine Aussage über die Ausprägung der Fähigkeiten ermöglichen. Die untersuchten Fähigkeiten sind zum einen die Ausnutzung der Aerodynamik und die Anpassungsfähigkeit bei lernenden Algorithmen. Des Weiteren wird die sogenannte Ausnutzbarkeit (engl. Exploitability) diskutiert.

Die aerodynamischen Eigenschaften eines Rennfahrzeugs spielen während eines Rennens und insbesondere während des Überholens von anderen Fahrzeugen eine große Rolle. Sie sorgen während der Fahrt zum einen dafür, dass Fahrzeugen mit ausgeprägtem aerodynamischen Abtrieb in den Kurven möglich ist, mit deutlich höheren Querbeschleunigungen zu fahren als Straßenfahrzeuge.[101] Zum anderen resultiert

[101]Trzesniowski, M.: Rennwagentechnik - Gesamtfahrzeug (2019), S. 200f.

durch die hohen Abtriebskräfte auch ein erhöhter Luftwiderstand, der auf der Geraden das Fahrzeug verlangsamt und die Höchstgeschwindigkeit begrenzt.[102] Wenn zwei Fahrzeuge hintereinander im Windschatten fahren, werden beide Effekte abgeschwächt.[103] Auf der Geraden erhöht sich die Höchstgeschwindigkeit und in den Kurven sind mitunter nur geringere Geschwindigkeiten möglich. Im realen Fahrzeug führt dieser Effekt dazu, dass durch die geringeren Aufstandskräfte an der Hinterachse oft auch ein höherer Schlupf auftritt.[104] Dieser Schlupf begünstigt die Degradation der Reifen. Aufgrund der Modellierungstiefe der Simulation, die in dieser Arbeit verwendet wird, ist dieser Effekt allerdings nicht zu beobachten. Da bislang in der Literatur kaum quantifizierte Forschung für den Einfluss der Aerodynamik für die Kurvenfahrt von mehreren Fahrzeugen existiert, wird dieser Effekt im Rahmen dieser Arbeit nicht näher betrachtet.

Um die Fähigkeit eines Algorithmus zu überprüfen, mit dem Effekt des niedrigeren Luftwiderstands bei der Fahrt im Windschatten eines vorausfahrenden Fahrzeugs umzugehen, wird folgender Test vorgeschlagen. Ähnlich dem bereits vorgestellten Streckenlayout aus Abschnitt 6.2.1 wird ein Szenario mit einer unendlich langen Kurve mit einer zufällig gewählten, hohen Krümmung entsprechend der Testdefinition für das erste Segment S1 aus Abschnitt 6.2.1 aufgebaut, bei dem der Algorithmus gegen einen Gegner antritt. Der Gegner wird an einer zufälligen lateralen Position initialisiert und hält diese laterale Position konstant über die gesamte Testdauer. Das hat den Grund, dass in einer unendlich langen Kurve die schnellste Linie so weit innen ist wie möglich, da hier der Weg am kürzesten ist. Ein Algorithmus, der keine Rücksicht auf ein vorausfahrendes Fahrzeug nimmt, würde daher mit sehr hoher Wahrscheinlichkeit diesen Weg nehmen. Die genutzte Leistungsmetrik ist bei diesem Versuch zum einen die verfügbare Vortriebsleistung des Gegner-Fahrzeugs, bei der ein Agent noch in der Lage ist, zu überholen. Zum anderen ist es möglich, den mittleren lateralen Abstand zwischen Ego-Fahrzeug und Gegner-Fahrzeug zu nutzen, um ein Fahren im Windschatten des Agenten zu identifizieren. Als Vergleichsreferenzen dienen dafür zum einen der LCADM- oder der Referenzalgorithmus, da diese keinen Mechanismus beinhalten, der ein Fahren im Windschatten erzeugt. Die beiden Algorithmen stellen damit die unteren Werte der Skala dar. Jede Fahrt im Windschatten geschieht in dem Szenario daher zufällig. Als obere Grenze der Skala wurde ein Agent für das Szenario entwickelt, der immer versucht, genau hinter dem vorne fahrenden Fahrzeug zu fahren und dann auf einer zufälligen Seite zu überholen. Eine genaue Funktionsbeschreibung des Algorithmus findet sich in Anhang B, die genauen Testbedingungen in Anhang D.3.

[102] Frömmig, L.: Grundkurs Rennwagentechnik (2019), S. 175ff.

[103] Džijan, I. et al.: Aerodynamic characteristics of two slipstreaming race cars (2021); Dominy, R. G.: The Influence of Slipstreaming on the Performance of a Grand Prix Racing Car (1990).

[104] Katz, J.: Aerodynamics of race cars (2006).

Neben der Nutzung der Aerodynamik für einen Überholvorgang besitzen vor allem lernende Algorithmen die Fähigkeit, sich an ein gegebenes Szenario anzupassen. Diese Fähigkeit wird im Folgenden als Adaptierbarkeit bezeichnet. Bei lernenden Algorithmen ist möglich, dass diese Fähigkeit dazu führt, dass die Performance sich deutlich steigert, wenn der Algorithmus die Gelegenheit erhält, direkt in einem bestimmten Szenario zu trainieren. Ebenfalls beinhalten manche neuere neuronalen Netzstrukturen zusätzliche Elemente, die wie ein Gedächtnis fungieren. Auf diese Weise ist einem Algorithmus möglich, während des Lernprozesses zu erlernen, ein Szenario systematisch zu erkunden und damit auch ohne erneute Lern-Rekursionen sich an ein Szenario anzupassen.[105] In diesem Fall wird von “Meta-Learning“ gesprochen, da die Algorithmen lernen “zu lernen“.

Karakovski und Togelius schlagen für ihr Benchmarking vor, dafür Tests zu erstellen, die die menschlichen Entwickler vorher nicht kennen. Der Agent hat dann im Zuge des Benchmarkings 10000 Versuche, auf dem Szenario zu trainieren. Das Ergebnis des 10001ten Versuchs wird danach bewertet und geht in die Punktzahl des Benchmarkings ein.[106] Ein derartiger Test wird aber an dieser Stelle nicht vorgeschlagen, da insbesondere ML-Algorithmen eine starke Abhängigkeit des Lernergebnisses von den Hyperparametern des Lernprozesses aufweisen. Des Weiteren sind die Hyperparameter zwischen verschiedenen Methoden nicht miteinander vergleichbar. Der in dieser Arbeit verwendete Proximal-Policy-Optimization-Algorithmus lernt bspw., indem er so viele Samples wie möglich generiert und nach jeder Lerniteration seine Erfahrungsspeicher lehrt.[107] Ein Soft-Actor-Critic-Algorithmus dagegen nutzt jeden berechneten Frame in mehreren Lernzyklen, um die Effizienz der Lernumgebung zu maximieren.[108] Ein fester und bewerteter Vergleich von Algorithmen, der jeden Algorithmus gleichermaßen unter vergleichbaren Bedingungen einschließt, ist daher nicht präzise definierbar und wird an dieser Stelle nicht weiter verfolgt.

Die Nutzung des hier vorgestellten Benchmarkings ist aber dennoch von Interesse für die Entwicklung intelligenter Algorithmen, da die Anwendung der hier vorgestellten Tests ermöglicht, die Performance ähnlicher Algorithmen untereinander zu vergleichen. Damit lässt sich bspw. die Effizienz eines bestimmten Lernverfahrens bestimmen, indem Rankings von verschiedenen evolutionären Stadien eines lernenden Algorithmus verglichen werden.[109] So ist die Beurteilung des Lernfortschritts bezogen auf die reale Aufgabe und nicht in Bezug auf das Lernszenario möglich. Ein solches Verfahren ist für

[105] Vgl. Nagabandi, A. et al.: Learning in Dynamic, Real-World Environments through Meta-RL (2018); vgl. Rakelly, K. et al.: Efficient Off-Policy Meta-Reinforcement Learning via Probabilistic Variables (2019).

[106] Vgl. Karakovskiy, S. et al.: The Mario AI Benchmark and Competitions (2012).

[107] Vgl. Schulman, J. et al.: Proximal Policy Optimization Algorithms (2017).

[108] Vgl. Haarnoja, T. et al.: Soft Actor-Critic (2018).

[109] Vgl. Vinyals, O. et al.: AlphaStar: Mastering the Real-Time Strategy Game StarCraft II (2019).

komplexe Aufgaben von besonderem Interesse, da im Bereich der Fahrzeugführung und des Fahrzeugverhaltens vermehrt auf Lern-Curricula gesetzt wird.[110]

Ein Algorithmus ist aber nicht nur in der Lage, selbst zu lernen. Ein wichtiger Teil bei der Modellierung eines Agenten in einer kompetitiven Umgebung ist auch, ob bzw. wie schnell ein anderer Algorithmus in der Lage ist, sich dem Agenten anzupassen und Strategien findet, um einen Vorteil gegenüber dem Agenten zu erzielen.[111] Diese Eigenschaft wird als **Exploitability** (deutsch Ausnutzbarkeit) bezeichnet. Die Notwendigkeit dafür resultiert direkt aus der intransitiven Relation verschiedener Strategien, da die Performance nicht durch Maximierung eines einzelnen Kriteriums optimiert wird.[112] Eine Agent, der, oder eine Strategie, die eine geringe Exploitability besitzen, sind damit robust gegen eine möglichst große Anzahl anderer Strategien.[113] Die Bedeutung dieser Robustheit wird von Veröffentlichungen wie von Vinyals[114] besonders hervorgehoben, die beim Training ihres Alpha-Star-Algorithmus für das Computerspiel StarCraft gezielt Agenten in ihr Training einbringen, die darauf trainiert werden, ausschließlich gegen die Strategie des Hauptalgorithmus zu spielen, um diese Robustheit gezielt zu erhöhen. Neuere Ansätze des Self-Play wie für das Spiel Go und Schach mit AlphaZero[115] oder für das Spiel Dota 2 mit OpenAIFive[116] basieren ebenfalls darauf, dass Algorithmen gegenseitig voneinander lernen und auch lernen, die Strategie des jeweils anderen auszunutzen. Eine Studie im Umfeld des automatisierten Fahrens von Cooper *et al.*[117] betont die Wichtigkeit dieser Eigenschaften, indem das Phänomen für ein einfaches Vorfahrt-Szenario untersucht wird. Die Autoren gehen insbesondere darauf ein, dass die ausgeprägten Sicherheitsmechanismen in automatisierten Fahrzeugen leicht durch menschliche Fahrer missbraucht werden, indem dem automatisierten Fahrzeug die Vorfahrt genommen wird. Um einen Unfall zu vermeiden, bleibt das automatisierte Fahrzeug stehen.

Um die Ausnutzbarkeit eines Agenten zu testen, ist ein ähnliches Vorgehen wie für die Adaptibility anwendbar. Dafür wird ein lernender Agent in einem festen Szenario trainiert. Das Gegner-Objekt in dem Szenario ist der zu untersuchende Algorithmus. Anhand der Evaluierung der Randbedingungen und des Erfolges des lernenden Algorithmus, ist möglich zu erkennen, ob die Strategie oder Methode des zu untersuchenden

[110] Vgl. Qiao, Z. et al.: Automatically Generated Curriculum based RL for Autonomous Vehicles (2018); vgl. Zhou, M. et al.: Scalable Multi-Agent RL Training School for Autonomous Driving (2020); vgl. Song, Y. et al.: Autonomous Overtaking in Gran Turismo Sport Using Curriculum RL (2021).

[111] Vgl. Vinyals, O. et al.: Grandmaster level in StarCraft II using multi-agent RL (2019).

[112] Vgl. Timbers, F. et al.: Approximate exploitability (2020).

[113] Vgl. Heinrich, J. et al.: Deep Learning from Self-Play in Imperfect-Information Games (2016).

[114] Vinyals, O. et al.: Grandmaster level in StarCraft II using multi-agent RL (2019).

[115] Vgl. Silver, D. et al.: A general RL algorithm that masters chess and Go through self-play (2018).

[116] Vgl. Berner, C. et al.: Dota 2 with Large Scale Deep Reinforcement Learning (2019).

[117] Cooper, M. et al.: Stackelberg Punishment and Bully-Proofing Autonomous Vehicles (2019).

Algorithmus ausnutzbare Schwachstellen besitzt. Ähnlich wie auch für die Adaptibility gilt auch für die Exploitability, dass eine Testdefinition aufgrund der Abhängigkeit der Hyperparameter der eingesetzten Lernverfahren ohne Bevorzugung einzelner Verfahren nicht möglich erscheint.

Der genutzte Maßstab ist an dieser Stelle ebenfalls für eine Bewertung hinderlich, da jeder Algorithmus von einem anderen Ausgangswert für den Erfolg in einem Szenario startet. Eine Verbesserung oder Verschlechterung während eines Lernprozesses ist daher immer nur in Relation zu einem vergleichbaren Ausgangswert zu betrachten. Die Exploitability ist daher ebenfalls kein fester Bestandteil des Rankings, das in dieser Arbeit vorgestellt wird.

7. Testumgebung und Algorithmen

Um eine Evaluation der in dieser Arbeit vorgestellten Methode zu ermöglichen, wurde ein Trajektorienplaner entwickelt, auf deren Basis mehrere Konzepte von Verhaltensalgorithmen umgesetzt wurden. Als Grundlage der Entwicklung stellt dieser Abschnitt zunächst den aktuellen Stand der Forschung der Trajektorienplanung in Abschnitt 7.1 und für die Berücksichtigung von anderen Objekten in Abschnitt 7.2 vor. Daran anschließend werden die hier vorgestellten Planungsalgorithmen in Abschnitt 7.3 motiviert und in Abschnitt 7.4.1-7.6 vorgestellt. Als Vergleichsmaßstab wurde ebenfalls ein Algorithmus aus der Literatur von Kloock *et al.*[118] für den Trajektorienplaner implementiert.

7.1. Stand der Forschung - Trajektorienberechnung

Eine Trajektorie beschreibt den Weg, dem ein Fahrzeug durch den Raum folgt, in Abhängigkeit von der Zeit. Die Berechnung einer Trajektorie kann durch die Anwendung von verschiedenen Optimierungsverfahren durchgeführt werden. Diese Optimierungsverfahren gliedern sich in drei wesentliche Unterkategorien.

- Graphensuche
- Modellprädiktive Regelung
- Neuronale Netze

Diese drei Kategorien werden im Folgenden zusammen mit relevanten aktuellen Veröffentlichungen und Erkenntnissen aus der Forschung vorgestellt.

7.1.1. Graphensuche

Eine der bekanntesten Methoden zur Berechnung von Trajektorien ist die Baum- oder Graphensuche. Diese Methode, die erstmals an der Universität in Stanford im Forschungsfeld der Robotik Anwendung fand[119], zeichnet sich dadurch aus, dass die Methode versucht, den kürzesten Weg zwischen zwei gegebenen Punkte in einer definierten Umgebung zu finden. Dabei wird zusätzlich noch zwischen Methoden unterschieden, die auf einer reinen Suche basieren (search based) und Methoden, die auf Stichproben basieren (sampling based). Die auf Suche basierenden Methoden sind dabei oft eine abgeleitete Form des sogenannten A*-Algorithmus.[120] Stichprobenbasierte Algorithmen dagegen basieren in den meisten Fällen auf RRT (Rapidly exploring

[118]Kloock, M. et al.: Networked Model Predictive Vehicle Race Control (2019).

[119]Vgl. Hart, P. E. et al.: A Formal Basis for the Heuristic Determination of Minimum Cost Paths (1968).

[120]Hart, P. E. et al.: A Formal Basis for the Heuristic Determination of Minimum Cost Paths (1968).

Random Tree)[121]. Beide Algorithmen wurden vielfach abgeändert und angepasst, um für den Anwendungsfall eines autonomen Zweispurfahrzeugs bessere Ergebnisse zu erzielen.

Ajanovic *et al.*[122] stellen einen hybriden Ansatz vor. Dieser basiert auf einem A*-Ansatz der um das hinterlegte Fahrdynamikmodell erweitert wurde. Es wird das Ziel verfolgt, die minimale Rundenzeit unter bekannten Streckenbedingungen mit niedrigem Reibkoeffizienten zwischen Reifen und Fahrbahn und ohne Berücksichtigung von anderen Fahrzeugen zu erreichen. Das hinterlegte Modell ist ein lineares Einspurmodell mit nichtlinearer Reifenkennlinie. Die Ergebnisse in einer "perfekten" Simulation zeigen, dass der Ansatz auch auf Strecken mit niedrigem Reibwert valide Trajektorien liefert. Die Simulation ist insofern "perfekt", da sie weder ein Fahrzeugmodell noch einen Regler nutzt. Das Fahrzeug bewegt sich, indem es nacheinander auf die nächste Pose der Trajektorie gesetzt wird. Die Autoren geben dabei keine Auskunft über die tatsächlich benötigte Rechenzeit.

Ein Ansatz basierend auf RRT wird von Arab *et al.*[123] vorgestellt. Hier wird versucht, die minimal mögliche Rundenzeit für eine Rennstrecke zu erreichen. Dabei wird eine Kombination aus einem verbesserten RRT-Algorithmus für die offline Planung der Trajektorie auf einer definierten Strecke und einem modellprädiktiven Regler (MPC) für die Regelung eines ferngesteuerten Versuchsfahrzeugs im Maßstab 1:7 eingesetzt. Die Ergebnisse der Trajektorienberechnung zeigen dabei im Vergleich zu anderen von den Autoren implementierten Methoden kürzere Rundenzeiten und auch die Regelung ist in der Lage, im Vergleich zu einem professionellen Fahrer schneller die kurze Strecke abzufahren. Die Methode ist aufgrund des rechenaufwendigen Suchalgorithmus nicht online im Fahrzeug einsetzbar.

In der Veröffentlichung von Bevilacqua *et al.*[124] werden drei verschiedene Optimierungsalgorithmen verglichen für die Aufgabe, eine möglichst schnelle Rundenzeit zu berechnen. Es werden ein genetischer Algorithmus, ein Optimierungsalgorithmus (fmincon) und ein Partikelschwarm miteinander verglichen. Die Strecke wird dabei in einzelne Sektionen eingeteilt, um die Rechenzeit zu minimieren. Dabei wird die Topologie der Strecke genutzt und eine Einteilung anhand der Kurven des Streckenlayouts vorgenommen. Die einzelnen Sektionen werden jeweils auf den geraden Abschnitten der Strecke zusammengefügt. Die Trajektorien werden anschließend anhand der errechneten Rundenzeiten evaluiert. Die Ergebnisse zeigen dabei lange Rechenzeiten zwischen 30 Minuten und mehreren Stunden für eine komplette Runde. Der Partikel-

[121] Lavalle, S. M.: Rapidly-Exploring Random Trees (1998).

[122] Ajanovic, Z. et al.: Search-Based Motion Planning for Performance Autonomous Driving (2019).

[123] Arab, A. et al.: Motion planning for aggressive autonomous vehicle maneuvers (2016).

[124] Bevilacqua, M. et al.: Particle swarm for path planning in a racing circuit simulation (2017).

schwarm erzielt in der Studie die kürzeste Rundenzeit, der genetische Algorithmus die langsamste.

Frego *et al.*[125] präsentieren einen modularen Ansatz, der das Problem, eine zeitoptimale Trajektorie für eine Strecke zu berechnen, in drei einzelne Probleme zerlegt. Zuerst wird ein Klothoid gesucht, der zwei geeignete Punkte auf der Strecke miteinander unter gegebenen Geschwindigkeiten verbindet, daran schließt sich eine Überprüfung auf Schnittpunkte mit etwaigen Objekten an. Die Überprüfung ist gefolgt von der Verbindung der einzelnen Klothoide zu einer gesamten Trajektorie. Die Methode zeigt sich in der Evaluation sowohl langsamer bezüglich Rechenzeit als auch Rundenzeit im Vergleich mit einem Referenz-Solver. Die Autoren geben aber zu bedenken, dass da es sich um ein suchbasiertes Verfahren handelt, die Robustheit höher sei als bei reinen MPCs. Es wird ebenfalls gezeigt, dass das Verfahren auch für statische Objekte auf der Strecke gute Ergebnisse liefert.

Für das Szenario einer Fahrt im Straßenverkehr zeigen Wei *et al.*[126] einen dreischichtigen Ansatz für die Trajektorienberechnung. Auf der obersten Ebene wird ein suchbasiertes Verfahren genutzt, um die Strategie bezüglich des zu befahrenden Fahrstreifens und Geschwindigkeit unter Berücksichtigung der geltenden Verkehrsregeln zu ermitteln. Im zweiten Schritt berechnet der Algorithmus eine Trajektorie, ohne dynamische Objekte in der Berechnung zu berücksichtigen. Daran anschließend werden die Regler-Ausgänge optimiert, um andere Objekte, die sich ebenfalls auf der Straße befinden, zu berücksichtigen. Die Ergebnisse zeigen den Ansatz gegenüber einfacheren stichprobenbasierten Methoden als überlegen, sowohl bezüglich Rechenzeit als auch erreichter Minimierung der Kostenfunktion. Die Methode wird von den Autoren auf einem funktionsfähigen Versuchsträger implementiert und ist in der Lage, ein statisches Objekt erfolgreich zu umfahren.

Die Methode von Werling *et al.*, die über mehrere Publikationen[127] weiterentwickelt wurde, zeigt, wie die Trennung des lateralen und longitudinalen Aktionsraums eines Fahrzeugs in einem Straßenszenario nutzbar ist, um effizient eine Vielzahl an Trajektorien zu generieren. Im ersten Schritt werden mehrere Sets von Eingaben für die longitudinale und laterale Bewegungsrichtung erzeugt, durch die Kombination der verschiedenen Eingaben errechnet sich danach ein Set aus vielen Trajektorien, die im Anschluss über die physikalischen Randbedingungen des Fahrzeugs, der Streckenbegrenzung und die Position der anderen Verkehrsteilnehmer gefiltert werden. Der Ansatz überzeugt in der Simulation mit guten Ergebnissen. Die Autoren geben aber an,

[125] Frego, M. et al.: Trajectory planning for car-like vehicles: A modular approach (2016).

[126] Wei, J. et al.: A behavioral planning framework for autonomous driving (2014).

[127] Ziegler, J. et al.: Navigating car-like robots using an obstacle sensitive cost function (2008);
Werling, M. et al.: Optimal trajectory generation for dynamic street scenarios (2010);
Werling, M. et al.: Optimal trajectories for time-critical street scenarios (2012).

dass es in dem von ihnen verwendeten Datensatz noch zu Kollisionen kommt aufgrund von später Wahrnehmung, falscher Prädiktion der anderen Verkehrsteilnehmer oder einem zu kurzen Planungshorizont.

Die Lösung auf die Frage einer geeigneten Renntrajektorie wird von Stahl *et al.*[128] mit einem knotenbasierten Suchansatz beantwortet. Dabei wird die gesamte Strecke jeweils in Längsabschnitte eingeteilt. Jeder Abschnitt besitzt gleichmäßig verteilt über der Streckenbreite mehrere Knoten. Um eine Online-Applikation des Algorithmus zu ermöglichen, werden für jede mögliche Verbindung zwischen den Knoten, die an aufeinander folgenden Längsabschnitten liegen, die verbindenden Pfade im Voraus berechnet. Das Fahrzeug ist dann über eine Kostenfunktion in der Lage, den Pfad zu finden, der am besten zu den physikalischen Grenzen des Fahrzeugs und etwaigen Hindernissen auf der Fahrbahn passt. Die Lösung zeigt eine gute Performance mit und ohne Hindernisse und funktioniert auch bei der Berücksichtigung langsam fahrender dynamischer Hindernisse. Die Lösung enthält aber nur eine sehr einfache Prädiktion anderer Verkehrsteilnehmer mit der Annahme konstanter Geschwindigkeit.

Der Ansatz von Kala *et al.*[129] sieht eine Trajektorienberechnung in vier Ebenen vor. Im Gegensatz zu anderen Veröffentlichungen wird ein Straßenszenario in einem Straßennetzwerk betrachtet, dass sowohl statische und dynamische Hindernisse enthält als auch die Annahme, dass Gegenverkehr nicht an einen Fahrstreifen gebunden ist. Es werden mehrere Fahrzeuge gleichzeitig im Straßennetzwerk berechnet. Die oberste Ebene der Planung ist die Routenfindung, also eine Graphensuche nach der kürzesten Route vom Start- zum Zielort. Im zweiten Schritt wird ein Pfad generiert, der den Weg durch das Straßennetzwerk beschreibt. Dabei werden statische Hindernisse und Wegblockaden berücksichtigt, allerdings keine dynamischen Objekte, indem eine angepasste Version eines RRT-Algorithmus angewendet wird. Als Auswahlkriterium für den Pfad wird die Länge des Pfades und die verfügbare Breite der jeweiligen Straßensegmente verwendet. In einem dritten Schritt wird für bestimmte Zeitpunkte die verfügbare Breite der Straße auf die zu diesem Zeitpunkt im Straßensegment vorhandenen Fahrzeuge aufgeteilt. Da es möglicherweise dazu kommt, dass zu viele Fahrzeuge gleichzeitig versuchen, denselben Streckenabschnitt zu nutzen, ist eine mögliche Rekursion auf den oberen Ebenen vorgesehen, um die Route des Fahrzeugs ggf. anzupassen. Erst in einem vierten Schritt werden die erzeugten Pfade geglättet, sodass sich für ein Zweispurfahrzeug fahrbare Pfade ergeben.

7.1.2. Modelprädiktive Regelung

Eine weitere Möglichkeit, eine Trajektorie zu berechnen, ist die modellprädiktive Regelung. Diesem Ansatz liegt immer eine bestimmte Modellvorstellung (bspw. ein

[128] Stahl, T. et al.: Multilayer Graph-Based Trajectory Planning for Race Vehicles (2019).

[129] Kala, R. et al.: Multi-Level Planning for Semi-autonomous Vehicles in Traffic Scenarios (2013).

Massenpunkt- oder Einspurmodell) zugrunde, mithilfe derer die Bewegungsgleichungen des Systems aufgestellt werden. Die Gleichungen werden für eine bestimmte Anzahl an Zeitschritten formuliert. Das Modell dient an dieser Stelle dafür, den Übergang zwischen den Zeitschritten zu definieren. Zusammen mit einem definierten Optimierungsziel ist möglich, eine Regelstrategie oder auch Trajektorie zu berechnen. Dabei werden unterschiedliche Diskretisierungen (örtlich und zeitlich) verwendet. Je nach Diskretisierung ändert sich die Komplexität des Problems, der Gleichungen und die Möglichkeit andere Fahrzeuge in der Optimierung mit zu berücksichtigen. Die Bewegungsgleichungen werden durch Randbedingungen eingeschränkt.

Das Optimierungsziel für die Trajektorienberechnung im Rennkontext ist häufig definiert, sodass ein möglichst langer Weg (bei zeitdiskreten Problemen) oder eine möglichst kurze Zeit (bei ortsdiskreten Problemen) innerhalb des Planungshorizontes erreicht wird. Die Randbedingungen werden genutzt, um die Bewegungsgleichungen einzuschränken (bspw. den Lenkwinkel), aber auch, um dem Fahrzeug den zur Verfügung stehenden Platz vorzugeben. Alle im Folgenden vorgestellten Veröffentlichungen beschäftigen sich, sofern nicht anders angegeben, mit dem Problem, eine Trajektorie für ein Rennfahrzeug zu berechnen.

In der Veröffentlichung von Alrifaee *et al.*[130] wird ein Optimierungsproblem vorgeschlagen, das auf einer sequenziellen Linearisierung und einem Massenpunktmodell basiert. Es werden für jeden Zeitschritt separate Gleichungen aufgestellt, indem der Zustand des Fahrzeugs und die Strecke auf Basis der vorher berechneten Lösung linearisiert werden. Die räumlichen Randbedingungen werden dabei über Halbräume realisiert, wie in Abbildung 7-1 abgebildet. Die Lösung wird in der Simulationssoftware IPG-Carmaker evaluiert und erreicht eine um $0{,}5\,\%$ schnellere Rundenzeit als der programminterne IPG-Racedriver. Die Zeit für die Trajektorienberechnung ist mit dem vorgestellten Ansatz kürzer als die zeitliche Schrittweite des Algorithmus. Eine Echtzeit-Implementierung ist daher mit der vorhandenen Konfiguration möglich.

[130] Alrifaee, B. et al.: Real-time Trajectory optimization for Autonomous Vehicle Racing (2018).

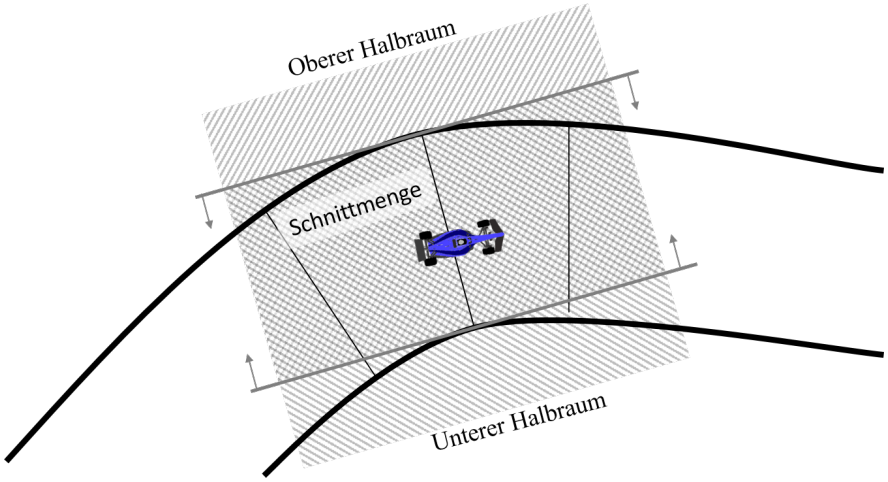


Abbildung 7-1.: Visualisierung Halbräume für einen einzelnen Zeitschritt. Der Raum, der für den Algorithmus zur Verfügung steht, ist die Schnittmenge des oberen und unteren Halbraumes.

Christ *et al.*[131] zeigen einen Ansatz, der ermöglicht, unterschiedliche Reibungskoeffizienten μ zwischen Reifen und Fahrbahn in verschiedenen Abschnitten der Fahrbahn zu berücksichtigen. Zur Berechnung der Trajektorie wird ein dynamisches Zweispurmodell verwendet und die Reifenkräfte werden durch die Pacejka-Magic-Formula[132] angenähert. Die Strecke selbst wird für die Berechnung als eine Gridmap dargestellt, sodass es möglich ist, für jeden Streckenabschnitt und auch variabel über der Breite der Strecke verteilt unterschiedliche μ darzustellen. Dadurch ist für jede Pose des Fahrzeugs jedem Reifen ein entsprechendes μ zuteilbar. Die simulative Evaluation zeigt, dass die Methode geeignet ist, um unterschiedliche Reibungsverhältnisse in der Planung zu berücksichtigen. In einer Simulation wird gezeigt, dass durch die Verwendung eines Zweispurmodells ein Streckenabschnitt mit lokal reduziertem μ derart durchfahren wird, dass der Bereich mit niedrigem μ sich zwischen der linken und rechten Spur des Fahrzeugs befindet und das Fahrzeug damit kein reduziertes μ an den Reifen erfährt. Die Autoren geben an, dass die Erweiterung von einem Ein- auf ein Zweispurmodell die Berechnungsdauer um $20\,\%$ verlängert. Eine Echtzeitberechnung der Trajektorie ist mit der Methode mit der gegebenen Hardware-Konfiguration nicht möglich, da die Berechnungsdauer für eine Runde mit $120\,\mathrm{s}$ knapp über der berechneten Rundenzeit von ca. $80\,\mathrm{s}$ liegt.

Liniger *et al.*[133] nutzen ein dynamisches Einspurmodell mit nichtlinearer Reifenkennlinie. Das Problem der Trajektorienberechnung wird darauf reduziert, einer Referenzlinie auf der Strecke zu folgen und gleichzeitig die zurückgelegte Strecke zu maximieren. Die Strecke wird dabei durch räumliche Randbedingungen in Form von Halbräumen

[131]Christ, F. et al.: Time-optimal trajectory planning considering variable tyre-road friction (2019).

[132]Vgl. Pacejka, H. B. et al.: The Magic Formula Tyre Model (1992).

[133]Liniger, A. et al.: Optimization-Based Autonomous Racing of 1:43 Scale RC Cars (2015).

eingegrenzt, sodass eine Berücksichtigung statischer Objekte durch Berechnung eines Korridors erfolgt, der durch die Strecke und darauf befindliche Objekte definiert ist. Ähnlich wie in der Methode von Alrifaee wird eine Linearisierung für jeden Zeitschritt durchgeführt, sodass die Lösung des Optimierungsproblems mit sehr kurzen Rechenzeiten möglich ist. Durch die Verwendung des nichtlinearen Reifenmodells ist es nötig, sehr kurze Zeitintervalle zu verwenden, da sonst durch die Linearisierung der nichtlinearen Kennlinien das Problem nicht mehr stabil lösbar ist. Zusätzlich sieht die Methode vor, jeden Zeitschritt mehrmals zu lösen, um die Stabilität der Berechnung zu erhöhen, was die Verwendung eines langen Planungshorizonts erschwert. Die Methode wird erfolgreich in Simulation und mit Miniaturfahrzeugen evaluiert.

Subosits *et al.*[134] präsentieren eine Möglichkeit, Trajektorien auf Basis eines MPC neu zu planen, um auf statische Ziele am Streckenrand zu reagieren. Für die Methode wird eine lineare Punktmasse als Modellvorstellung verwendet. Die Bewegung des Modells ist durch die Reibungsellipse und die mögliche longitudinale Beschleunigung des Fahrzeugs eingeschränkt. Die statischen Ziele, auf die das Fahrzeug reagiert, sind am Rand der Strecke platziert, sodass eine konvexe Beschreibung in der sequenziellen Linearisierung der Streckengeometrie weiterhin möglich ist. Mit einer maximalen Dauer von $20\,\mathrm{ms}$ für die Lösung des Problems ist die Methode auch in einem realen Fahrzeug anwendbar.

Plessen *et al.*[135] zeigen einen über den Weg diskretisierten Ansatz, um eine Trajektorie auf einer Strecke mit statischen Hindernissen zu planen. Die statischen Hindernisse sind hierbei als rechteckige Verengungen der Fahrbahn ausgeführt. Mithilfe von sequenzieller Linearisierung und der Minimierung der Kurvenkrümmung wird eine Trajektorie berechnet. Die Autoren verweisen darauf, dass dynamische Objekte zwar theoretisch modellierbar sind, aber durch die räumliche Diskretisierung einer Verortung der Objekte durch die nicht berücksichtige Zeitkomponente nur durch Prädiktion der Objekte und des eigenen Fahrzeugs möglich ist. Die Besonderheit des Ansatzes ist, dass durch die örtliche Diskretisierung und die damit verbundene genaue Beschreibung der Umgebung statische Objekte sehr eng umfahren werden.

Altche *et al.*[136] zeigen einen Ansatz auf Basis eines MPC, der mit einem dynamischen Zweispurmodell in der Lage ist, auch statische Objekte zu umfahren, die nicht am Streckenrand platziert wurden. Dafür wird für jedes Objekt eine umschließende Ellipse aufgespannt. Als Ziel für den Optimierungsalgorithmus wird eine maximale Geschwindigkeit bei gleichzeitig minimaler Abweichung von der Mittellinie gesetzt. Die Methode zeigt in der simulativen Evaluation, dass die Objekte erfolgreich umfahren werden.

[134]Subosits, J. et al.: Real-time Trajectory Replanning for Autonomous Driving (2019).

[135]Plessen, M. G. et al.: Trajectory Planning Under Vehicle Dimension Constraints Using SLP (2017).

[136]Altche, F. et al.: High-speed trajectory planning for autonomous vehicles (2017).

Das Ziel, eine zeitoptimale Trajektorie zu berechnen, wird aber nicht erreicht. Das Optimierungsziel, das Fahrzeug möglichst nah an der Mittellinie der Straße zu bewegen, führt dazu, dass das Fahrzeug sich in longitudinaler Richtung nur langsam den Objekten nähert. Hier zeigt sich, dass die Auswahl und auch die Parametrisierung des Optimierungsproblems einen starken Einfluss auf das Ergebnis besitzt.

7.1.3. Neuronale Netze

Die Methode, mithilfe von NN die Trajektorie bzw. den Pfad des Fahrzeugs durch ein neuronales Netz berechnen zu lassen, ist bislang nicht stark verbreitet. Daher stellt dieser Abschnitt bereits bestehende Ansätze aus dem allgemeinen Automobilbereich vor und beleuchtet auch, welche Forschungsarbeiten zu diesem Thema bereits im Bereich der Drohnen und Robotik existieren.

Wagener *et al.*[137] präsentieren in ihrer Veröffentlichung ein NN, das durch einen Lernprozess in der Lage ist, ein Fahrzeug auf einer Rennstrecke zu fahren. Die Trajektorie wird daher lediglich implizit durch das neuronale Netz berechnet. Die Ergebnisse zeigen, dass die erreichbare Rundenzeit stark von den gewählten Hyperparametern abhängt, die den Lernprozess definieren. Das Fahrzeug ist in der Lage, einen einfachen Kurs zu absolvieren. Es werden dabei keine hohen Geschwindigkeiten ($<2\,\frac{\mathrm{m}}{\mathrm{s}}$) erreicht.

Eine explizite Berechnung der Trajektorie wird von Garlick und Bradley[138] durchgeführt. Sie lernen ein NN mithilfe eines großen Datensatzes an, eine Renntrajektorie zu berechnen. Dabei werden immer nur einzelne kleine Abschnitte für jede Strecke berechnet. Das NN wird über Supervised Learning auf die Lösungen trainiert, die zuvor mit einem MPC-Ansatz berechnet wurden. Das NN erhält als relevante Features jeweils die relativen Veränderungen der Streckensegmente zueinander und der Optimierungsalgorithmus versucht die Ausgabe des Netzes auf die zuvor berechnete Lösung zu optimieren. Die Ergebnisse zeigen, dass die Methode die Funktion des zugrunde liegenden MPCs ersetzt. Die Genauigkeit ist zu dem verwendeten MPC-Ansatz vergleichbar, die Zeit, die benötigt wird, ist dabei aber nur ein Bruchteil ($33\,\mathrm{ms}$ im Mittel pro Strecke) eines MPCs und zeigt auch nur ein unterproportionales Wachstum mit steigender Streckenlänge.

Ein Ansatz für die Trajektorienberechnung in einem Rennszenario bieten Weiss *et al.*[139] mit einer direkten Trajektorienausgabe. Das NN wird trainiert, die Trajektorie in Form von Stützpunkten einer Béziers-Kurve auszugeben. Diese erstellte Planung wird in einer tieferen Ebene von einem Pure-Pursuit-Regler abgefahren. Das Netz wird auf Basis von Bilddaten trainiert und verwendet als Fehlermetrik die Unterschiede in (a)

[137] Wagener, N. et al.: An Online Learning Approach to Model Predictive Control (2019).

[138] Garlick, S. et al.: Real-Time Trajectory Planning Using Machine Learning (2021).

[139] Weiss, T. et al.: DeepRacing: Parameterized Trajectories for Autonomous Racing (2020).

Position und Geschwindigkeit des Fahrzeugs und (b) Position der gesetzten Wegpunkte des Algorithmus. Das Netz wird über Supervised Learning angelernt. Es erkundet daher die Umgebung nicht selbstständig, sondern wird lediglich trainiert, die korrekten Outputs für gegebene Inputs zu liefern. Die Autoren vergleichen ihren Ansatz der Trajektoriengenerierung über Béziers-Kurven mit einer direkten Generierung von Trajektorienpunkten und zwei Varianten der direkten Steuerung durch je ein NN auf Basis des PilotNet-Ansatzes von NVidia[140]. Lediglich der von den Autoren vorgestellte Ansatz schafft es, in einer Closed-Loop-Simulation in einer Rennsimulation eine Runde erfolgreich zu beenden, wobei der Bézier-Kurven-Ansatz eine Runde in deutlich kürzerer ($-12\,\%$) Zeit zurücklegt. Die Performance liegt dennoch deutlich unter ($-20\,\%$) der eines menschlichen Fahrers.[141]

Im Bereich der Quadrocopter angesiedelt, zeigt Ates[142] einen Algorithmus auf Basis von Maschinellem Lernen (ML) mit integrierter Trajektorienplanung auf Basis von Reinforcement Learning (RL). Das NN erkundet durch Regelung einer virtuellen Drone selbstständig die Umgebung, die Dronen-Rennstrecke, und erhält je nach der Güte des erreichten Zustands einen Reward. Anhand des Rewards optimiert der ML-Algorithmus das NN. Als Reward dient dabei bspw. der Abstand zum nächsten zu durchfliegenden Checkpoint. Die Aufgabe des Agenten ist in dem Szenario, einen bestimmten Kurs zu absolvieren. Es ist daher notwendig, dass das NN selbstständig seine eigene Trajektorie plant. Die Autoren identifizieren als komplexesten Teil der Aufgabe die Definition der Reward-Funktion, die maßgeblich das erreichte Verhalten des Algorithmus beeinflusst.

Die Veröffentlichung von Butyrev *et al.*[143] stellt einen Algorithmus vor, der einen nichtholonomen Roboter auf einer 2-dimensionalen Fläche steuert. Der Roboter hat die Aufgabe, eine bestimmte Position im Raum zu erreichen. Das NN wird über RL trainiert und ist bereits nach einem kurzen Trainingsprozess in der Lage, mehrere dynamisch auftauchende Zielpositionen erfolgreich zu erreichen. Das NN berechnet also nicht direkt eine Trajektorie, es übernimmt die Aufgabe von Planung und Regelung und gibt indirekt eine Trajektorie aus. Die abgefahrene Trajektorie wird anschließend mit einer Spline-Interpolation der anzusteuernden Punkte verglichen. Die Trajektorie des NN ist im Gegensatz zur Spline-Interpolation vollständig kompatibel mit der Kinematik des Roboters.

Der Ansatz von Lei *et al.*[144] konzentriert sich auf ein dynamisches Szenario eines nichtholonomen Roboters. Die verwendeten Szenarien bestehen aus einer 2D-Umgebung

[140]Vgl. Bojarski, M. et al.: End to End Learning for Self-Driving Cars (2016);
vgl. Bojarski, M. et al.: End-to-End Trained Deep Neural Network Steers a Car (2017).

[141]Vgl. Michael Romanidis: Codemasters F1 2019 Australia World Record (2019).

[142]Ates, U.: Long-Term Planning with Deep Reinforcement Learning on Autonomous Drones (2020).

[143]Butyrev, L. et al.: Deep Reinforcement Learning for Motion Planning of Mobile Robots (2019).

[144]Lei, X. et al.: Dynamic Path Planning of Unknown Environment Based on Deep RL (2018).

mit statischen Hindernissen und kleinen dynamischen Objekten. Der Roboter bewegt sich kontinuierlich durch die Umgebung und besitzt zur Wahrnehmung der Umgebung einen virtuellen Abstandssensor für eine 360 Grad rundum Ansicht. Die Start- und Zielpositionen werden für das Lernen des Algorithmus mittels RL zufällig variiert. Die Evaluierung mit einem Miniaturfahrzeug mit Abstandssensor zeigt ein erfolgreiches Erreichen der Zielposition in der aufgebauten Umgebung.

Ein kombinierter Ansatz wird von Bae *et al.*[145] verfolgt. Sie nutzen einen A*-Algorithmus als Referenz für die Kostenfunktion, um ein NN per RL anzulernen. Das Ziel des Roboters ist, in einem 2D-Grid mittels diskreter Bewegungen von einem zufälligen Startpunkt die Zielposition zu erreichen. Input für den Roboter ist dabei eine Ansicht des gesamten Spielfeldes und der darin enthaltenen Hindernisse und Objekte. Das Spielfeld ist wie ein Labyrinth aufgebaut, sodass der kürzeste Weg nicht trivial zu finden ist. In der Evaluation schafft der Algorithmus in dynamischen Szenarien, den A*-Algorithmus hinsichtlich Rechenzeit zu schlagen. Aufgrund der sehr einfachen Umweltrepräsentation in einem digitalen Grid findet der A*-Algorithmus immer die kürzeste Route, was in einem kontinuierlichen Szenario nur für eine Rechenzeit, die gegen unendlich geht, der Fall ist.[146]

7.1.4. Fazit

Graphensuche ist eine rechenaufwendige Methode, eine passende Trajektorie zu berechnen. Entweder werden die Graphen so erzeugt, dass direkt eine valide Trajektorie entsteht[147] oder die gewählte Trajektorie wird im Nachhinein geglättet.[148] Werden andere Fahrzeuge berücksichtigt, steigt die Komplexität des Problems weiter an, da in diesem Fall (a) eine Prädiktion der anderen Verkehrsteilnehmer notwendig wird, um die Bewegung der anderen Fahrzeuge in der eigenen Trajektorie zu berücksichtigen, (b) für jeden Zeitschritt eine Prüfung notwendig ist, ob es für eine hypothetische Trajektorie zu einer Kollision kommt oder (c) eine Kostenfunktion auf Basis eines Prädiktionsmodells aufgestellt und evaluiert wird. Diese Problematik zeigt sich bereits bei der Berücksichtigung von einem einzelnen dynamischen Verkehrsteilnehmer[149] und steigt mit zunehmender Teilnehmerzahl je nach verwendeten Algorithmen in unterschiedlichem Ausmaß. Der Effekt wird beeinflusst von (a) der Anzahl der für das Ego-Fahrzeug erzeugten Trajektorien, (b) des Rechenaufwandes für die Prädiktion der anderen Objekte, (c) der Art der Umfeldrepräsentation, (d) der Länge des Zeitho-

[145]Bae, H. et al.: Multi-Robot Path Planning Method Using Reinforcement Learning (2019).

[146]Vgl. Paden, B. et al.: Survey of Motion Planning and Control Techniques for Urban Vehicles (2016).

[147]Vgl. Stahl, T. et al.: Multilayer Graph-Based Trajectory Planning for Race Vehicles (2019).

[148]Vgl. Kala, R. et al.: Multi-Level Planning for Semi-autonomous Vehicles in Traffic Scenarios (2013).

[149]Vgl. Liniger, A. et al.: A Non-Cooperative Game Approach to Autonomous Racing (2017).

rizonts, (e) der Implementierung durch die jeweiligen Autoren[150], den numerischen Solvern bzw. der Komplexität der gewählten Modellierung und (f) weiteren von der jeweiligen Methode abhängigen Faktoren.

Nach der Generierung des Sets aus Trajektorien erfolgt deren Bewertung. Je nach Komplexität der Methode ist dann ebenfalls ein Optimierungsproblem zu lösen. Die Auswahl der Trajektorie erfolgt dann wie auch bei einem MPC oder NN durch eine Kostenfunktion, die bestimmt, welche Trajektorie die "optimale" ist. Die Methode bestimmt in diesem Fall die Komplexität der Kostenfunktion und damit, wie rechenaufwendig die Berechnungen sind. Die Bandbreite reicht dabei von einer einfachen Prüfung über den größten Fortschritt einer Trajektorie auf der gewünschten Strecke zu komplexen Berechnungen, die versuchen, durch die Auswertung der Prädiktion eines anderen Fahrzeugs, dieses Fahrzeug am Überholen zu hindern.[151]

Bei der Aufstellung der Gleichungen für einen MPC gibt es einige Punkte zu beachten. Es werden in den meisten Fällen nur sehr einfache Modellvorstellungen verwendet, da mit steigender Komplexität der Gleichungen auch die Rechenzeiten entsprechend länger werden.[152] Ein komplexes Bewegungsmodell führt wie im Fall von Liniger möglicherweise dazu, dass zusätzliche Berechnungen notwendig sind, sodass die Methode nicht instabil wird.[153] Daher ist ein Ziel, die Darstellung der Umgebung möglichst einfach zu halten. Das führt zu zwei wesentlichen Aspekten. In nahezu allen Veröffentlichungen werden zeitliche Diskretisierungen verwendet, eine örtliche Diskretisierung erlaubt zwar eine präzise Ausnutzung des vorhandenen Platzes[154], allerdings ist nicht möglich, damit komplexe Streckengeometrien als konvexe Räume darzustellen.[155] Bei Nutzung einer zeitlichen Diskretisierung wird die Strecke daher über je eine Randbedingung am linken bzw. rechten Rand der Strecke begrenzt. Daher erlauben die meisten MPC-Ansätze die Berücksichtigung statischer Objekte, die oft am Streckenrand platziert werden. (Siehe Studien von Gutjahr[156], Zubača[157], Kloeser[158] und Yi[159]) In diesem Fall ist lediglich eine Verschiebung der Streckenabgrenzung notwendig, um das Hindernis zu umfahren. Wie von Liniger *et al.*[160] beschrieben

[150]Der Vergleich von verschieden Veröffentlichungen zeigt, dass je nach Implementierung die Rechenzeiten sich um mehrere Größenordnungen unterscheiden.

[151]Vgl. Liniger, A. et al.: A Non-Cooperative Game Approach to Autonomous Racing (2017).

[152]Vgl. Christ, F. et al.: Time-optimal trajectory planning considering variable tyre-road friction (2019).

[153]Vgl. Liniger, A. et al.: A Non-Cooperative Game Approach to Autonomous Racing (2017).

[154]Vgl. Plessen, M. G. et al.: Trajectory Planning Under Vehicle Dimension Constraints Using SLP (2017).

[155]Vgl. Alrifaee, B. et al.: Real-time Trajectory optimization for Autonomous Vehicle Racing (2018).

[156]Gutjahr, B. et al.: Lateral Vehicle Trajectory Optimization Using Linear MPC (2016).

[157]Zubača, J. et al.: Smooth Reference Line Generation for a Race Track with Gates (2020).

[158]Kloeser, D. et al.: NMPC for Racing Using a Singularity-Free Model with Obstacle Avoidance (2020).

[159]Yi, B.: Integrated Planning and Control for Collision Avoidance Systems (2018).

[160]Liniger, A. et al.: Optimization-Based Autonomous Racing of 1:43 Scale RC Cars (2015).

ist für den Fall, dass Objekte über der Breite der Strecke verteilt sind, ein weiteres Problem zu lösen. Zuerst wird versucht, einen Korridor durch die Objekte zu finden, der vom Fahrzeug durchfahren werden kann. Die Komplexität dieser Aufgabe steigt entsprechend an, wenn es darum geht, auch dynamische Objekte zuzulassen, da dann der Korridor für eine prädizierte Bewegung der Objekte bestimmt wird und auch nicht immer garantiert ist, dass überhaupt ein Korridor existiert, der in der Lage, ist alle Zeitschritte miteinander in der Raumdimension zu verbinden. In diesem Fall wird im Ansatz eine alternative Strategie benötigt, damit es nicht zur Kollision von Objekten kommt.

Die Implementierung von Altche *et al.*[161] zeigt auch, dass die Kostenfunktion des Optimierungsproblems selbst bereits von großer Bedeutung ist. Wie die Studie zeigt, ist es möglich, dass die Gewichtung der einzelnen Faktoren, die in der Kostenfunktion berücksichtigt werden, dazu führt, dass das resultierende Verhalten nicht den Erwartungen entspricht. Diese Problematik verschärft sich mit steigender Anzahl an Einflussfaktoren in der Kostenfunktion.

Die vorgestellten Ansätze zur Nutzung von neuronalen Netzen zeigen, dass mit Ausnahme der Implementierung von Weiss[162] und Garlick[163] neuronale Netze weniger für die Trajektorienberechnung und mehr für die direkte Steuerung der Roboter oder Fahrzeuge genutzt werden. Eine Trajektorie wird vom NN nur implizit berechnet, indem es den Pfad abfährt, der zum größten Reward für den Algorithmus führt. Hier ist daher die gewählte oder gelernte Reward-Funktion von Bedeutung und einer der Faktoren mit dem größten Einfluss auf das resultierende Verhalten des Fahrzeugs.[164]

Die vorgestellten Ansätze zeigen hauptsächlich ein Training über RL. Ein möglicher Grund liegt in der Umfeldrepräsentation. Je komplexer die Repräsentation, desto wahrscheinlicher ist, dass das Fahrzeug in Situationen gerät, die für das System unbekannt sind. Die Folge ist dann eine undefinierte Reaktion.[165] Es ist daher notwendig, dass Algorithmen, die für die Regelung von Fahrzeugen genutzt werden, in so unterschiedlichen Umgebungen genutzt werden, wie es die Simulationsumgebungen und auch die verwendete Repräsentation zulassen bzw. erfordern.[166]

Dieser Trend zeigt sich auch in vermehrten aktuellen Studien wie bspw. von Fuchs

[161]Altche, F. et al.: High-speed trajectory planning for autonomous vehicles (2017).

[162]Weiss, T. et al.: DeepRacing: Parameterized Trajectories for Autonomous Racing (2020).

[163]Garlick, S. et al.: Real-Time Trajectory Planning Using Machine Learning (2021).

[164]Vgl. Ates, U.: Long-Term Planning with Deep Reinforcement Learning on Autonomous Drones (2020); vgl. Wiewiora, E.: Reward Shaping (2010), S. 863ff.

[165]Vgl. Cang Ye et al.: A fuzzy controller with supervised learning assisted RL (2003).

[166]Vgl. Tobin, J. et al.: Domain Randomization for Transferring Deep NN to the Real World (2017).

et al.[167] und Song *et al.*[168], die für das Spiel Grand Tourismo Sport einen Agenten im Spiel über RL trainieren und damit eine Performance für ein einzelnes Szenario erzielen, die die Fähigkeiten eines menschlichen Fahrers übersteigt. Im Falle von Song *et al.* sind im Szenario auch andere Fahrzeuge mit involviert. Der resultierende Agent beschränkt sich auf ein einzelnes definiertes Szenario und ist nicht allgemeingültig einsetzbar. Eine weitere Veröffentlichung, die sich mit dem Thema von RL für die Regelung von Rennfahrzeugen beschäftigt, wird von Herman *et al.*[169] präsentiert. Sie stellen eine 3D-Simulationsumgebung vor, die direkt ein Rennszenario abbildet und in der ein Training von NN, für die Aufgabe, ein Rennfahrzeug zu steuern, möglich ist. Eine Implementierung von Multi-Agent-Szenarien ist zum Zeitpunkt dieser Veröffentlichung von den Autoren angestrebt, aber noch nicht implementiert.

Die Tatsache, dass die Trajektorien nicht direkt berechnet werden, ist aber auch von Bedeutung für sogenannte kooperative Algorithmen. Diese Planungsalgorithmen basieren darauf, dass ihnen als Grundlage für die eigene Planung von den anderen Fahrzeugen die berechnete Trajektorie oder Regelgrößen zur Verfügung gestellt werden. Auf diese Weise ist den Fahrzeugen durch die gemeinsame Planung möglich, Kollisionen zu vermeiden. Solche Konzepte sind mit einer impliziten Trajektorienplanung nur sehr begrenzt vereinbar. Es bedeutet aber auch, dass für die Bewertung eines Verhaltensalgorithmus nicht möglich ist, die berechnete Trajektorie zu nutzen, da sie nicht in jeder Ausführung von Verhaltensalgorithmus vorliegt. Nur die tatsächlich abgefahrene Trajektorie ist nutzbar, um die Algorithmen zu bewerten.

Der größte Vorteil, den die Ansätze auf Basis von NN bieten, ist ihre schnelle Rechenzeit, wie die Studie von Garlick und Bradley[170] zeigt. Die Performance wird dabei begünstigt durch die Nutzung von potenten Grafik-Recheneinheiten, die für den Einsatz der üblichen numerischen Solver nicht nutzbar sind und der hohe Grad an Parallelisierung, die ML-Bibliotheken ermöglichen.

7.2. Stand der Forschung - Berücksichtigung von Objekten auf der Fahrbahn

Nachdem die Optionen vorgestellt wurden, die für eine Trajektorienberechnung in Frage kommen, gibt der folgende Abschnitt einen Einblick in die Möglichkeiten, wie andere Verkehrsteilnehmer in verschiedenen Szenarien in der Trajektorienplanung berücksichtigt werden.

[167] Fuchs, F. et al.: Super-Human Performance in Gran Turismo Sport Using Deep RL (2020).

[168] Song, Y. et al.: Autonomous Overtaking in Gran Turismo Sport Using Curriculum RL (2021).

[169] Herman, J. et al.: Learn-to-Race (2021).

[170] Garlick, S. et al.: Real-Time Trajectory Planning Using Machine Learning (2021).

7.2.1. Anwendungen

Gerdts *et al.*[171] stellen eine Methode für die Berechnung von Trajektorien in verschiedenen Straßenszenarien vor. Die Trajektorienberechnung folgt einem kooperativen Ansatz, bei dem die umliegenden Verkehrsteilnehmer (VT) jeweils über die geplante Trajektorie der umliegenden VT informiert sind. Die VT werden in dem Ansatz nach Priorität kategorisiert. Für die Kategorisierung werden zum einen Verkehrsregeln verwendet und zum anderen, wie weit ein Fahrzeug von seinem eigentlichen Plan abweichen müsste, um einem anderen Fahrzeug Vorrang zu geben. Der Raum, den ein Fahrzeug einnimmt, wird als eine Ellipse modelliert, in die das Fahrzeug nicht hineinfahren darf. In der Evaluation wird in mehreren Szenarien erfolgreich gezeigt, wie die Fahrzeuge nicht eindeutige Situationen ohne Kollision durchfahren. Dadurch, dass die Fahrzeuge einem bereits vordefinierten Pfad durch das Straßennetzwerk folgen, sorgt die Einbeziehung der anderen Fahrzeuge lediglich zu einem Abbremsen oder einem geringen Versatz vom ursprünglich geplanten Pfad.

Die Veröffentlichung von Gutjahr *et al.*[172] zeigt einen über der Zeit diskretisierten MPC. Die Bewegung anderer Fahrzeuge wird über ein nicht spezifiziertes Bewegungsmodell prädiziert. Anhand der prädizierten Position zu einem bestimmten Zeitpunkt wird der lateral zur Verfügung stehende Platz für den MPC eingeschränkt. In der Evaluation in einem Versuchsfahrzeug ist mit diesem Ansatz nicht nur die Fahrt in einem Szenario mit statischen Objekten möglich, sondern auch in einem einfachen Szenario mit einem dynamischen Objekt.

Im suchbasierten Ansatz von Rizano *et al.*[173] wird für die berechnete Trajektorie des eigenen Fahrzeugs überprüft, ob sie sich mit der eines anderen Fahrzeugs überschneidet. Ist das der Fall, wird für den Abschnitt, in der die Überschneidung stattfindet, der Suchalgorithmus erneut angewendet, bis eine valide Lösung gefunden ist. Laut Aussage der Autoren funktioniert diese Art des Überholens nur mit deutlich langsameren Gegner-Fahrzeugen.

Stahl *et al.*[174] passen die Suche nach einer Trajektorie derart an, dass die Bewegung eines anderen Fahrzeugs mittels eines einfachen Bewegungsmodells prädiziert und die zugehörigen Knoten im Suchraum des Algorithmus entfernt werden. Eine Schwierigkeit dabei ist, dass die Knoten zunächst in der Raumdimension vorliegen. Die Information, wann ein Fahrzeug sich an einer bestimmten Position befindet, ist aber zeitabhängig. Dies gilt nicht nur für das prädizierte vorausfahrende Fahrzeug, auch die longitudinale Position des eigenen Fahrzeugs auf der Strecke ist abhängig von der eigenen Linienwahl.

[171] Gerdts, M. et al.: Motion Planning in Driving Scenarios with Communicating Vehicles (2018).

[172] Gutjahr, B. et al.: Lateral Vehicle Trajectory Optimization Using Linear MPC (2016).

[173] Rizano, T. et al.: Global path planning for competitive robotic cars (2013).

[174] Stahl, T. et al.: Multilayer Graph-Based Trajectory Planning for Race Vehicles (2019).

Ein präziser Abgleich zwischen Position von eigenem und fremdem Fahrzeug stellt sich daher mit dieser Methode als schwierig dar. Die Autoren beschränken sich daher in der Evaluation auf ein langsam geradeaus-fahrendes Objekt.

In der Studie von Inoue *et al.*[175] wird eine Methode vorgestellt, die, um die Sicherheit von automatisierten Fahrfunktionen zu erhöhen, auf Basis einer Sicherheitsmetrik die Trajektorie des Fahrzeugs bestimmt. Anhand der Sicherheitsmetrik wird das genutzte Potenzialfeld für die Trajektorie des Fahrzeugs angepasst. Hindernisse auf der Straße gelten dabei automatisch als Sicherheitsrisiko. Die Kosten, die für die Trajektorie berechnet werden, sind daher bei der Vorbeifahrt an einem Objekt besonders hoch, wenn der Abstand zum Objekt nur sehr klein ist. Das führt dazu, dass bspw. viel Platz zu einem Objekt gelassen wird, wenn das Fahrzeug daran vorbeifährt. Die Methode ähnelt daher dem Ausschließen des Raumes, in dem ein Fahrzeug sich nach der Prädiktion der Bewegung befinden wird. Die dargestellte Methode zeigt in der Evaluation gute Ergebnisse in einfachen Szenarien und ist in der Lage, das Risiko für schwere Kollisionen zu verringern.

Liniger *et al.*[176] zeigt in einer Studie, wie mithilfe von Spieltheorie zwei Fahrzeuge zusammen auf einer Rennstrecke ohne Kollisionen fahren. Das Prinzip ist, dass für beide Fahrzeuge jeweils ein Set aus möglichen Trajektorien erzeugt wird. In einem zweiten Schritt wird über eine Kostenfunktion ein valides Paar Trajektorien bestimmt. Liniger betrachtet zur Lösung der Trajektorienauswahl zwei verschiedene Gleichgewichte aus der Spieltheorie.

Das Stackelberg-Gleichgewicht beschreibt einen Zustand, in dem ein führender Spielteilnehmer zuerst seinen Zug wählt und im Anschluss daran der zweite Teilnehmer in Kenntnis der Entscheidung des vorangegangenen Spielers die optimale Entscheidung trifft.[177] Übertragen auf ein Rennen ergibt sich damit ein sequenzielles Szenario und beinhaltet, dass nur das hintere Fahrzeug versucht, eine Kollision mit dem vorderen Fahrzeug zu vermeiden. Das vordere Fahrzeug schenkt dem Hinteren keine Beachtung, da es im Motorsport in diesem Fall “Vorfahrt“ besitzt.

Das Nash-Gleichgewicht stellt sich ein, wenn beide Spielteilnehmer gemessen an der Entscheidung des jeweils anderen Teilnehmers die ideale Entscheidung treffen.[178] Übertragen bedeutet es, dass ein Trajektorienpaar die Kostenfunktionen für beide Fahrzeuge minimiert. Das zweite Szenario unterscheidet sich daher vom Ersten insofern, dass auch vom vorne fahrenden Fahrzeug Kollisionen in der Kostenfunktion berücksichtigt werden. Im dritten Szenario wird der Fortschritt des hinteren Fahrzeugs

[175]Inoue, H. et al.: Intelligent Driving System for Safer Automobiles (2017).

[176]Liniger, A. et al.: A Non-Cooperative Game Approach to Autonomous Racing (2017).

[177]Vgl. Simaan, M. et al.: On the Stackelberg strategy in nonzero-sum games (1973).

[178]Vgl. Facchinei, F. et al.: Generalized Nash equilibrium problems (2007).

auf der Strecke mit Kosten belegt. Dadurch versucht das vorne fahrende Fahrzeug aktiv, das andere Fahrzeug am Überholen zu hindern.

Die Methode ist aufgrund der vielen notwendigen Kollisionsberechnungen nicht echtzeitfähig, da für jeden Zeitschritt und jede Trajektorienkombination eine Kollisionsprüfung erfolgt. Die Auswertung der Kollisionen zeigt, dass bei Anwendung der Methode Kollisionen mit $0{,}5\,\%$ nur selten vorkommen. Die Anzahl der Überholvorgänge bei den drei Szenarien zeigt, dass bei der vorgestellten Methode das Ignorieren des hinten fahrenden Fahrzeugs zu den wenigsten Überholmanövern führt. Dabei fällt bei einem Vergleich von einem eher defensiven mit einem aggressiven Fahrstil auf, dass beim aggressiven Fahrstil im ersten Szenario nahezu keine ($<1\,\%$) Überholmanöver stattfinden. Die vollständige Einhaltung der Randbedingungen zur Vermeidung von Kollisionen führt also dazu, dass bei einem aggressiven vorausfahrenden Fahrzeug ein Überholen erschwert wird.

Auf Basis von Liniger's Ansatz stellen Kloock *et al.*[179] ihre Methode zur Berücksichtigung von anderen Fahrzeugen vor. Dabei werden wie auch für die Strecke lineare Randbedingungen genutzt, um den Abstand zwischen zwei Fahrzeugen auf einem gesetzten Mindestwert zu halten. Als Basis für die Berechnung der Randbedingungen dient jeweils die Lösung des vorangegangenen Zeitschritts, sodass sich die Lösungen jeweils gegenseitig beeinflussen. Die Ergebnisse zeigen, dass die Methode in der Lage ist, nacheinander mehrere langsamere Fahrzeuge ohne Kollision zu überholen.

Hubmann *et al.*[180] präsentieren eine Methode auf Basis eines Markov-Entscheidungsprozesses (engl. Markov Decision Process, MDP). Dabei werden über ein Prädiktionsmodell die Bewegungen und Interaktionen der anderen Verkehrsteilnehmer vorausgesagt und in Form einer Unschärfe-Aussage in der Trajektorienberechnung berücksichtigt. Durch die Vielzahl an Vorhersagen über den zu fahrenden Pfad eines anderen VT (bspw. fährt das andere Fahrzeug geradeaus oder biegt es links ab und welche Geschwindigkeit wählt es dabei), wählt das Fahrzeug laut den Autoren konservativere Trajektorien. Das führt z. B. dazu, dass eine Entscheidung über den Fahrstreifenwechsel mit einer Verzögerung des Fahrzeugs hinausgezögert wird, um zu einem späteren Zeitpunkt mit höherer Sicherheit eine Entscheidung zu treffen.

Huang *et al.*[181] zeigen mit ihrer Methode, wie es möglich ist, ML zu nutzen, um ein anderes Fahrzeug in einer Rennsimulation zu berücksichtigen. Dabei wird im Rennsimulator TORCS ein NN trainiert, zum einen, um Überholverhalten zu ermöglichen und zum andern um zu lernen, ein anderes Fahrzeug am Überholen zu hindern. Dazu werden die Bilddaten des Simulators als Eingang für das NN benutzt. Diese Methode

[179] Kloock, M. et al.: Networked Model Predictive Vehicle Race Control (2019).

[180] Hubmann, C. et al.: Planning With Interaction and Uncertain Prediction (2018).

[181] Huang, H.-H. et al.: Learning overtaking and blocking skills in simulated car racing (2015).

ermöglicht dadurch, beliebig viele andere Fahrzeuge in der Planung der Trajektorie mit einzubeziehen. Die Ergebnisse der Arbeit zeigen, dass mit einem anderen Fahrzeug auf der Strecke mit der Methode ein Überholen ermöglicht wird. Die Autoren verwenden dabei ein deutlich langsameres Gegner-Fahrzeug, sodass eine tiefgehende Bewertung nicht durchführbar ist. Das Netz lernt ebenfalls erfolgreich andere Fahrzeuge am Überholen zu hindern. Allerdings zeigen die Daten, dass die verfolgte Strategie dabei weniger prädizierend ist und das Verhalten des vorne fahrenden Fahrzeugs ausnutzt, sondern das NN lediglich versucht, in den Weg des deutlich schnelleren Überholers zu fahren und so ein Überholmanöver zu verhindern.

Der Ansatz von Alrifaee[182] zeigt einen verteilten MPC. Jedes Fahrzeug löst in dem Ansatz nur sein eigenes Optimierungsproblem. Dabei werden die einzelnen VT für jeden Zeitschritt zunächst kategorisiert, sodass ein Fahrzeug nur die Randbedingungen anderer VT berücksichtigt, wenn die VT (a) eine höhere Priorität besitzen und (b) sich in der Nähe des Fahrzeugs befinden. Dadurch wird die Rechenzeit in komplexen Szenarien reduziert. Da die Berechnung der Trajektorie allerdings immer auf der Trajektorie der anderen VT des letzten Zeitschrittes basiert, stellt Alrifaee fest, dass eine strikte Rangfolge von Fahrzeugen in seltenen Fällen zu kritischen Situationen führt und er erweitert den Ansatz um parallele Prioritäten. Diese minimieren den Zeitverzug, unter dem die Fahrzeuge ihre Trajektorie planen. Um jedem Fahrzeug eine Priorität zuzuweisen, schlägt Alrifaee einen Verhandlungsalgorithmus für eine dezentrale Trajektorienberechnung vor, der dafür sorgt, dass die VT sich untereinander auf die jeweilige Priorität einigen, ohne dabei eine konkrete Implementierung vorzustellen. Er gibt zu bedenken, dass eine ideale Prioritätenvergabe (a) die Summe der Kostenfunktionen aller Fahrzeug minimiert und (b) für möglichst viele gleich gewichtete Fahrzeuge sorgt, um Rechenzeit einzusparen. Die parallele Berechnung von Trajektorien ist insofern zeitsparend, da ein Fahrzeug nicht auf die Trajektorie des Fahrzeugs mit höherer Priorität wartet. Abschließend vergleicht Alrifaee den Ansatz mit einem Ansatz kooperativer (also gleichzeitiger Berechnung der Trajektorien aller VT) und stellt dabei fest, dass gerade für Szenarien mit vielen VT eine verteilte Berechnung in vielen Fällen zwar keine optimale Lösung liefert, aber dafür in Echtzeit ein suboptimales Ergebnis ermöglicht.

Die Kombination aus einem NN und einem regelbasierten Verhaltensalgorithmus wird von Likmeta *et al.*[183] vorgestellt. Dabei ist das Ziel, die Transparenz eines ML-Ansatzes zu erhöhen, indem der Algorithmus anstatt das Fahrzeug eigenständig zu steuern, nur zwischen verschiedenen händisch erstellten Regelmodi wechselt. Dadurch wird die Entscheidung, die der Algorithmus trifft, für einen Menschen einfacher zu interpretieren, da bspw. die Trajektorienwahl nicht implizit durch den Algorithmus geschieht,

[182]Alrifaee, B.: Vernetzte modellbasierte prädiktive Regelung zur Kollisionsvermeidung (2017).

[183]Likmeta, A. et al.: Combining RL with rule-based controllers for autonomous driving (2020).

sondern eine externe Trajektorienplanung nutzbar ist. In einem Autobahnszenario wird dadurch der Aktionsraum für den Algorithmus bspw. reduziert auf die Aktionen:

1. Im Fahrstreifen bleiben und die Geschwindigkeit an das vorne fahrende Fahrzeug anpassen
2. Fahrstreifenwechsel nach links ausführen
3. Fahrstreifenwechsel nach rechts ausführen

Die Daten, die in das NN eingespeist werden, sind bei diesem Ansatz flexibel, wodurch möglich ist, beliebig komplexe Szenarien zu fahren.

7.2.2. Fazit

Bei suchbasierten Ansätzen zeigt sich, dass die Einbeziehung von anderen Verkehrsteilnehmern zwar bei einigen Ansätzen zu guten Ergebnissen führt. Einige der Autoren betonen aber, dass die Komplexität der Berechnung stark ansteigt, wenn die Zahl der Teilnehmer ansteigt. Wenn für jedes Fahrzeug eine bestimmte Anzahl verschiedener Trajektorien generiert wird und zu jedem Fahrzeug zu jedem Zeitschritt auf eine mögliche Kollision geprüft wird, skaliert der Rechenaufwand entsprechend zu jedem der Faktoren linear. Im Falle einer kooperativen Berechnung wie bei Liniger[184a], bei dem alle Fahrzeuge zu einer gemeinsamen Strategie kommen, steigt der Aufwand sogar überproportional mit der Anzahl der Teilnehmer.

Zu diesem Schluss kommt auch Alrifaee[185] und modifiziert daher seinen Ansatz, sodass einzelnen Fahrzeugen eine Priorität gegenüber anderen Fahrzeugen eingeräumt wird. Dadurch ist möglich, dass ein VT vor der Berechnung seiner eigenen Trajektorie nicht auf die Trajektorie der anderen VT wartet. Durch eine Parallelisierung von mehreren VT ist möglich, die notwendige Rechenleistung weiter zu senken. Im Straßenverkehr ist bspw. möglich, diese Priorisierung anhand der Verkehrsregeln zu begründen.[186] Die beiden Ansätze von Gerdts *et al*. und Alrifaee sind jeweils verteilte MPC. Das bedeutet, dass die Trajektorien für alle Fahrzeuge zentral und gemeinsam berechnet werden. Für ein Rennszenario bedeutet das auch für den Fall eines kompetitiven Szenarios, dass für jedes Fahrzeug bekannt ist, welchen Fahrzeugen welche Priorität zugeordnet ist.

Veröffentlichungen, die Szenarien im Bereich des Straßenverkehrs behandeln, umgehen das Problem der hohen Komplexität häufig, indem sie die Fahrt durch den zweidimensionalen Raum auf die longitudinale Bewegung entlang eines vorgegebenen

[184]Liniger, A. et al.: A Non-Cooperative Game Approach to Autonomous Racing (2017).

[185]Alrifaee, B.: Vernetzte modellbasierte prädiktive Regelung zur Kollisionsvermeidung (2017).

[186]Vgl. Gerdts, M. et al.: Motion Planning in Driving Scenarios with Communicating Vehicles (2018).

Pfades beschränken.[187] Im Falle eines Rennszenarios ist diese Lösung des Problems allerdings ausgeschlossen, da im Motorsport die Linienwahl zu einer der wichtigsten Faktoren zählt, die zum Sieg in einem Rennen beitragen.[188]

Die Modellierung spielt bei der Berücksichtigung von anderen VT ebenfalls eine zentrale Rolle. Für den Fall, dass ein oder mehrere Verkehrsteilnehmer auf einer Rennstrecke zu berücksichtigen sind, gibt es mehrere Möglichkeiten. Die vermeintlich einfachste Möglichkeit ist, den Raum, in dem sich ein anderes Fahrzeug befindet, durch einen Kreis oder Ellipse als verfügbaren Raum auszuschließen.[189] In einer einfacheren Version dieser Methode ist auch eine einzelne lineare Randbedingung nutzbar, um den Abstand zwischen zwei Objekten so groß zu halten, dass es zu keiner Kollision zwischen den Objekten kommt.[190] Als zweite Variante ist möglich, über eine Baumsuche aus mehreren generierten Trajektorien die beste auszuwählen.[191] Die Bewertungskriterien, nach denen die Trajektorie ausgewählt wird, sind für einen solchen Ansatz von ebenso großer Bedeutung wie die Reward-Funktion für RL-Algorithmen.[184b] Eine dritte Möglichkeit ist, die Strecke in Längs- und Querrichtung in ein Netz aus Knoten einzuteilen. In einem ersten Schritt wird dann die beste Kombination aus Knoten gesucht. Durch die Position der anderen VT ist möglich, Knoten auszuschließen. Der so gewonnene Pfad wird im Anschluss für die Trajektorienberechnung genutzt.[192] Zuletzt ist möglich, einen Korridor auf der Strecke im Voraus zu bestimmen, der dem Fahrzeug als Platz zur Verfügung gestellt wird.[193]

Die Lösungsansätze, die nicht direkt eine Lösung für die anderen Fahrzeuge berechnen, haben den Nachteil, dass eine Prädiktion nur mit vereinfachten Annahmen durchgeführt wird, was für ein Rennszenario nur bedingt geeignet ist, durch die hochdynamische Fahrweise der Fahrzeuge.[194] Einfache Annahmen wie konstante Geschwindigkeit, Lenkwinkel oder Beschleunigung besitzen nur über kurze Zeiträume ihre Gültigkeit. Das gilt insbesondere für die dynamische Kurvenfahrt, dort sind Annahmen über statische Bewegungsgrößen von besonders geringem Wert. Das birgt ein großes Problem, da je nach den Fahreigenschaften des Fahrzeugs das Überholen in Kurven oft eine der wenigen Gelegenheiten zum Überholen anderer Fahrzeuge bietet. Damit verbunden ist eine Planung, die eine Prädiktion eines anderen Verkehrsteilnehmers umfasst, nur für

[187] Vgl. Hubmann, C. et al.: Planning With Interaction and Uncertain Prediction (2018); vgl. Gerdts, M. et al.: Motion Planning in Driving Scenarios with Communicating Vehicles (2018).

[188] Vgl. Doubek, F. et al.: What makes a good driver on public roads and race tracks? (2020).

[189] Vgl. Gerdts, M. et al.: Motion Planning in Driving Scenarios with Communicating Vehicles (2018).

[190] Vgl. Kloock, M. et al.: Networked Model Predictive Vehicle Race Control (2019).

[191] Vgl. Liniger, A. et al.: Optimization-Based Autonomous Racing of 1:43 Scale RC Cars (2015).

[192] Vgl. Stahl, T. et al.: Multilayer Graph-Based Trajectory Planning for Race Vehicles (2019).

[193] Vgl. Liniger, A. et al.: Optimization-Based Autonomous Racing of 1:43 Scale RC Cars (2015).

[194] Vgl. Stahl, T. et al.: Multilayer Graph-Based Trajectory Planning for Race Vehicles (2019).

einen kurzen Zeithorizont wertvoll, da die Ungenauigkeit der Prädiktion mit einem weiter in die Zukunft reichenden Zeithorizont abnimmt.

Um Kollisionen zu vermeiden, werden andere VT in der Trajektorienberechnung berücksichtigt und dafür gilt in allen Ansätzen, dass die Lösung des Optimierungsproblems keine Kollision zulassen darf. Hierin alleine liegt aber bereits ein Problem, dass speziell im Kontext des Motorsports auftritt. In der Studie von Liniger *et al.*[184c] zeigt sich für den Fall, dass ein anderes Fahrzeug versucht, eine Kollision zu verhindern, die effektivste Strategie für das Verhindern eines Überholmanövers ist, das andere Fahrzeug zu ignorieren und so schnell wie möglich zu fahren. Auf diese Weise wird der notwendige Raum für den Überholer auf einer Rennstrecke versperrt. Der Grund, warum bei menschlichen Fahrern diese Strategie nicht erfolgreich funktioniert, ist, dass ein menschlicher Fahrer auf die Anpassungsfähigkeit der anderen Fahrzeuge vertraut. Er missachtet demnach die Randbedingungen zur Kollisionsvermeidung unter der Annahme, dass das andere Fahrzeug in der Lage ist, seinen ursprünglichen Pfad ebenfalls neu zu planen.

Bei der Nutzung von neuronalen Netzen zur Berücksichtigung von anderen Fahrzeugen hängt der Erfolg des Trainings stark von der verwendeten Referenz bzw. dem verwendeten Szenario ab.[195] Damit ein NN in der Lage ist, gute Ergebnisse in einem gegebenen Szenario zu erzielen, sind folgende Bedingungen entscheidend. Das Szenario erzeugt Beobachtungen, die so ähnlich wie möglich denen sind, mit denen das Netzwerk trainiert wurde. Daher ist eine möglichst einfach Umfeldrepräsentation anzustreben. Das Netz wird im Idealfall mit so vielfältigen Szenarien trainiert, dass die Daten, auf denen es im Anwendungsszenario rechnet, lediglich wie eine Variation während des Trainings wahrgenommen werden.[196]

Im Gegensatz zu einer steigenden Komplexität der Modellierung bei MPC und suchbasierten Ansätzen ist die benötigte Rechenleistung bei vielen Input-Formen für NN für unbegrenzt viele andere Fahrzeuge als konstant zu betrachten.[197] Allerdings ist aus diesem Grund der Zustandsraum von NN häufig bedeutend komplexer als für einen MPC.

7.3. Identifikation von Lücken im Stand der Forschung

Der momentane Stand der Forschung offenbart, dass für die Trajektorienberechnung zwei wesentliche Punkte bislang noch die Entwicklung eines Verhaltensalgorithmus verhindern, der in der Lage ist, auf kompetitive Weise mit anderen Fahrzeugen zusammen auf einer Rennstrecke gegeneinander zu fahren.

[195]Vgl. He, H. et al.: Opponent Modeling in Deep Reinforcement Learning (2016).

[196]Vgl. Tobin, J. et al.: Domain Randomization for Transferring Deep NN to the Real World (2017).

[197]Vgl. Garlick, S. et al.: Real-Time Trajectory Planning Using Machine Learning (2021).

Der erste Punkt, der in heute üblichen Methoden für die Trajektorienberechnung integriert ist, ist die Kollisionsvermeidung. Diese ist notwendig, um dafür zu sorgen, dass es zu keiner Kollision zwischen VT kommt. Sie sorgt allerdings auch dafür, dass ein Überholen in einem Rennkontext nahezu unmöglich ist, wenn von folgenden drei Prämissen ausgegangen wird.

1. Alle Fahrzeuge, die sich auf der Rennstrecke befinden, haben dieselben physischen Eigenschaften.
2. Alle Fahrzeuge versuchen Rundenzeiten zu fahren, die so kurz wie möglich sind.
3. Alle Fahrzeuge sind in der Lage, die gesamte Breite der Strecke zu nutzen.

Die Tatsache, dass alle Fahrzeuge versuchen, so schnell wie möglich zu fahren, bedingt bereits, dass sie versuchen, auf der sogenannten Ideallinie zu fahren. Diese nutzt vor allem in Kurven den gesamten lateralen Platz der Strecke, um die Geschwindigkeiten über der gesamten Länge der Strecke zu maximieren. Versucht ein gleich schnelles Fahrzeug zu überholen, ist das nur über ein Verlassen der Ideallinie möglich. Dadurch ist kein schnelleres Fortkommen mehr möglich als das des Fahrzeugs, das sich auf der Ideallinie bewegt. In diesem Fall ist tatsächlich die effektivste Strategie, ein Überholen zu verhindern, so schnell wie möglich auf der Ideallinie zu fahren.

Der zweite damit verwandte Punkt, der bislang bei nahezu allen Trajektorien-Berechnungsalgorithmen zu Problemen führt, ist der Umgang mit vielen verschiedenen dynamischen Objekten. Durch die Prädiktion der anderen VT ist möglich, dass der verfügbare Raum so stark eingeschränkt wird, dass dem Fahrzeug kein Raum mehr für die eigene Bewegung bleibt. Das sog. “freezing robot“ Problem ist bedingt durch viele dynamische Objekte und die Kollisionsvermeidung. Fan *et al.*[198] und Trautman *et al.*[199] versuchten das Problem am Beispiel der Navigation von Robotern in Menschenmengen zu lösen. Es sorgt dafür, dass ein Fahrzeug in einer Menge von anderen Fahrzeugen einfach stehen bleibt.

Der zugrunde liegende Ansatz bei der Kollisionsvermeidung vernachlässigt allerdings, dass jedes Fahrzeug auf der Strecke in der Lage ist, seine Trajektorie neu zu planen und auf die unerwartete Veränderung eines überholenden Fahrzeugs zu reagieren, vorausgesetzt es existiert ausreichend Platz für das Fahrzeug für eine angemessene Reaktion.

Der im Folgenden vorgestellte Ansatz ermöglicht daher eine Berücksichtigung von anderen Fahrzeugen in Abhängigkeit der Priorität auf der Rennstrecke und ermöglicht, die Prädiktion eines anderen Fahrzeugs partiell zu berücksichtigen und auf diese Weise

[198]Fan, T. et al.: Getting Robots Unfrozen and Unlost in Dense Pedestrian Crowds (2018).

[199]Trautman, P. et al.: Unfreezing the robot (2010).

ein Überholen zu ermöglichen. Gleichzeitig erlaubt der Ansatz, dass dem anderen Fahrzeug ausreichend Platz zur Verfügung steht, um seine Trajektorie an ein aggressives Überholmanöver anzupassen.

Das zugrunde liegende Bewegungsmodell ist als Massenpunktmodell ausgeführt, um durch kurze Rechenzeiten und eine hohe Robustheit auch den Einsatz in Kombination mit einem lernenden Algorithmus zu ermöglichen. Andere Fahrzeuge werden durch lineare Randbedingungen modelliert. Auf diese Weise wird ein quadratisches Problem und die damit verbundenen langen Rechenzeiten vermieden. Zur Kompensation des dadurch entstehenden Nachteils der einfachen Modellierung des verfügbaren Raumes wird eine Kostenfunktion eingesetzt, die bestimmt, auf welcher Seite ein anderes Fahrzeug überholt wird.

7.4. Modell und Randbedingungen

7.4.1. Genereller Ansatz

Um mehrere Fahrzeuge auf einer Strecke in einer künstlichen Umgebung fahren zu lassen, wird der Ansatz einer sequenziellen Linearisierung verwendet, wie er auch von Alrifaee [200] beschrieben ist. Der Grundgedanke ist, dass ausgehend von einer berechneten Lösung die Randbedingungen für den nächsten Berechnungsschritt definiert werden. Der Hauptfaktor ist die Position des Fahrzeugs auf der Strecke für jeden Zeitschritt $n \in [0, N]$. Nach der Berechnung einer Lösung wird jedem Zeitschritt n das Streckensegment mit Index j_n zugeordnet, in dem sich die berechnete Position $\boldsymbol{p}_n$ des Fahrzeugs zum jeweiligen Zeitpunkt befindet.

Für das Problem in dieser Arbeit wird kein Simulationsmodell in Echtzeit verwendet. Um die Fahrzeuge zu bewegen, wird deswegen nach jeder Berechnung einer Lösung der Zeitschritt mit $n = 0$ entfernt und der Zustand mit $n = 1$ als Ausgangspunkt für die nachfolgende Berechnung verwendet. Um die Länge der Trajektorie zu erhalten, wird der letzte berechnete Zeitschritt N an das Ende der Trajektorie kopiert. Dieser Vorgang wird fortlaufend wiederholt, oder bis das Szenario beendet ist.

Der gesamte Prozess der Generierung der Lösung für das Optimierungsproblem ist in Algorithmus 2 zusammengefasst.

Der in dieser Arbeit verwendete Optimierungsalgorithmus erlaubt nur die Verwendung von linearen Gleichungen. Das macht ihn zeiteffizient, hat aber erhebliche Auswirkungen darauf, wie das Modell und die Randbedingungen formuliert werden.

[200] Alrifaee, B. et al.: Real-time Trajectory optimization for Autonomous Vehicle Racing (2018).

7.4.2. Bewegungsmodell

Das in dieser Arbeit verwendete Fahrzeugmodell basiert auf einem Punktmassenmodell. Die Einhaltung der Modellgleichung wird durch die Einschränkung der Beschleunigungen in Raumrichtungen ähnlich wie in der Veröffentlichung von Alrifaee[201] sichergestellt. Das Modell wird für jeden Zeitschritt mit der Dauer $\eth t$ basierend auf der zuvor berechneten Lösung linearisiert. Das System wird durch den Zustand $\boldsymbol{x}_n$ beschrieben. Dieser besteht aus der Position p, der Geschwindigkeit v und der Beschleunigung a. Der Input für das Modell $\boldsymbol{u}_n$ besteht aus dem Ruck $\mathcal{J}$. Die Variablen ${}_E\square$ sind in einem erdfesten Koordinatensystem beschrieben.

Algorithmus 2 Allgemeiner Berechnungsprozess für ein Szenario

1: Erraten einer initialen Lösung (bspw. Stillstand)
2: **while** Szenarion nicht beendet **do**
3: **for** Jedes Fahrzeug auf der Strecke **do**
4: Räumliche Randbedingungen in Bezug auf die zuvor berechnete Lösung festlegen (7.4.5)
5: Beschleunigungs-Randbedingungen in Bezug auf die eigene zuvor berechnete Lösung (7.4.3) setzen. Dabei wird auch die Position des Gegners berücksichtigt (7.4.4)
6: Hinzufügen von räumlichen Beschränkungen zur Berücksichtigung von Fahrzeugen in der Nähe
7: Berechnen der Lösung
8: Diskretisieren der Lösung durch Berechnen von $j_n = \text{nearestSegment}({}_E Pp_n)$
9: Entfernen des ersten Schritts $n = 0$, um das Fahrzeug vorwärts zu bewegen
10: Duplizieren des letzten Zeitschritts $n = N$, um die Trajektorie zu komplettieren
11: **end for**
12: **end while**

Die Zustandsraumgleichung wird durch (7-3) beschrieben. Die Randbedingungen des Systems werden durch k_n Ungleichungen in (7-4) beschrieben.[202] $\boldsymbol{C}_n$ besteht dabei aus k_n Zeilen mit jeweils sechs Zahlen. Jede dieser Zeilen beschreibt eine Randbedingung.

[201] Alrifaee, B. et al.: Real-time Trajectory optimization for Autonomous Vehicle Racing (2018).

[202] Es ist zu beachten, dass aufgrund der zusätzlichen räumlichen Randbedingungen, die in 7.4.5 eingeführt werden, die Anzahl der Randbedingungen spezifisch für den Iterationsschritt n ist und von der berechneten Lösung eines nahe gelegenen Fahrzeugs abhängt.

$$\boldsymbol{x}_n = \left[{}_E p_{x,n},\, {}_E p_{y,n},\, {}_E v_{x,n},\, {}_E v_{y,n},\, {}_E a_{x,n},\, {}_E a_{y,n}\right]^T \tag{7-1}$$

$$\boldsymbol{u}_n = \left[{}_E \jmath_{x,n},\, {}_E \jmath_{y,n}\right]^T \tag{7-2}$$

$$\boldsymbol{x}_{n+1} = \boldsymbol{A}_n \boldsymbol{x}_n + \boldsymbol{B}_n \boldsymbol{u}_n + \boldsymbol{b}_n \tag{7-3}$$

$$\boldsymbol{x}_{l,n} \leq \boldsymbol{C}_n \boldsymbol{x}_n \leq \boldsymbol{x}_{u,n} \tag{7-4}$$

$$\boldsymbol{C}_n = \left[\boldsymbol{c}_{n,0}, \boldsymbol{c}_{n,1}, ..., \boldsymbol{c}_{n,k_n}\right]^T \tag{7-5}$$

Das zweidimensionale Punktmassenmodell wird durch die folgenden Matrizen beschrieben:

$$\boldsymbol{A}_n = \begin{bmatrix} 1 & 0 & \eth t & 0 & \frac{1}{2}\eth t^2 & 0 \\ 0 & 1 & 0 & \eth t & 0 & \frac{1}{2}\eth t^2 \\ 0 & 0 & 1 & 0 & \eth t & 0 \\ 0 & 0 & 0 & 1 & 0 & \eth t \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \boldsymbol{B}_n = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \eth t & 0 \\ 0 & \eth t \end{bmatrix} \tag{7-6}$$

Um den momentanen Ausgangszustand des Fahrzeugs zu definieren, wird der Offsetvektor $\boldsymbol{b}_n$ verwendet. Anstelle der Beschleunigung wird der Ruck $\jmath$ als Eingabewert genutzt. Auf diese Weise ist eine Begrenzung des Rucks möglich und die Formulierung der in Anhang A beschriebenen Kostenfunktion wird erleichtert.

7.4.3. Randbedingungen der Beschleunigung

Die Bewegung des Modells wird nur durch die Beschleunigung in den beiden Raumrichtungen gemäß der Reibungsellipse sowie den Luftwiderstand begrenzt. Die Begrenzungen des Modells ergeben sich also aus den grundlegenden Fahrwiderständen und der Kinematik und werden durch die folgenden vier Parameter beschrieben. Die maximale Verzögerung beim Bremsen D_{max}, die maximale Beschleunigung in Querrichtung $a_{\text{lat,max}}$ und der Reibungskoeffizient μ werden als Konstanten gesetzt. Die Längsbeschleunigung wird begrenzt durch (a) das Produkt μg aus dem Reibungskoeffizienten μ zwischen Reifen und Fahrbahn und der Erdbeschleunigung g, (b) eine konstante maximale Vortriebsleistung P_{max} und (c) den Luftwiderstand. Damit ergibt sich folgende Definition der longitudinalen Beschleunigung

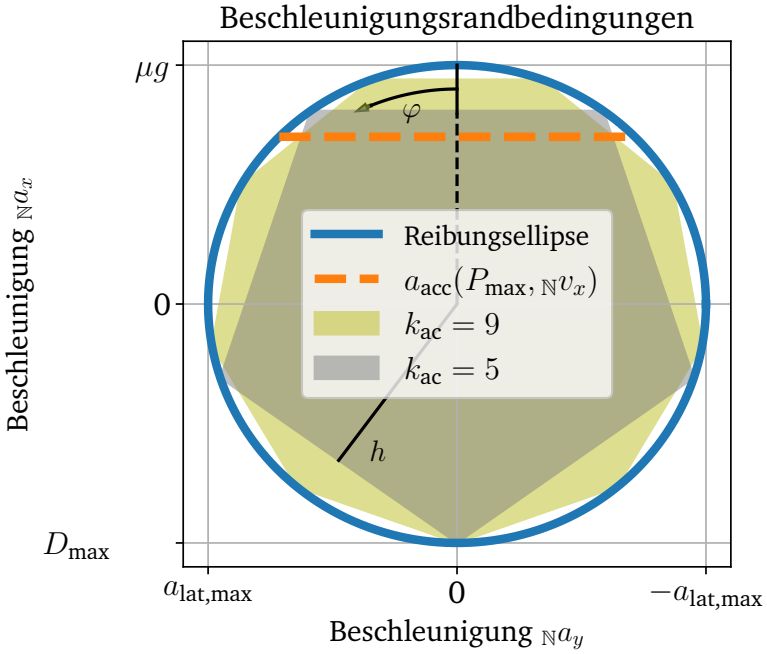


Abbildung 7-2.: Visualisierung der Beschleunigungsrandbedingungen für $k_{ac} = 5$ und $k_{ac} = 9$. Die orangefarbene Linie stellt die geschwindigkeitsabhängige Begrenzung der Längsbeschleunigung dar, die durch den Luftwiderstand und die Antriebsleistung begrenzt wird.

$$a_{\text{acc}}(P_{\text{max}}, {}_{\mathbb{N}}v_x) = \min\left(\begin{bmatrix} \mu g \\ \frac{P_{\text{max}}}{{}_{\mathbb{N}}v_x \cdot m} - \frac{c_w \cdot A \cdot \rho \cdot {}_{\mathbb{N}}v_x{}^2}{2 \cdot m} \end{bmatrix}\right) \tag{7-7}$$

mit der Fahrzeugmasse m, der projizierten Fläche A, der Fahrzeuglängsgeschwindigkeit ${}_{\mathbb{N}}v_x$ in natürlichen Fahrzeugkoordinaten, dem Luftwiderstandsbeiwert c_w und der Luftdichte ρ.

Um sicherzustellen, dass diese Modellgrenzen eingehalten werden, werden $k_{ac} = 9$ lineare Randbedingungen für die Beschleunigung (engl. acceleration constraint) formuliert, die die Reibungsellipse beschreiben. Die Reibungsellipse wird durch μg, $a_{\text{lat,max}}$ und D_{max} charakterisiert. Zusätzlich wird zur Begrenzung der maximalen Längsbeschleunigung, die mit zunehmender Geschwindigkeit abnimmt, eine weitere Randbedingung formuliert. Abbildung 7-2 zeigt diese Randbedingungen für die Quer- und Längsbeschleunigung für die theoretische Reibungsellipse. Dargestellt sind zwei Sätze linearisierter Randbedingungen für $k_{ac} = 5$ und $k_{ac} = 9$. Je mehr Randbedingungen verwendet werden, desto näher kommt der Planer an die Grenzellipse heran. In x-Richtung zeigt die orange Linie zusätzlich die Begrenzung der Antriebsleistung bei schneller Fahrt.

Die Randbedingungen zur Beschreibung der Reibungsellipse werden für die Laufvaria-

ble $i \in [1, k_{ac}]$ wie folgt formuliert:

$$\boldsymbol{c}_{n,\text{acc},i} = \begin{bmatrix} 0 & 0 & 0 & 0 & {}_E a_{x,n}(i) & {}_E a_{y,n}(i) \end{bmatrix} \tag{7-8}$$

$$\begin{bmatrix} {}_E a_{x,n}(i) \\ {}_E a_{y,n}(i) \end{bmatrix} = \begin{bmatrix} a_{\text{lat,max}} \cos\left(i \frac{2\pi}{k_{ac}}\right) \\ {}_{\mathbb{N}} a_x(i) \sin\left(i \frac{2\pi}{k_{ac}}\right) \end{bmatrix}^{\top} \cdot \boldsymbol{M} \tag{7-9}$$

$$\boldsymbol{M} = \begin{bmatrix} \cos(\psi_{\text{c},n}) & -\sin(\psi_{\text{c},n}) \\ \sin(\psi_{\text{c},n}) & \cos(\psi_{\text{c},n}) \end{bmatrix} \tag{7-10}$$

$$\psi_{\text{c},n} = \text{atan2}({}_E v_{x,n}, {}_E v_{y,n}) \tag{7-11}$$

$${}_{\mathbb{N}} a_x(i) = \begin{cases} D_{\text{max}}, & i \in \left[\frac{1}{4} k_{\text{ac}}, \frac{3}{4} k_{\text{ac}}\right] \\ \mu g & \text{andernfalls} \end{cases} \tag{7-12}$$

$$x_{\text{u},n,\text{acc},i} = {}_{\mathbb{N}} a_x(i) \cdot a_{\text{lat,max}} \cdot h \tag{7-13}$$

$$x_{\text{l},n,\text{acc},i} = -\infty \tag{7-14}$$

$$h = \cos(\pi / k_{\text{ac}}) \tag{7-15}$$

Eine Randbedingung wird durch den Vektor $\boldsymbol{c}_{n,\text{acc},i}$ und die obere Grenze $x_{u,n,\text{acc},i}$ beschrieben. $\boldsymbol{c}_{n,\text{acc},i}$ stellt einen Vektor vom Mittelpunkt in Richtung der Außengrenze der Reibungsellipse dar. Mit dem Kurswinkel ψ_{c}, der mit der atan2-Funktion[203] berechnet wird, wird der Vektor vom natürlichen Fahrzeugkoordinatensystem in das erdfeste System mithilfe der Drehmatrix $\boldsymbol{M}$ transformiert. In diesem Schritt werden die Randbedingungen des aktuellen Schrittes mit der Lösung des vorherigen Schrittes verbunden. Die entscheidenden Komponenten der Randbedingung sind die Längskomponente ${}_{\mathbb{N}} a_x(i)$ und die Querkomponente $a_{\text{lat,max}}$. Die Längskomponente unterscheidet sich beim Beschleunigen und beim Bremsen. Für den Wert der oberen Schranke $x_{u,n,\text{acc},i}$ werden nicht die beiden Radien der Ellipse verwendet, sondern die Apotheme[204] h. Dadurch wird sichergestellt, dass die Beschleunigungswerte innerhalb der Ellipse bleiben, auch wenn nur wenige Randbedingungen verwendet werden.

Um die abnehmende Längsbeschleunigung mit zunehmender Geschwindigkeit zu berücksichtigen, wird die folgende Randbedingung hinzugefügt.

[203] Die atan2-Funktion ist eine verbesserte Version der Arcustangens-Funktion. Sie teilt sich in sechs separate Fälle auf, um das Problem zu überwinden, dass der Arcustangens nur zwischen $-\frac{\pi}{2}$ und $\frac{\pi}{2}$ gültig ist.

[204] Die Apotheme ist die minimale Länge vom Mittelpunkt eines regelmäßigen Polygons zu seiner Kante.

$$\boldsymbol{c}_{n,P} = \begin{bmatrix} 0 & 0 & 0 & 0 & \cos(\psi_{\mathrm{c},n}) & \sin(\psi_{\mathrm{c},n}) \end{bmatrix} \tag{7-16}$$

$$x_{\mathrm{u},n,P} = \min(\mu g h, a_{\mathrm{acc}}(P_{\mathrm{max}}, {}_{\mathbb{N}}v_x)) \tag{7-17}$$

$$x_{\mathrm{l},n,P} = -\infty \tag{7-18}$$

Die Längsbeschleunigung wird also entweder durch den Reibungskoeffizienten μ oder die Vortriebsleistung des Fahrzeugs begrenzt. Es ist möglich, diese Beschränkungen auch weiter zu modifizieren, um aerodynamische Effekte zu berücksichtigen, wie im folgenden Abschnitt beschrieben.

Der Hinweis auf Parametervariationen im nachfolgenden Verlauf dieser Arbeit bezieht sich stets auf die drei eben genannten Parameter für die Vortriebsleistung P_{max}, die maximale laterale Beschleunigung $a_{\mathrm{lat,max}}$ und die maximale Bremsverzögerung D_{max}.

7.4.4. Anpassungen zur Berücksichtigung der Aerodynamik

Für die Untersuchungen mit lernenden Fahralgorithmen werden die Beschleunigungsrandbedingungen modifiziert, um aerodynamische Effekte im folgenden heuristischen Ansatz zu berücksichtigen. Das Modell basiert auf der Forschung von Dominy[205] und den folgenden Annahmen. Die Luft, die den Luftwiderstand und den Abtrieb des Fahrzeugs verursacht, wird durch ein durchfahrendes Fahrzeug in Längsrichtung beschleunigt und verwirbelt. Nach einer Zeit von $\tau_{\text{Æ,min}}$ beginnt dieser Effekt zu verblassen und ist nach insgesamt $\tau_{\text{Æ,max}}$ verschwunden. Die Intensität des Effekts wird maßgeblich durch den Anteil der Fahrzeugfront beeinflusst, mit dem das Fahrzeug durch die verwirbelte Luft fährt. Dies wirkt sich nicht nur auf den Luftwiderstand des Fahrzeugs aus, sondern auch auf die mögliche Querbeschleunigung.

Die reduzierten Grenzen der Längsbeschleunigung $a_{\mathrm{acc,red}}$ und der Querbeschleunigung $a_{\mathrm{lat,max,red}}$ sind durch die Grenzwerte ohne Beeinflussung und den jeweiligen maximalen Betrag der Anpassung für die jeweilige Beschleunigung definiert. Die Längsbeschleunigung kann nur um den Beitrag des Luftwiderstands (engl. drag) $a_{\mathrm{dr,red,max}}$ angehoben werden, der von einem verringerten Luftwiderstandsbeiwert $c_{w,\mathrm{red}}$ durch ein vorausfahrendes Fahrzeug erzeugt wird. Die Änderung der Querbeschleunigung wird in Abhängigkeit der maximal reduzierbaren Radlast $F_{z,\mathrm{red,max}}$ limitiert. Diese beiden Werte werden als spezifisch für ein Fahrzeug angenommen.

[205] Dominy, R. G.: The Influence of Slipstreaming on the Performance of a Grand Prix Racing Car (1990).

$$a_{\text{acc,red}} = a_{\text{acc}}(P_{\max}, {}_{\mathbb{N}}v_x) + a_{\text{dr,red,max}} \cdot \lambda_{\text{long}}(\tau_{\text{fo}}, d_y) \tag{7-19}$$

$$a_{\text{dr,red,max}} = \frac{c_{w,\text{red}} \cdot A \cdot \rho \cdot {}_{\mathbb{N}}v_x^2}{2 \cdot m} \tag{7-20}$$

$$a_{\text{lat,max,red}} = a_{\text{lat,max}} \cdot \left(1 - \lambda_{\text{lat}}(\tau_{\text{fo}}, d_y)\frac{F_{z,\text{red,max}}}{F_z}\right) \tag{7-21}$$

Die Beschleunigungen werden in beiden Raumrichtungen jeweils mittels Reduktionsfaktoren limitiert. Die Reduktionsfaktoren für die laterale Beschleunigung $\lambda_{\text{lat}} \in [0, 1]$ und für die longitudinale Beschleunigung $\lambda_{\text{long}} \in [0, 1]$ bestimmen den Betrag der Anpassung für die jeweiligen Beschleunigungen. Beide Faktoren setzen sich jeweils aus einer lateralen und einer longitudinalen Komponente zusammen. Weiterhin gilt die Annahme, dass die Luftwiderstands-/Abtriebs-Reduktion auf der aufgewirbelten Luft beruht und nicht auf der lateralen/longitudinalen Position hinter dem Vordermann zum aktuellen Zeitschritt. Aus diesem Grund wird nicht der räumliche, sondern der zeitliche Abstand τ_{fo} verwendet, in dem das Fahrzeug dem Gegner in Längsrichtung folgt. Er ist definiert durch den Zeitpunkt, an dem die aktuelle Position des Ego-Fahrzeugs ${}_F x_{\text{ego}}$ in Streckenkoordinaten zum Zeitpunkt t_0 gleich der Position des anderen Fahrzeugs in der Vergangenheit ${}_F x_{\text{opp,past}}$ ist. Die laterale Komponente wird durch den lateralen Abstand $|d_y|$ zum anderen Fahrzeug bestimmt. Dieser ist ebenfalls für den Zeitpunkt $t_0 - \tau_{\text{fo}}$ definiert.

$$\lambda_{\text{red,lat}}(d_y) = \max\left(0, \frac{w_{\text{car}} - |d_y|}{w_{\text{car}}}\right) \tag{7-22}$$

$$d_y = {}_F d_y\left({}_E\boldsymbol{p}_{\text{opp}}(t_0 - \tau_{\text{fo}}), {}_E\boldsymbol{p}_{\text{ego}}(t_0)\right) \tag{7-23}$$

$$\lambda_{\text{red,t}}(\tau_{\text{fo}}) = \begin{cases} 1, & \tau_{Æ,\min} > \tau_{\text{fo}} \\ 1 - \dfrac{\tau_{\text{fo}} - \tau_{Æ,\min}}{\tau_{Æ,\max} - \tau_{Æ,\min}}, & \\ \quad \tau_{\text{fo}} \in [\tau_{Æ,\min}, \tau_{Æ,\max}] & \\ 0, & \tau_{\text{fo}} > \tau_{Æ,\max} \end{cases} \tag{7-24}$$

$$\lambda_{\text{long}}(\tau_{\text{fo}}, d_y) = \lambda_{\text{red,lat}}(d_y)\lambda_{\text{red,t}}(\tau_{\text{fo}}) \tag{7-25}$$

$$\lambda_{\text{lat}}(\tau_{\text{fo}}, d_y) = \lambda_{\text{red,lat}}(d_y)^2\lambda_{\text{red,t}}(\tau_{\text{fo}}) \tag{7-26}$$

$$\tau_{\text{fo}} = t({}_F x_{\text{opp,past}} = {}_F x_{\text{ego}}) - t_0 \tag{7-27}$$

Um den Erkenntnissen von Dominy[206] Rechnung zu tragen, wird der Einfluss der lateralen Abweichung auf die Querbeschleunigung quadratisch gewählt. Die λ-Werte

[206]Dominy, R. G.: The Influence of Slipstreaming on the Performance of a Grand Prix Racing Car (1990).

für verschiedene τ_{fo} und d_y sind in Abbildung 7-3 visualisiert. Die Werte in der oberen Hälfte zeigen die Verringerung des Luftwiderstands, während die Werte in der unteren Hälfte die Verringerung der Querbeschleunigung zeigen.

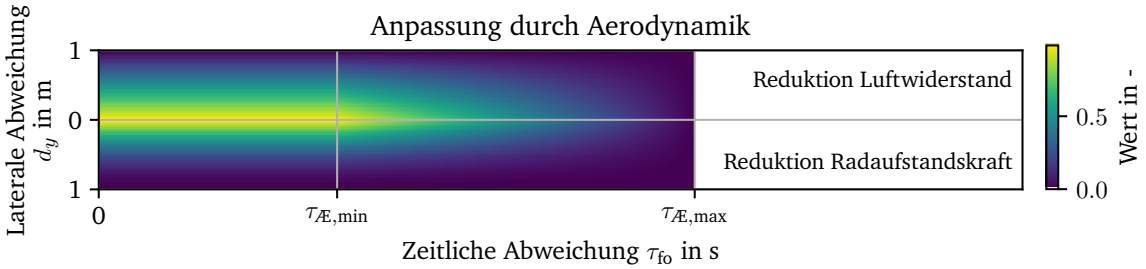


Abbildung 7-3.: Visualisierung der aerodynamischen Reduktionsfunktion λ_{long} in der oberen Hälfte und λ_{lat} in der unteren Hälfte über der seitlichen Abweichung d_y und dem Zeitabstand τ_{fo}.

7.4.5. Randbedingungen zur Berücksichtigung anderer Fahrzeuge

Zur Formulierung der räumlichen Randbedingungen wird die folgende verallgemeinerte Form genutzt.

$$\boldsymbol{c}_{\text{spatial}} = \begin{bmatrix} e_x & e_y & 0 & 0 & 0 & 0 \end{bmatrix} \tag{7-28}$$

$$\boldsymbol{e}_{\text{spatial}} = \begin{bmatrix} e_x & e_y \end{bmatrix} \tag{7-29}$$

$$1 = \sqrt{e_x^2 + e_y^2} \tag{7-30}$$

$${}_E\boldsymbol{p}_{\text{u}} = \begin{bmatrix} {}_Ep_{x,\text{u}} \\ {}_Ep_{y,\text{u}} \end{bmatrix}, \quad {}_E\boldsymbol{p}_{\text{l}} = \begin{bmatrix} {}_Ep_{x,\text{l}} \\ {}_Ep_{y,\text{l}} \end{bmatrix} \tag{7-31}$$

$$x_{\text{u,spatial}} = \boldsymbol{e}_{\text{spatial}} \bullet {}_E\boldsymbol{p}_{\text{u}} \tag{7-32}$$

$$x_{\text{l,spatial}} = \boldsymbol{e}_{\text{spatial}} \bullet {}_E\boldsymbol{p}_{\text{l}} \tag{7-33}$$

Die Randbedingung $\boldsymbol{c}_{n,\text{spatial}}$ besteht nur aus den normierten Komponenten e_x und e_y für beide Richtungen im Raum, die auch den Richtungsvektor $\boldsymbol{e}_{\text{spatial}}$ definieren. $\boldsymbol{e}_{\text{spatial}}$ zusammen mit zwei Punkten im Raum (${}_E\boldsymbol{p}_{\text{u}}$ und ${}_E\boldsymbol{p}_{\text{l}}$) bilden die Grundlage für die Randbedingung, um die obere $x_{\text{u},n,\text{spatial}}$ und untere Schranke $x_{\text{l},n,\text{spatial}}$ zu berechnen.

Die Strecke selbst ist in Segmente gleicher Länge von $\Delta s = 1\,\text{m}$ in ${}_Fx$ bezüglich der Mittellinie diskretisiert, jeweils mit dem entsprechenden Index j. Die Menge aller Streckepunkte $\mathcal{T}$ definiert sich damit zu

$$\mathcal{T} = ({}_E\boldsymbol{p}_j) \text{ mit } j \in [0, J],$$

wobei J die Anzahl an Streckensegmente für eine spezifische Strecke beschreibt.

Der verfügbare Raum für das Modell wird mit einer linearen Randbedingung für jeden Zeitschritt n eingegrenzt. Die linke Streckenbegrenzung wird durch die obere Schranke und die rechte Streckenbegrenzung durch die untere Schranke dargestellt. Die Randbedingungen hängen von der vorherigen Lösung für jeden Zeitschritt n ab, für den prädiziert wird, dass sich das Fahrzeug im Segment

$$j := j_n = \text{nearestSegment}({}_E\boldsymbol{p}_n)$$

der Strecke befindet.

Die Funktion nearestSegment dient dazu, nach der Berechnung der Trajektorie einen berechneten Punkt ${}_E\boldsymbol{p}_n$ einem Streckensegment j_n zuzuordnen. Dafür sucht die Funktion innerhalb der Menge der Streckenpunkte $\mathcal{T}$ den Punkt mit der geringsten Distanz zum Punkt der Trajektorie.

$$\text{nearestSegment}({}_E\boldsymbol{p}_n) = \arg\min_j \left(\|{}_E\boldsymbol{p}_n - {}_E\boldsymbol{p}_j\|\right) \text{ mit } j \in [0, J] \tag{7-34}$$

Die Randbedingungen für den Zeitschritt n definiert sich demnach zu:

$$\boldsymbol{e}_{\text{spatial}} = \boldsymbol{e}_{\text{y,tr},j} = \begin{bmatrix} -\sin(\psi_{\text{tr},j}) & \cos(\psi_{\text{tr},j}) \end{bmatrix} \tag{7-35}$$

$${}_E\boldsymbol{p}_{\text{u}} = {}_E\boldsymbol{p}_{\text{LB},j} = \begin{bmatrix} {}_E x_{j,\text{LB}} \\ {}_E y_{j,\text{LB}} \end{bmatrix} \tag{7-36}$$

$${}_E\boldsymbol{p}_{\text{l}} = {}_E\boldsymbol{p}_{\text{RB},j} = \begin{bmatrix} {}_E x_{j,\text{RB}} \\ {}_E y_{j,\text{RB}} \end{bmatrix} \tag{7-37}$$

Dabei ist $\boldsymbol{e}_{\text{y,tr},j}$ ein Vektor, der senkrecht zum Streckenabschnitt j (engl. track) mit dem Orientierungswinkel $\psi_{\text{tr},j}$ zeigt, in dem sich der Prädiktion zufolge das Fahrzeug zum Zeitschritt n befindet. Die linke Streckenbegrenzung ${}_E\boldsymbol{p}_{\text{LB},j}$ dient dabei als Grundlage für die obere Schranke und die rechte Streckenbegrenzung ${}_E\boldsymbol{p}_{\text{RB},j}$ als Grundlage für die untere Schranke.

Um ein anderes Fahrzeug auf der Strecke zu berücksichtigen, werden zusätzliche räumliche Randbedingungen hinzugefügt, um den Raum auszuschließen, in dem das andere Fahrzeug voraussichtlich fahren wird. Die Randbedingungen werden zusätzlich für einen bestimmten Zeitschritt n hinzugefügt. Diese Methode ist daher in der, Lage

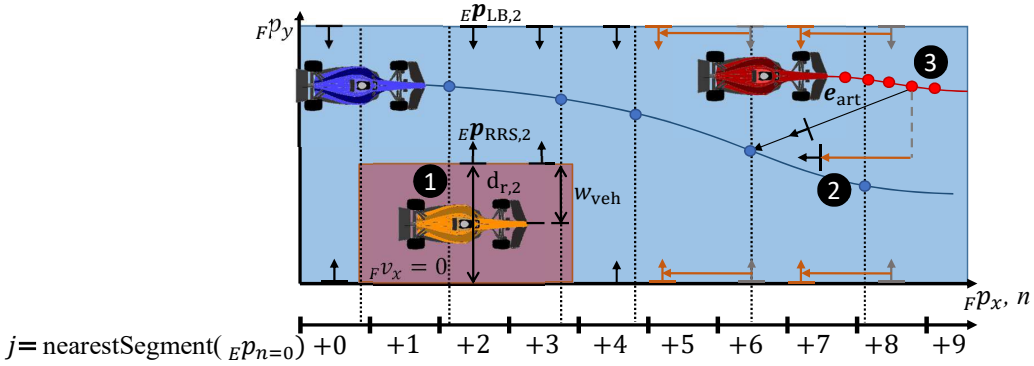


Abbildung 7-4.: Visualisierung der drei möglichen Randbedingungen. Die gestrichelten Linien stellen die berechnete Position ${}_F p_{x,n}$ zu den Zeitschritten n dar. Das blaue Fahrzeug ist das kontrollierte Fahrzeug, für das die Lösung (blaue Punkte) berechnet wird. Das orange Fahrzeug ist ein sich nicht bewegender Gegner. Das rote Fahrzeug ist ein sich langsam bewegender Gegner, für den die roten Punkte als vorausberechnete Lösung gelten.

mehrere Fahrzeuge gleichzeitig zu berücksichtigen. Um dies zu erreichen, wird in diesem Ansatz eine Trajektorie für das andere Fahrzeug mit dem identischen Algorithmus berechnet. Es wird also angenommen, dass die Trajektorie des anderen Fahrzeugs vollständig bekannt ist. Die im Voraus berechnete Trajektorie wird jedoch nur unter der Annahme verwendet, dass die Genauigkeit der Prädiktion mit der Distanz zum Zeitpunkt $n = 0$ abnimmt.

In dieser Arbeit werden drei Haupttypen von Randbedingungen verwendet, um den Raum für das Fahrzeug innerhalb des Optimierungsproblems einzuschränken. Sie sind alle in Abb. 7-4 visualisiert.

Um nur den seitlichen Raum (❶ in Abbildung 7-4) nach Gleichungen (7-35)-(7-37) für das Problem zu beschränken, wird lediglich der Bezugspunkt ${}_E \boldsymbol{p}_{\mathrm{RB/LB},j}$ um den Abstand $d_{\mathrm{l},j}$ von der linken oder $d_{\mathrm{r},j}$ rechten Seite zu den Punkten des beschränkten Raumes (engl. restricted space) ${}_E \boldsymbol{p}_{\mathrm{RSL|RSR},n}$ verschoben. Da dieser Ansatz nur ein Punktmassenmodell verwendet, ist mindestens eine Fahrzeugbreite w_{veh} Spielraum notwendig, um die räumliche Ausdehnung der realen Fahrzeuge zu berücksichtigen.

$$ {}_E \boldsymbol{p}_{\mathrm{RSL|RSR},n} = {}_E \boldsymbol{p}_{\mathrm{LB|RB},j} + d_{\mathrm{l|r},n} \cdot \boldsymbol{e}_{y,\mathrm{tr},j} \tag{7-38} $$

Um den Längsfortschritt des Fahrzeugs zu reduzieren ❷, wird eine einseitige räumliche Randbedingung mit dem Richtungsvektor $\boldsymbol{e}_{\mathrm{spatial}} = \boldsymbol{e}_{\mathrm{x,tr},j}$ gesetzt, die nicht senkrecht, sondern parallel zur Strecke zeigt. Dabei ist ein Abstand von mindestens einer Wagenlänge l_{veh} notwendig, um die reale räumliche Ausdehnung der Fahrzeuge zu berücksichtigen. Der relevante Punkt für die obere Schranke ist in diesem Fall der

Punkt der Mittellinie des Streckenabschnitts, der l_{veh} hinter dem Gegner-Fahrzeug liegt.

$$ {}_E\boldsymbol{p}_{\text{u}} = {}_E\boldsymbol{p}_{\text{tr}}(j = j({}_E\boldsymbol{p}_{\text{opp}}) - \frac{l_{\text{veh}}}{\Delta s}) \tag{7-39} $$

Zusätzlich ist notwendig, aufgrund des geminderten Fortschritts alle nachfolgenden räumlichen Randbedingungen anzupassen, da das Fahrzeug in den nachfolgenden Zeitschritten nicht so viel Strecke zurücklegen darf wie vorhergesagt (orangefarbene Pfeile in Abbildung 7-4). Dies geschieht, indem der vorhergesagte Fortschritt des Fahrzeugs um den Betrag reduziert wird, um den das Fahrzeug in Längsrichtung eingeschränkt wird. Um die Lösbarkeit des Problems zu verbessern, darf jeweils nur eine Längsbeschränkung gesetzt werden.

Eine dritte Möglichkeit besteht darin, eine vollständig synthetische Randbedingung ❸ zu setzen, wie von Kloock *et al.*[207] vorgestellt. Hier ist der Richtungsvektor $\boldsymbol{e}_{\text{art}}$ (engl. artificial) die Differenz der Positionen des Ego-Fahrzeugs und des Gegner-Fahrzeugs, normiert auf den Abstand zwischen den Fahrzeugen. Die obere Schranke wird dann mit einem Referenzpunkt berechnet, der mindestens die Länge der diagonalen Distanz der Bounding-Box des Fahrzeugs $\sqrt{w_{\text{veh}}^2 + l_{\text{veh}}^2}$ von der vorhergesagten Position des Gegners entfernt ist. Die volle Diagonallänge wird verwendet, um die räumliche Ausdehnung beider Fahrzeuge zu berücksichtigen.

$$ \boldsymbol{e}_{\text{spatial}} = \boldsymbol{e}_{\text{art}} = \frac{{}_E\boldsymbol{p}_{\text{opp}} - {}_E\boldsymbol{p}_{\text{ego}}}{\left\| {}_E\boldsymbol{p}_{\text{opp}} - {}_E\boldsymbol{p}_{\text{ego}} \right\|} \tag{7-40} $$

$$ {}_E\boldsymbol{p}_{\text{u}} = {}_E\boldsymbol{p}_{\text{opp}} - \sqrt{w_{\text{veh}}^2 + l_{\text{veh}}^2} \cdot \boldsymbol{e}_{\text{art}} \tag{7-41} $$

Diese Art und Weise, Randbedingungen zu formulieren, um ein anderes Fahrzeug auf der Strecke zu berücksichtigen, bildet den Vergleichsmaßstab für diese Arbeit und wird im Folgenden als **Referenz** bezeichnet.

7.5. Zustandsautomat

Nach der Erläuterung der grundsätzlichen Modellstruktur wird im Folgenden der in dieser Arbeit genutzte Zustandsautomat definiert. Dieser beschreibt die Randbedingungen, die die Planung einer Trajektorie in Anwesenheit von anderen Fahrzeugen

[207] Kloock, M. et al.: Networked Model Predictive Vehicle Race Control (2019).

ermöglichen. Der Zustandsautomat beinhaltet zunächst eine Unterscheidung, an welcher Stelle sich ein anderes Fahrzeug in longitudinaler Streckenrichtung in Relation zum Ego-Fahrzeug befindet.

7.5.1. Phasen des Rennverhaltens

Um eine Trajektorie auf einer Rennstrecke zu planen, die nur lineare räumliche Randbedingungen berücksichtigt, wird das Verhalten in vier verschiedene Phasen aufgeteilt. Diese Phasen sind grundsätzlich in vier Gebiete unterteilt, je nachdem, wo der Gegner sich relativ zum Ego-Fahrzeug auf der Strecke befindet. Diese vier Phasen werden in den folgenden Abschnitten mit den Randbedingungen beschrieben, die zur Bewältigung der jeweiligen Situation gesetzt werden.

Wenn keine anderen Fahrzeuge in der Nähe sind, darf das Auto frei fahren und es sind keine zusätzlichen Randbedingungen erforderlich.

Wenn sich ein anderes Fahrzeug vor dem eigenen Fahrzeug befindet, ist das Ziel dieses Fahrzeug zu überholen, ohne eine Kollision zu verursachen. In diesem Fall gibt es drei verschiedene Möglichkeiten. Entweder wird das Auto rechts oder links überholt oder in manchen Situationen ist ein Überholen unmöglich.

Wenn zwei Fahrzeuge nebeneinander fahren, ist das Ziel, einen Zusammenstoß zu vermeiden und dem Gegner mindestens eine Wagenbreite Platz zu lassen.

Wenn ein Gegner hinter dem eigenen Fahrzeug fährt, ist notwendig, dass das Fahrzeug seine eigene Position verteidigt. Daher ist möglich, das Optimierungsproblem so zu modifizieren, dass das Fahrzeug nach links, rechts oder die Mitte der Strecke gezogen wird, um dem Gegner das Überholen zu erschweren. In manchen Fällen ist durch die Regeln aber auch vorgeschrieben, dass das Fahrzeug den Gegner passieren lässt.[208]

In den folgenden Abschnitten wird eine Methode zur Formulierung linearer räumlicher Beschränkungen mit erweiterter Entscheidungsfindung (engl. Linear Constraints with Advanced Decision Making LCADM) für die Phasen des Nebeneinanderfahrens und des Überholens beschrieben. Das Abwehren eines anderen Fahrzeugs wird über den Ansatz in Abschnitt 7.6 mit einem NN untersucht, da die Simulationen das gleiche Ergebnis wie von Liniger[209] ergaben, dass ein regelbasierter Ansatz zum Verhindern des Überholens dem Gegner das Überholen in vielen Fällen erleichtert.

7.5.2. Problemformulierung zum Nebeneinanderfahren

Die Phase des Nebeneinanderfahrens ist definiert, wenn die Fahrzeuge in ${}_F p_x$ so nahe beieinander fahren, dass sie kollidieren würden, wenn sie auf der gleichen lateralen

[208] Vgl. FIA: Appendix H - Supervision of the road and emergency services (2021), S. 18f.

[209] Liniger, A. et al.: A Non-Cooperative Game Approach to Autonomous Racing (2017).

Position ${}_F p_y$ fahren würden. Für den Längsabstand gilt also bei $n = 0$ folgende Bedingung.

$$ {}_F d_{x,0} \left({}_E \boldsymbol{p}_{\text{ego}}, {}_E \boldsymbol{p}_{\text{opp}} \right) \leq l_{\text{veh}} \tag{7-42} $$

In diesem Fall wird die vorhergesagte Lösung des anderen Fahrzeugs ignoriert. Nur der seitliche Abstand vom Rand der Strecke ${}_F d_{y,0} \left({}_E \boldsymbol{p}_{\text{opp}}, {}_E \boldsymbol{p}_{\text{LB|RB}} \right)$ zum Zeitschritt $n = 0$ wird zur Formulierung der seitlichen Randbedingungen verwendet. Die Reduktion des lateralen Raums erfolgt mit der in Gleichung (7-38) beschriebenen Methode. Die seitliche Abweichung des Referenzpunktes für die Randbedingung wird durch die folgenden Gleichungen adaptiert und ist in Abb. 7-5 visualisiert.

$$ d_{\text{l|r},n} = \max \left(\begin{bmatrix} \lambda_{\text{PD}}(n) \cdot {}_F d_{y,0} \left({}_E \boldsymbol{p}_{\text{opp}}, {}_E \boldsymbol{p}_{\text{LB|RB}} \right) \\ w_{\text{veh}} \end{bmatrix} \right) \tag{7-43} $$

$$ \lambda_{\text{PD}}(n) = 1 - \left(\frac{n}{n_{\text{PD}}} \right)^2 \qquad n_{\text{PD}} = \frac{\tau_{\text{PD,max}}}{\mathfrak{d}t} \tag{7-44} $$

$$ n \leq n_{\text{PD}} \tag{7-45} $$

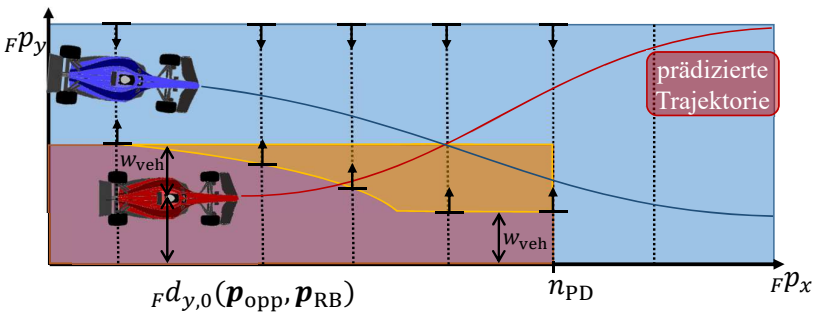


Abbildung 7-5.: Lineare räumliche Randbedingungen für das Fahren neben dem gegnerischen Fahrzeug mit verblassendem Einfluss des Gegners. Blaues Fahrzeug: gesteuertes Fahrzeug, rotes Fahrzeug: Gegner-Fahrzeug. Roter Bereich: nicht zum Fahren nutzbar, beschrieben durch die laterale Position des roten Fahrzeugs, gelber Bereich: erweiterter lateraler Raum durch Skalierungsfaktor λ_{PD}. Die rote und blaue Linie stellen die vorhergesagte Lösung für das entsprechende Fahrzeug dar und gehen nicht in die Bestimmung der Bereiche ein.

Die momentane laterale Distanz des anderen Fahrzeugs mit w_{veh} als Minimalabstand wird für paralleles Fahren (engl. parallel driving) um λ_{PD} skaliert, wobei n_{PD} der Zeitschritt $\tau_{\text{PD,max}}$ Sekunden in der Zukunft ist. Dieser Skalierungsfaktor stellt sicher, dass das Fahrzeug in der Lage ist, einen konkurrenzfähigen Weg um den Gegner herum zu planen, wenn das Ego-Fahrzeug in der Lage ist schneller zu fahren als das Gegner-Fahrzeug. Die Funktion wurde empirisch mit dem Ziel gewählt, Überholvorgänge zu ermöglichen, indem der Einfluss des anderen Fahrzeugs auf den Planungsalgorithmus

Algorithmus 3 Entscheidung für eine Seite zum Überholen

1: Ermitteln der Seite, auf der der Gegner relativ zum Ego-Fahrzeug fährt $\mapsto$ *GegnerSeite*.
2: Bestimme *GewählteSeite* aus dem verfügbaren Raum bis zu $n_{\text{OT,max}}$ in die Zukunft. Verfügbarer Raum $> w_{\text{veh}}$
3: **if** *GewählteSeite* $\neq$ *GegnerSeite* **then**
4: Überhole auf *GewählteSeite*
5: **else**
6: **if** Abstand zum vorausfahrenden Fahrzeug $<$ Spielraum **then**
7: **if** Seitenwechsel ist nicht sicher **then**
8: Wechsle nicht die Seite
9: **end if**
10: **else**
11: Überhole auf Seite mit höher gewichtetem Raumintegral (7-46)
12: **end if**
13: **end if**

des Ego-Fahrzeugs abgeschwächt wird. Die Hauptgründe für diesen Ansatz sind, dass sich die Trajektorien andernfalls stark beeinflussen. Ebenfalls kann nicht angenommen werden, dass der Weg, den der Gegner fahren wird, immer bekannt ist, auch wenn er im Voraus prädiziert wird. Hinzu kommt, dass im Motorsport das vorausfahrende Fahrzeug Vorfahrt besitzt. Das bedeutet, dass es nur gezwungen ist, so viel Platz zu lassen, dass das andere Fahrzeug nicht von der Strecke abkommt.

7.5.3. Eine Seite zum Überholen auswählen

Beim Versuch, den Gegner zu überholen, wird eine weitere Komponente relevant, um den Quer- und/oder Längsraum des eigenen Fahrzeugs einzuschränken, die Entscheidung, auf welcher Seite überholt wird. Das andere Fahrzeug ist für die Problemformulierung essenziell, da es nicht möglich ist, konkave oder disjunkte Räume allein mit linearen Nebenbedingungen zu beschreiben. Um zu entscheiden, auf welcher Seite überholt wird, wurde der folgende logische Algorithmus 3 entwickelt. Ausgangspunkt für den Algorithmus ist eine vorhergesagte Trajektorie des Ego- und des Gegner-Fahrzeugs, die mit der hier vorgestellten Methode berechnet werden. Dabei geht nicht mit ein, ob die Bewegung des Gegner-Fahrzeugs tatsächlich mittels der Methode berechnet wird.

Zunächst wird die Seite, auf der der Gegner fährt, bestimmt. Danach wird für steigende n geprüft, ob auf einer der beiden Seiten ausreichend Platz vorhanden ist. Wenn auf einer Seite nicht genug Platz ist, wird die andere Seite vorausgewählt. Wenn das Überholen auf der vorausgewählten Seite keinen Wechsel der Seiten erfordert, versucht das Fahrzeug auf der Seite zu überholen, auf der es momentan fährt. Ansonsten erfolgt eine Abfrage, ob ein Seitenwechsel sicher ist, um schnelle Richtungsänderungen

des Fahrzeugs zu verhindern. Die sichere Zone definiert sich auf Basis des lateralen Abstands zum Gegner-Fahrzeug ${}_F d_{y,0}({}_E\boldsymbol{p}_{\text{ego}}, {}_E\boldsymbol{p}_{\text{opp}})$ und des longitudinalen Abstands zum Gegner-Fahrzeug ${}_F d_{x,0}({}_E\boldsymbol{p}_{\text{ego}}, {}_E\boldsymbol{p}_{\text{opp}})$. Ein Seitenwechsel wird als sicher eingestuft

$$
\begin{aligned}
&\text{wenn } {}_F d_{x,0}({}_E\boldsymbol{p}_{\text{ego}}, {}_E\boldsymbol{p}_{\text{opp}}) < d_{\text{SZ,co}} \\
&\quad {}_F d_{y,0}({}_E\boldsymbol{p}_{\text{ego}}, {}_E\boldsymbol{p}_{\text{opp}}) \qquad < \frac{w_{\text{veh}}}{2}, \\
&\text{sonst} \\
&\quad {}_F d_{y,0}({}_E\boldsymbol{p}_{\text{ego}}, {}_E\boldsymbol{p}_{\text{opp}}) \qquad < \frac{w_{\text{veh}}}{2} + \left(w_{\text{tr},j} - \frac{w_{\text{veh}}}{2} \right) \\
&\qquad\qquad \cdot \left(\frac{{}_F d_{x,0}({}_E\boldsymbol{p}_{\text{ego}}, {}_E\boldsymbol{p}_{\text{opp}}) - d_{\text{SZ,co}}}{d_{\text{SZ,po}} - d_{\text{SZ,co}}} \right)^2 .
\end{aligned}
$$

Wenn das Fahrzeug näher als $d_{\text{SZ,co}}$ ist, ist die Sicherheits-Zone konstant durch die Fahrzeugbreite definiert, andernfalls steigt die seitliche Abweichung, mit der das Fahrzeug fahren darf, um einen Seitenwechsel durchführen zu dürfen, mit der zweiten Potenz. Ist das Fahrzeug weiter als $d_{\text{SZ,po}}$ entfernt, wird die Begrenzung irrelevant.

Für den Fall, dass die Sicherheitszone keine Entscheidung darüber zulässt, auf welcher Seite überholt wird, wird eine gewichtete Summe für beide Seiten berechnet.

$$
\sum_{n=0}^{n_{\text{OT,max}}} \frac{1}{n+1} \cdot \left| {}_F d_y \left({}_E\boldsymbol{p}_{\text{opp,n}}, {}_E\boldsymbol{p}_{\text{LB|RB},j} \right) \right| \tag{7-46}
$$

Es wird die Seite gewählt, die mehr Platz bietet. Die Summe wird mit dem Zeitschritt n gewichtet, für den der seitliche Abstand berechnet wird. Auf diese Weise wird berücksichtigt, dass die Genauigkeit der Vorhersage mit zunehmendem Zeithorizont abnimmt. Die Summe wird bis zu $n_{\text{OT,max}}$ gebildet, dem größten Index, bis zu dem der Seitenabstand angepasst wird. Dieses Vorgehen ermöglicht Überholmanöver in eine Kurve hinein, da in einer solchen Situation ein Überholen nur möglich ist, wenn die prädizierte Trajektorie des Gegner-Fahrzeugs gekreuzt wird.

7.5.4. Problemformulierung beim Überholen

Wenn eine Seite für das Fahrzeug zum Überholen ausgewählt wurde, werden die Randbedingungen formuliert, um die Kollision der Fahrzeuge zu verhindern. Dies geschieht ähnlich wie beim Nebeneinanderfahren mit der Methode aus Gleichung (7-38). Die wesentlichen Unterschiede sind, dass die vorhergesagte Trajektorie des Gegners berücksichtigt wird, nicht nur die momentane laterale Position. Der Einfluss der Trajektorie auf die Randbedingungen wird aber auch um eine Zone bis zum Index $n_{\text{OT,max}}$ erweitert. In dieser Zone wird die Trajektorie bis $n_{\text{OT,min}}$ voll berücksichtigt.

Danach nimmt der Einfluss der prädizierten Trajektorie mit dem Skalierungsfaktor λ_{OT} bis $n_{\text{OT,max}}$ ab.

$$d_{\text{l|r}}(n) = \max\left(\begin{bmatrix}\lambda_{\text{OT}}(n) \cdot {}_F d_{y,n}\left({}_E\boldsymbol{p}_{\text{opp}}, {}_E\boldsymbol{p}_{\text{LB|RB}}\right)\\ w_{\text{veh}}\end{bmatrix}\right) \tag{7-47}$$

$$\lambda_{\text{OT}}(n) = 1 - \left(\frac{\max\left(\begin{bmatrix}n - n_{\text{OT,min}}\\ 0\end{bmatrix}\right)}{n_{\text{OT,max}} - n_{\text{OT,min}}}\right)^2 \tag{7-48}$$

$$n_{\text{OT,max}} = \frac{\tau_{\text{OT,max}}}{\mathfrak{d}t} \qquad n_{\text{OT,min}} = \frac{\tau_{\text{OT,min}}}{\mathfrak{d}t} \tag{7-49}$$

7.5.5. Problemformulierung zur Vermeidung von Überholvorgängen

In manchen Fällen gibt es keine Seite, die genügend Platz zum Überholen bietet. Dies wird anhand der folgenden Bedingung überprüft:

$$\left|{}_F d_y\left({}_E\boldsymbol{p}_{\text{opp,n}}, {}_E\boldsymbol{p}_{\text{LB|RB},j}\right)\right| < w_{\text{track}} - w_{\text{veh}}. \tag{7-50}$$

In diesem Fall wird eine longitudinale Randbedingung für das Fahrzeug in dem Zeitschritt gesetzt, in dem die Bedingung wahr ist. Dadurch wird der Fortschritt des Fahrzeugs auf eine bestimmte Position auf der Strecke beschränkt, indem eine longitudinale Randbedingung mit einem Offset, wie in Gleichung (7-39) definiert, gesetzt wird. In diesem Fall werden die nachfolgenden Randbedingungen gemäß Abschnitt 7.4.5 adaptiert.

7.5.6. Weitere Bedingungen

Nachdem alle Teile des Optimierungsproblems abgeschlossen sind, wird das Problem mit dem Ziel formuliert, den Fortschritt auf der Strecke ${}_F x_N$ im letzten Zeitschritt N zu maximieren.

Die Randbedingungen $\boldsymbol{C}_n$ bestehen aus den Randbedingungen der Strecke, dem Fahrzeugmodell und allen zusätzlichen Randbedingungen, die aufgrund eines sich in der Nähe befindlichen Fahrzeugs notwendig sind.

$$\boldsymbol{C}_n = \begin{bmatrix} \boldsymbol{c}_{n,\text{tr}} \\ \boldsymbol{c}_{n,\text{acc},1} \\ \cdots \\ \boldsymbol{c}_{n,\text{acc},k_\text{ac}} \\ \boldsymbol{c}_{n,P} \\ (\boldsymbol{c}_{n,\text{opp}}) \end{bmatrix} \tag{7-51}$$

Um zu gewährleisten, dass die berechnete Trajektorie immer ein sicheres Ende hat und das Fahrzeug in der Lage ist, zum Stillstand zu kommen, auch wenn der Trajektorienplaner plötzlich versagen würde, wird zusätzlich für den letzten Zeitschritt die Bedingung

$$\begin{bmatrix} {}_E v_{x,N} \\ {}_E v_{y,N} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{7-52}$$

definiert. Das gesamte Optimierungsproblem ist in Anhang A zusammengefasst.

7.5.7. Verifikation der Algorithmen

Zur Verifikation des zuvor vorgestellten Trajektorienplaners und dem dazu gehörigen Bewegungsmodells wurde eine Simulationsumgebung entwickelt, die erlaubt, den vorgestellten Planer in einer Rennsimulation zu verwenden. Diese Umgebung und die Ergebnisse der Verifikation des Modells werden im Folgenden in Kürze vorgestellt.

Für die Verifikation wurde die Rennsimulation Project Cars 2 von der Firma Slightlymad Studios verwendet. Die Steuerung von Lenkung und Gaspedal erfolgt über die vJoy-Schnittstelle in Windows. Für diesen Test ist nur das Ego-Fahrzeug steuerbar. Das Gegner-Fahrzeug wird durch das Spiel gesteuert. Die Regelung des Gaspedals des Fahrzeugs ist durch einen PD-Regler realisiert, während die laterale Regelung von einem Deichselregler übernommen wird, der in Abbildung 7-6 dargestellt ist. Die Grundidee ist, dass die Vorderräder des Fahrzeugs auf die Position ${}_E\boldsymbol{p}({}_F x_\text{ref})$ gerichtet sind, die die Sollposition des Fahrzeugs in τ_steer Sekunden darstellt. Zusätzlich zur Vorschauzeit τ_steer der Lenkung errechnet sich diese Sollposition aus der Ist-Position des Ego-Fahrzeugs in Frenet-Koordinaten ${}_F x_0$ und der momentanen, longitudinalen Geschwindigkeit des Ego-Fahrzeugs in Frenet-Koordinaten ${}_F v_x$. Daher wird der Lenkwinkel δ durch folgendes Regelgesetz beschrieben.

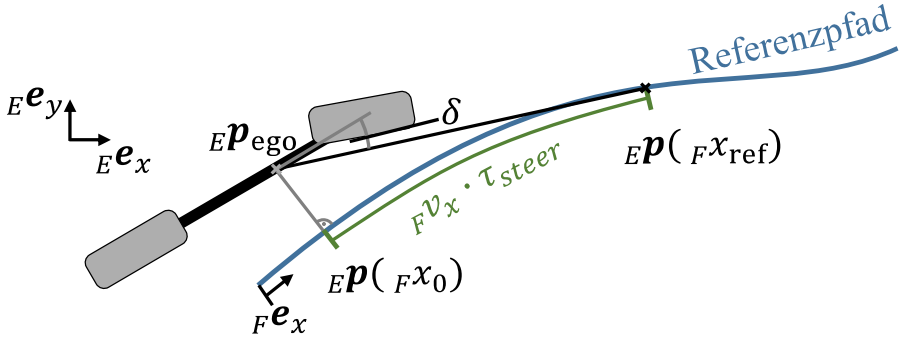


Abbildung 7-6.: Visualisierung des Deichsel-Regler-Konzepts, das für die laterale Regelung der Trajektorie in der Rennsimulation verwendet wird.

$$\delta = \text{atan2}\left({}_E\boldsymbol{p}({}_F x_{\text{ref}}) - {}_E\boldsymbol{p}_{\text{ego}}\right) \tag{7-53}$$

$${}_F x_{\text{ref}} = {}_F x_0 + {}_F v_x \tau_{\text{steer}} \tag{7-54}$$

Indem nur das Ego-Fahrzeug gesteuert wird, ist es möglich, die Gültigkeit des Ansatzes, eine Vorhersage des anderen Fahrzeugs zu nutzen, um das eigene Verhalten zu planen, zu testen. Die Ergebnisse dieses Tests zeigen nicht nur, dass die Trajektorien, die der Algorithmus plant, zu sehr schnellen Rundenzeiten führen, sondern auch, dass der Algorithmus in der Lage ist, kollisionsfreie Überholmanöver im fahrdynamischen Grenzbereich zu zeigen. Ein Video von diesem Test ist auf TU Datalib [210] veröffentlicht. Das Video zeigt eine Reihe von Überholmanövern mit dem vorgestellten LCADM-Algorithmus und dem Referenzalgorithmus in dem Rennspiel Project Cars 2 sowie die Leistung des Reglers beim Fahren ohne andere Fahrzeuge.

7.5.8. Evaluation der Rechenzeit

Eine Evaluation der Geschwindigkeit des Planungs-Algorithmus ergab eine Rechenzeit von $3\,\text{ms}$ pro Rechenschritt pro Fahrzeug. Für jede Trajektorie werden dafür $N = 50$ Zeitschritte mit einer Schrittweite von $\eth t = 0{,}2\,\text{s}$ berechnet. Damit ergibt sich ein Zeithorizont der Trajektorie von $10\,\text{s}$. Die dabei verwendete Hardware ist ein AMD Ryzen 9 3900 XT 12 Kerne Prozessor mit 32 GB Arbeitsspeicher. Die Berechnung des Algorithmus ist bedingt durch die Implementierung auf mehreren Kernen gleichzeitig möglich, sodass auf der genannten Hardware Berechnungsraten bis zu $3\,\text{kHz}$ erreicht werden. Der genutzte Solver dafür ist hpipm[211], der das Blasfeo[212] Paket für Subroutinen der linearen Algebra für die Lösung der Gleichungen nutzt. Der Code für den

[210] Sippel, M. et al.: Result Video: Two Behavior Algorithms Driving the Racecar (2020).

[211] Frison, G. et al.: HPIPM: a high-performance quadratic programming framework (2020).

[212] Frison, G. et al.: BLASFEO (2018).

Trajektorienplaner wurde bereits vor-veröffentlicht und ist online abrufbar.[213] Alle weiteren Hyperparameter des Verfahrens sind in Anhang A zusammengefasst.

7.6. Neuronales Netz

Die beiden bisher vorgestellten Ansätze sind maßgeblich darauf ausgelegt, einem Agent das Überholen auf der Rennstrecke zu ermöglichen und vernachlässigen die Aufgabe, ein anderes Fahrzeug am Überholen zu hindern. Erste Untersuchungen von Liniger *et al*. zu dem Thema ergaben, dass eine Strategie, die aktiv versucht, ein anderes Fahrzeug am Überholen zu hindern, tatsächlich die Chancen des hinten fahrenden Fahrzeugs steigert, erfolgreich zu überholen.[214] Kloock *et al*. zeigen einen Algorithmus, der versucht, bei einem sich von hinten nähernden anderen Fahrzeug dieselbe laterale Position auf der Strecke einzunehmen und dadurch ein Überholmanöver zu verhindern. Allerdings bieten sie keine quantitative Auswertung der Methode.[215] Dazu kommt, dass ein solches Verhalten zwar auf der Geraden möglicherweise ein Überholmanöver verhindert, allerdings ist durch die Regeln im Motorsport ein solches Verhalten nur begrenzt möglich, da ein Blocken des Gegners auf der Geraden potenziell gefährlich ist.[216] Aus diesem Grund sind Richtungswechsel, um ein Überholen zu verhindern, auf einer Geraden nur einmal pro Anfahrt auf eine Kurve erlaubt.[217]

Zur Problematik von riskantem Blocken auf der Geraden kommt, dass die Fahrt auf der identischen lateralen Position in der Kurve einem überholenden Fahrzeug das Überholen erleichtert, da beim Anfahren einer Kurve durch dieses Verhalten dem überholenden Fahrzeug Platz zum Überholen geschaffen wird.

Aus genannten Gründen wurde im Rahmen dieser Arbeit ein Ansatz auf Basis von einem NN entwickelt. Das NN wird der darauf trainiert, andere Fahrzeuge am Überholen zu hindern und auf diese Weise auch den Einfluss von defensiven Strategien im Benchmarking zu evaluieren. Dazu wurde ebenfalls ein Szenario definiert, in dem dem Algorithmus ermöglicht wird, das Überholen von anderen Fahrzeugen zu erlernen. Beide Agenten werden im Zuge der beispielhaften Anwendung der vorgestellten Ranking-Methode in Abschnitt 8.2 evaluiert. Ein NN bietet den Vorteil, dass der Prozess, der das Netz zum gewünschten Verhalten hin optimiert, zum großen Teil durch die statistische Verteilung der verwendeten Samples beeinflusst wird.[218] Auf diese Weise wird eine Strategie erlernt, die nicht nur in einem Einzelfall funktioniert, sondern die für eine möglichst große Zahl an verschiedenen Situationen zum Erfolg führt.

[213] Sippel, M. et al.: Source Code: Sequential Linearized Motion Planner for Racecar Behavior (2020).

[214] Vgl. Liniger, A. et al.: A Non-Cooperative Game Approach to Autonomous Racing (2017).

[215] Vgl. Kloock, M. et al.: Networked Model Predictive Vehicle Race Control (2019).

[216] Vgl. Onieva, E. et al.: Overtaking opponents with blocking strategies using fuzzy logic (2010).

[217] Vgl. FIA: Appendix L - Driver's licenses, equipment and conduct (2021), S. 47f.

[218] Vgl. Zhang, J.: Gradient Descent based Optimization Algorithms for Deep Learning (2019).

Es ist anzumerken, dass die vorgestellte Implementierung nur eine mögliche unter vielen darstellt und keine Evaluierung hinsichtlich der Performance im Vergleich mit anderen Umfeldrepräsentationen und Aktionen durchgeführt wurde. Die Implementierung verfolgt nicht das Ziel, einen möglichst performanten Agenten zu generieren. Vielmehr war die Zielsetzung, bei der Implementierung einen passenden Vergleichsagenten zu kreieren.

Der entwickelte Ansatz wird im Folgenden im Details vorgestellt. Abschnitt 7.6.1 beschreibt zuerst die Umfeldrepräsentation, die für die Umsetzung definiert wurde und damit die Variablen, die für das NN wahrnehmbar sind, sowie die gewählte Netz-Struktur. Als Zweites geht Abschnitt 7.6.2 auf die Möglichkeiten ein, die dem Netz gegeben werden, um den Trajektorienplaner zu beeinflussen und so ein Überholmanöver zu verhindern bzw. auszuführen. Abschließend werden in Abschnitt 7.6.3 die definierten Trainingsszenarien beschrieben, in denen der Algorithmus trainiert wird, zusammen mit den definierten Reward-Funktionen.

7.6.1. Zustandsraum und Netzstruktur

Der Zustandsraum, der für den Algorithmus sichtbar ist, bildet die Basis, auf der der Algorithmus sein Verhalten errechnet. Er teilt sich in Parameter, die (a) das eigene Fahrzeug beschreiben, (b) Parameter, die die Strecke beschreiben, und (c) Parameter, die andere statische oder dynamische Objekte beschreiben.

Die Parameter, die das eigene Fahrverhalten und die Fahrbahn beschreiben sind:

- Geschwindigkeit/Beschleunigung in x- und y-Richtung des Fahrzeugs
- Lateraler Abstand zwischen Fahrzeug und Fahrbahnbegrenzung zur linken bzw. rechten Seite
- Lateraler Abstand zur berechneten Ideallinie im nächsten Zeitschritt
- Geschwindigkeitsdifferenz zur berechneten Ideallinie
- Fahrbahnkrümmung an n_{curv} Punkten in je $10\,\text{m}$ Abständen

Das Umfeld des Fahrzeugs wird beschrieben durch einen virtuellen Abstandssensor, der den normierten Abstand zu umliegenden anderen Fahrzeugen für n_{beam} äquidistante Strahlen beschreibt. Das Prinzip ist in Abbildung 7-7 visualisiert. Auf diese Weise ist es möglich, auch gegen mehrere Fahrzeuge zu trainieren, ohne die Netzstruktur anzupassen. Das gleiche Prinzip wird ebenfalls verwendet, um den Abstand zum Streckenrand zu beschreiben.

Die gewählte Netzstruktur ist in Abbildung 7-8 abgebildet. Die beobachteten Metriken werden in einem rekursiven LSTM-Layer (engl. Long Short Term Memory) verarbeitet und danach in zwei Zweige aus jeweils einem Block aufgeteilt. Ein **Layer** beschreibt

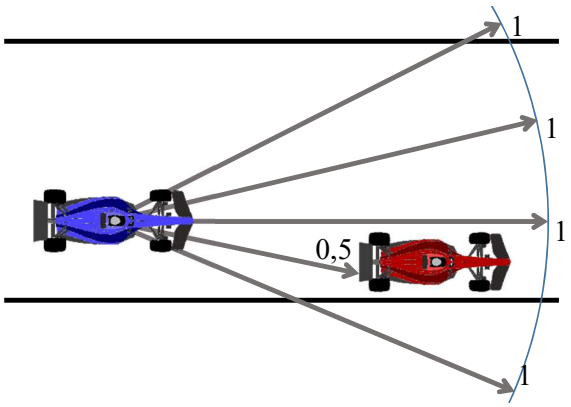


Abbildung 7-7.: Darstellung der Funktionsweise des virtuellen Abstandssensors für das Beispiel eines anderen Fahrzeugs

dabei eine einzelne Schicht des NN, inklusive der zugehörigen Aktivierungsfunktion. Die Bezeichnung **Block** bezeichnet dabei den Zusammenschluss mehrerer Layer. Das LSTM-Layer ist im Gegensatz zu einem normalen Dense-Layer in der Lage, Zustände zwischenzuspeichern und auf diese Weise dynamischer auf sich entwickelnde Situationen zu reagieren.[219] Die zwei Zweige bestehen jeweils aus zwei Dense-Layern und einer Aktivierungsschicht. Der linke Zweig ist der Policy-Block, der die gewählte Aktion ausgibt. Der Value-Block beurteilt die Güte der ausgeführten Aktion. Anhand von beiden Ausgängen und den von der Umgebung generierten Rewards für den momentanen Zustand wird das Netz mittels PPO (engl. Proximal Policy Optimization) angelernt. Der wesentliche Kernpunkt der Methode ist, dass eine schnelle Entwicklung des Netzes durch die Beschränkung des Gradienten während der Optimierung begrenzt wird. Auf diese Weise ist die Methode weniger anfällig für die Beeinflussung durch einzelne Ereignisse, da die Anpassungrate pro Optimierungsschritt begrenzt ist.[220]

Das Lernverfahren setzt sich aus zwei Phasen zusammen. Der Agent agiert für eine bestimmte Anzahl an Zeitschritten innerhalb eines Szenarios. Die Aktionen des Agenten, die Beobachtungen aus der Umgebung und die resultierenden Rewards werden jeweils in einem Erfahrungsspeicher festgehalten. Ist die maximale Anzahl an Zeitschritten erreicht, wird das NN gemäß des PPO-Verfahrens optimiert. Die Datenmenge, die für die Optimierung verwendet wird, wird als **Batch** bezeichnet. Danach wird der Erfahrungsspeicher geleert und der Agent sammelt erneut weitere Erfahrungen.

Die gewählten Hyperparameter für das Lernverfahren sowie der detaillierte Aufbau des Netzes finden sich in Anhang C.

[219] Vgl. Hochreiter, S. et al.: Long Short-term Memory (1997).

[220] Vgl. Schulman, J. et al.: Proximal Policy Optimization Algorithms (2017).

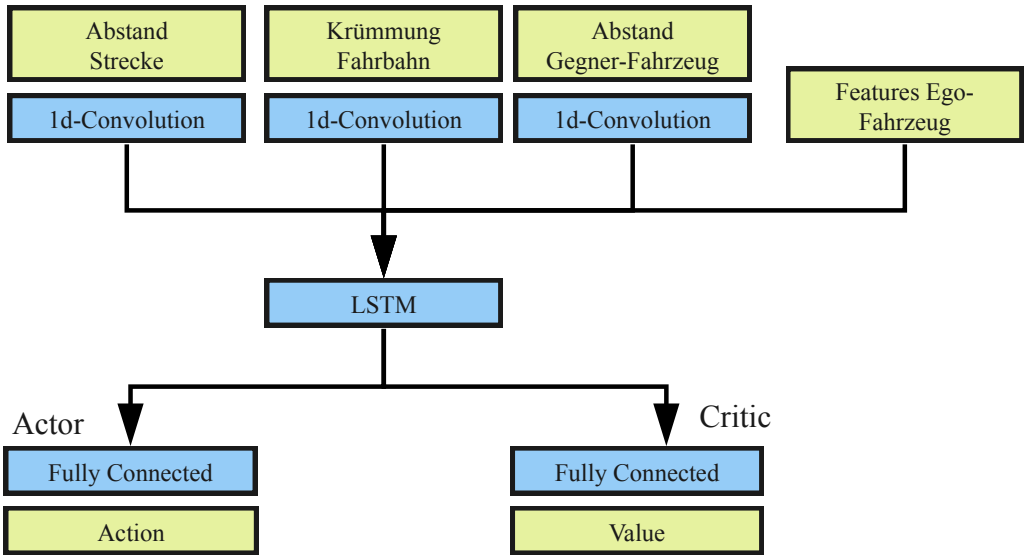


Abbildung 7-8.: Struktur des gewählten NN. Die berechneten Metriken werden in ein rekursives LSTM Layer geleitet. Daraus zweigen sich die Werte in den Actor-Block (Aktion des NN) und den Critic-Block (Bewertung des Zustands) ab.

7.6.2. Actionsraum

Die Aktion, die durch den Policy-Block generiert wird, gibt drei kontinuierliche Werte $(u_{\text{width}}, u_{\text{lat}}, u_{\text{ref}}) \in [-1, 1]$ aus. Um die laterale Position des Fahrzeugs zu kontrollieren, wird eine Randbedingung generiert, die einen Korridor für den Zeitschritt $n = 5$ darstellt.[221] Dieser Korridor wird vom NN über die Action auf eine dedizierte Breite u_{width} und laterale Position u_{lat} parametriert. Der zu parametrierende Korridor ist in Abbildung 7-9 dargestellt. Die erstellte Randbedingung ist entsprechend den Gleichungen (7-35)-(7-37) für die räumliche Begrenzung der Planung definiert durch:

$$ {}_E\boldsymbol{p}_{\text{u}} = {}_E\boldsymbol{p}_{\text{RB}} + ({}_E\boldsymbol{p}_{\text{LB}} - {}_E\boldsymbol{p}_{\text{RB}}) \left(\frac{1 + u_{\text{lat}}}{2} + \frac{1 + u_{\text{width}}}{4} \right) \tag{7-55} $$

$$ {}_E\boldsymbol{p}_{\text{l}} = {}_E\boldsymbol{p}_{\text{RB}} + ({}_E\boldsymbol{p}_{\text{LB}} - {}_E\boldsymbol{p}_{\text{RB}}) \left(\frac{1 + u_{\text{lat}}}{2} - \frac{1 + u_{\text{width}}}{4} \right) \tag{7-56} $$

$$ \text{mit } 0 \leq \left(\frac{1 + u_{\text{lat}}}{2} - \frac{1 + u_{\text{width}}}{4} \right), \left(\frac{1 + u_{\text{lat}}}{2} - \frac{1 + u_{\text{width}}}{4} \right) \leq 1 \tag{7-57} $$

Der Term $\frac{1+u_{\text{lat|width}}}{2}$ normiert dabei die Aktionen u_{lat} und u_{width} des NN jeweils von $[-1, 1]$ auf $[0, 1]$. Auf diese Weise ist das NN in der Lage, die laterale Position des Fahrzeugs auf der Strecke zu beeinflussen.

[221] Der Zeitschritt $n = 5$ ist mit einer Sekunde ausreichend weit in der Zukunft, um ein unlösbares Problem durch eine extrem gesetzte Randbedingung zu vermeiden aber nicht so weit in der Zukunft, dass er keine Einfluss mehr auf das Verhalten des Fahrzeugs besitzt.

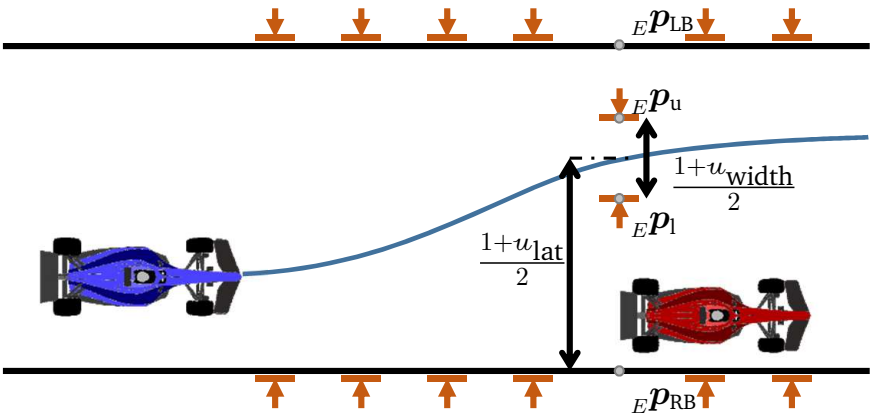


Abbildung 7-9.: Adaption der Randbedingungen durch NN

Zusätzlich zur lateralen Beeinflussung werden für den Agenten zu jedem Zeitschritt die Randbedingungen der Referenzmethode gesetzt, da diese ermöglichen, sowohl in longitudinaler als auch lateraler Richtung eine Kollision zu verhindern. Der dritte Parameter u_{ref} normiert die Gewichtung der Randbedingungen linear, sodass für einen Wert von $u_{\mathrm{ref}} = 1$ die Randbedingungen mit der identischen Gewichtung der Referenzmethode im Optimierungsproblem berücksichtigt werden. Für einen Wert von $u_{\mathrm{ref}} = -1$ besitzen die Randbedingungen eine Gewichtung von 0.

7.6.3. Trainingsszenario

Das Szenario, in dem der Agent trainiert wird, ist wie folgt aufgebaut. Der Agent startet das Szenario mit einer festen Geschwindigkeit, die für jedes Szenario identisch bei $10\,\frac{\mathrm{m}}{\mathrm{s}}$ für das Ego-Fahrzeug und für das Gegner-Fahrzeug liegt. Auf diese Weise ist möglich, für jede Startposition identische Startbedingungen zu schaffen. Der Ansatz, die Startgeschwindigkeiten je nach Streckenverlauf zu variieren oder unterschiedliche Anfangsgeschwindigkeiten für die beiden Fahrzeuge zu wählen, wurde an diese Stelle verworfen, da nur durch eine immer identische Startgeschwindigkeit eine gleichwertige Ausgangslage garantiert wird. Andernfalls entstehen häufig Startbedingungen, die eines der beiden Fahrzeuge bevorzugen oder auch ein Szenario für eines der Fahrzeuge erzeugen, das dem Agenten keine Lösung ermöglicht, wenn die Geschwindigkeit die maximal mögliche Geschwindigkeit an einer Stelle der Strecke übersteigt. Der Abstand zwischen beiden Fahrzeugen wird zu $20\,\mathrm{m}$ definiert. Damit wird ein Startzustand geschaffen, der einem fliegenden Start oder auch dem Zustand der gelben Flagge in einem Rennen ähnelt.[222] Die laterale Position der beiden Fahrzeuge wird für beide Fahrzeuge zufällig bestimmt.

Für den Fall des defensiven Szenarios wird das Gegner-Fahrzeug hinter dem Ego-

[222]Die gelbe Flagge wird in Rennen üblicherweise bei Gefahrenstellen bspw. durch einen Unfall auf der Strecke geschwenkt. In einem solchen Fall ist es den Fahrzeugen auf der Strecke verboten einander zu überholen bzw. eine bestimmte Geschwindigkeit zu überschreiten.

Fahrzeug positioniert, das versucht, den Agenten zu überholen. Das Ego-Fahrzeug wird gemäß des vorgestellten Parameterraumes aus Abschnitt 6.3 zufällig parametrisiert, mit der einzigen Bedingung, dass es eine niedrigere Rundenzeit fährt als das Gegner-Fahrzeug. Die Reward-Struktur für den Agenten ist einfach gehalten, um einen Mis-Use durch den Agenten so schwer wie möglich zu gestalten. Die Struktur ist in Tabelle 7-1 dargestellt. Das Szenario endet, wenn entweder eine Kollision zwischen den beiden Fahrzeugen eintritt, das Ende der Strecke bzw. Runde erreicht oder das gegnerische Fahrzeug erfolgreich überholt wird. An dieser Stelle gilt ein Szenario als erfolgreich überholt, sobald das hinten fahrende Fahrzeug in Streckenkoordinaten eine Position vor dem Ego-Fahrzeug einnimmt. Die Reward-Funktion zeigt, wie bereits geschildert, den Unterschied auf zwischen einer Reward-Funktion, die genutzt wird, um einen Algorithmus anzulernen, im Vergleich mit einer Funktion, die für eine Evaluation genutzt wird. Die Kollision führt nicht zu einem negativen Reward, da andernfalls der Agent eine Optimierung hin zu der Vermeidung einer Kollision durchführt und nicht versucht, ein Überholen an sich zu verhindern.

Tabelle 7-1.: Reward-Funktion für Trainingsszenario mit Ziel des Verteidigens

Punktzahl	Event
+1	Bei jedem Zeitschritt
	Wert auf Basis der Relativgeschwindigkeit nach Song[223a] pro Zeitschritt
-100	Überholt durch gegnerisches Fahrzeug
+100	Erreichen des Streckenendes oder Zeitlimits
0	Kollision mit anderem Fahrzeug

Ähnlich verhält es sich für das Überholszenario. Die Randbedingungen des Szenarios sind dabei identisch mit dem Unterschied, dass das Ego-Fahrzeug das Ziel erhält, das andere Fahrzeug zu überholen. Dementsprechend ist das Gegner-Fahrzeug vor dem Ego-Fahrzeug platziert und zufällig innerhalb des Parameterraumes aus Abschnitt 6.3 parametriert. Die Reward-Funktion, die in Tabelle 7-2 zusammengefasst ist, gibt lediglich einen Reward für ein erfolgreiches Überholen und für die Relativgeschwindigkeit zwischen den beiden Fahrzeugen. Der Agent erhält also einen Reward, wenn er sich dem anderen Fahrzeug nähert und verliert Punkte, wenn er sich vom anderen Fahrzeug entfernt.

Tabelle 7-2.: Reward-Funktion für Trainingsszenario mit Ziel des Überholens

Punktzahl	Event
	Wert auf Basis der Relativgeschwindigkeit nach Song[223b] pro Zeitschritt
+50	Überholt gegnerisches Fahrzeug
0	Erreichen des Streckenendes oder Zeitlimits
0	Kollision mit anderem Fahrzeug

[223]Song, Y. et al.: Autonomous Overtaking in Gran Turismo Sport Using Curriculum RL (2021).

Die erzeugten Szenarien produzieren sehr unterschiedliche Inputs für das NN in Bezug auf die Komplexität des Szenarios und damit verbunden der Möglichkeit, es erfolgreich abzuschließen. Aus der Analyse der Studie von Knox *et al.*[225] geht hervor, dass beim Design der Reward-Funktion gefordert wird, dass es durch ein Ranking verschiedener möglicher Ausgänge und dem resultierenden Reward für einzelne Zeitschritte erfolgt. Ein erfolgreiches Überholmanöver besitzt hier die höchste Priorität und es ist gefordert, dass eine Kollision durch die Reward-Funktion vermieden wird und ist daher niedriger priorisiert. Nur wenn diese Rangfolge der Prioritäten eingehalten und von der Reward-Funktion abgebildet wird, ist es möglich, den Agenten auf das gewünschte Verhalten zu trainieren. Je vielfältiger an dieser Stelle die Szenarien innerhalb eines Batches sind, desto wahrscheinlicher ist eine potenzielle Störung der Rangfolge und Priorisierung der Szenario-Ergebnisse.

An dieser Stelle spielt insbesondere die Startposition auf der Strecke eine große Rolle, da sich ein Überholmanöver auf einer langen Geraden einfacher umsetzen lässt als in einer Kombination aus Kurven. Aus diesem Grund wird für jeden Batch immer dieselbe Startposition verwendet, um die Vergleichbarkeit der Erfahrungen innerhalb einer Lerneinheit möglichst hochzuhalten. Die Startposition bzw. die Strecke werden nach jedem Lernvorgang zufällig neu bestimmt, wenn der Erfahrungsspeicher geleert wird. Ebenso verhält es sich mit unterschiedlichen Arten von Szenarien, die möglicherweise auch unterschiedliche Reward-Funktionen besitzen und daher nicht innerhalb eines Lernzyklus als Erfahrung nutzbar sind. Werden verschiedenen Szenarien mit verschiedenen Reward-Funktionen in einem Batch zum Lernen verwendet, werden lediglich die Verhaltensweisen verstärkt, die bereits erfolgreich gelernt wurden. Denn Strategien, die zu einer schnellen Erhöhung des Rewards führen, werden aufgrund der Optimierungsalgorithmen zuerst erlernt. Eine dedizierte Situation, in der der Algorithmus bislang aber keine guten Ergebnisse erzielt, geht bei einer Mischung von verschiedenen Szenarien und Startbedingungen in der Masse der anderen Ergebnisse unter, da (a) die Situation, für die der Algorithmus keine erfolgreiche Strategie gelernt hat, nur einen kleinen Anteil der Daten ausmacht und (b) wenn die bisher nicht gelernte Situation erfolgreich durchgeführt wird, andere Szenarien mit hoher Wahrscheinlichkeit trotzdem einen höheren Reward erzielen.

7.7. Testumgebung

Die Durchführung der Tests im Rahmen dieser Arbeit setzen eine Testumgebung voraus, in der verschiedenen Arten von Verhaltensalgorithmen in der Lage sind, sich miteinander auf einer gemeinsamen virtuellen Strecke zu bewegen. Dafür wurde das in Abbildung 7-10 dargestellte Framework verwendet. Die Testumgebung, in der die einzelnen Parameter für jeden Agenten hinterlegt sind, besitzt ein Fahrzeug-Modell,

[225] Knox, W. B. et al.: Reward (Mis)design for Autonomous Driving (2021).

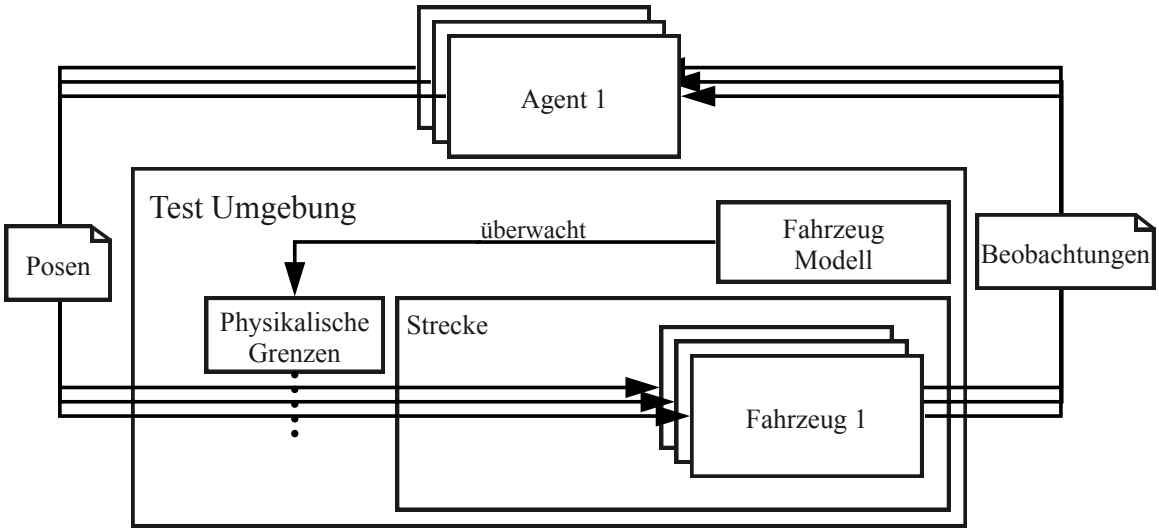


Abbildung 7-10.: Architektur des verwendeten Test-Frameworks

das für jeden Zeitschritt überwacht, dass die physikalischen Grenzen eingehalten werden. Die Kontrolle über das Fahrzeug liegt aber zur Gänze beim Agenten direkt. Er erhält eine Beobachtung aus der Umgebung und bestimmt daraus eigenständig seine neue Pose. Damit ist das Framework in der Lage, auch mehrere verschiedene Arten von Planungsalgorithmen miteinander zu testen, da keine Abhängigkeit bezüglich etwaiger Regelalgorithmen oder Planern besteht.

8. Beispielhafte Anwendung des Benchmarkings zur Analyse mehrerer Algorithmen

Die in Kapitel 7 vorgestellten Algorithmen stellen drei grundsätzlich unterschiedliche Ansätze für das Problem des Überholens dar, auch wenn sie auf demselben Bewegungsmodell basieren. Der Referenzalgorithmus geht von einer vorberechneten Lösung aus und stellt die Randbedingungen für das eigene Fahrzeug anhand dieser Lösung auf. Dabei werden zu jedem Zeitpunkt die Randbedingungen so formuliert, dass es nicht zu einer Überschneidung der Trajektorien der beiden Fahrzeuge kommt.

Der LCADM-Algorithmus geht davon aus, dass eine vorberechnete Lösung für ein anderes Fahrzeug existiert. Es wird angenommen, dass die Genauigkeit, mit der die berechnete Lösung mit der tatsächlichen übereinstimmt, mit einem wachsenden Zeithorizont sinkt. Die Methode ermöglicht, dass der Einfluss, den die berechnete Lösung auf die Trajektorie des Ego-Fahrzeugs nimmt, ebenfalls mit einem länger werdenden Prädiktionshorizont schwindet. Auf diese Weise sind auch aggressive Überholmanöver in eine Kurve hinein möglich.

Der NN-basierte Ansatz wird in einem Lernszenario angelernt, sowohl ein Überholen anderer Algorithmen zu verhindern als auch das Überholen eines anderen Fahrzeugs zu ermöglichen. Dabei basiert das Verhalten auf der definierten Reward-Funktion im Szenario. Durch den hybriden Ansatz von MPC und NN ist der Algorithmus in der Lage, auf beliebigen Strecken zu fahren, der Erfolg hängt dabei aber von der Generalisierbarkeit der Trainingsbedingungen und deren Übertragbarkeit auf ein bestimmtes Szenario ab.

Gemäß der Methode, die in dieser Arbeit vorgestellt wurde, teilt sich die Evaluierung der Algorithmen in zwei wesentliche Bereiche. Die qualitative Evaluierung in Abschnitt 8.1 vergleicht die beiden deterministischen Ansätze des Referenz- und des LCADM-Algorithmus miteinander. Dabei wird insbesondere auf Schwächen der einzelnen Algorithmen eingegangen. Im Anschluss daran werden die drei vorgestellten Algorithmen in Abschnitt 8.2 im Rahmen eines Rankings direkt miteinander verglichen und die Ergebnisse analysiert. Der Abschnitt diskutiert ebenfalls, inwieweit die Ergebnisse durch die gewählten Testbedingungen und den Einsatz einer diskreten gegenüber einer kontinuierlichen Ergebnismetrik beeinflusst werden. Das berechnete Ranking erlaubt ebenfalls die Beantwortung von Forschungsfrage 4 nach dem Einfluss der intransitiven Relationen auf die Möglichkeit und die Bedingungen für die Bildung eines Rankings von Algorithmen im Rahmen dieser Arbeit.

8.1. Vergleich des Überholverhaltens zweier Algorithmen

Die Erwartung an den Vergleich der Algorithmen ist, dass sich die genannten grundsätzlichen Unterschiede im qualitativen Vergleich zeigen. Anhand der Daten werden ebenfalls die jeweiligen Schwachstellen der beiden Algorithmen untersucht. Der Vergleich der Algorithmen auf zwei Strecken zeigt zusätzlich die Diskrepanz zwischen den Tests auf zufällig generierten Strecken und einem realen Layout. Der Einfluss der Eigenschaften der Algorithmen zeigt sich dabei stark abhängig von dem spezifischen Layout einer Strecke.

Da beide Algorithmen nicht über Mechanismen verfügen, ein Überholen von einem Gegner-Fahrzeug zu verhindern, wird in diesem Abschnitt nur auf das Überholverhalten der Algorithmen eingegangen. Die Bedingungen, unter denen die Tests durchgeführt wurden, werden im Folgenden vorgestellt.

8.1.1. Testbedingungen

Der Test der Algorithmen erfolgt zunächst in einem zufällig generiertem Szenario gemäß Abschnitt 6.2.1 und danach auf den beiden Grandprix-Strecken Hockenheim und Barcelona-Catalunya. Die beiden Strecken sind dabei als Vertreter von Strecken mit verschiedenen Layouts und Einflüssen der Parametrisierung auf die erreichbaren Rundenzeiten anzusehen. Bei dem Test auf realen Strecken sind die Startpositionen jeweils zufällig über der Länge der Runde verteilt und werden für jedes Szenario zufällig ausgewählt. Aufgrund der Vielfältigkeit des Streckenlayouts werden die Szenarien auf jeweils 2000 zufällig generierten Layouts durchgeführt. Die Szenarien auf realen Strecken werden dagegen nur 200 Mal mit unterschiedlichen Startpositionen wiederholt.

Auf Basis der Analyse der verfügbaren dynamischen Objekte in Abschnitt 6.1 wird das Gegner-Fahrzeug mithilfe des LCADM-Algorithmus gesteuert, da dieser nicht auf schnell fahrende Fahrzeuge von hinten reagiert. Lediglich, wenn beide Fahrzeuge nebeneinander fahren, sorgt der Algorithmus dafür, dass dem anderen Fahrzeug mindestens eine Fahrzeugbreite Platz zur Verfügung steht.

Die Parametrisierung des Gegner-Fahrzeugs wird nach der Betrachtung des Einflusses in Abschnitt 6.3 anhand von zwei unterschiedlichen Parameter-Gittern festgelegt. Für die zufällig generierten Szenarien wird ein äquidistantes 5x5-Gitter für die maximale Querbeschleunigung und die Vortriebsleistung aufgespannt mit:

$$P_{\text{max}} \in [300\,\text{kW}, 450\,\text{kW}], a_{\text{lat,max}} \in \left[17\,\frac{\text{m}}{\text{s}^2}, 23\,\frac{\text{m}}{\text{s}^2}\right], D_{\text{max}} = a_{\text{lat,max}} \tag{8-1}$$

Das Gitter wurde so definiert, dass für die schnellste Gegner-Parametrisierung kein Überholen mehr möglich ist. Die Kürze des zufällig generierten Szenarios erlaubt hier, dass Fahrzeuge überholt werden, die in der Lage sind, schneller als das Ego-Fahrzeug zu fahren.

Das Ego-Fahrzeug fährt mit der Referenz-Parametrisierung von:

$$P_{\text{max}} = 400\,\text{kW}, a_{\text{lat,max}} = 20\,\frac{\text{m}}{\text{s}^2}, D_{\text{max}} = 20\,\frac{\text{m}}{\text{s}^2} \tag{8-2}$$

Die Parametrisierungen für den Test auf den beiden realen Streckenlayouts werden jeweils anhand der Rundenzeit und der Vortriebsleistung ermittelt. Auch hier wird ein für den jeweiligen Parameter äquidistantes 5x5-Gitter aufgespannt. Die genauen Werte sind in Anhang D.1 festgehalten.

Untersucht wurden drei wesentliche Metriken. Der **Ausgang eines Szenarios** beschreibt, (a) ob ein Szenario mit einem erfolgreichen Überholmanöver beendet wurde, (b) ob es zu einer Kollision zwischen beiden Fahrzeugen kam oder (c) kein Überholmanöver stattgefunden hat. Die verwendeten Abbildungen zeigen jeweils für eine Parametrisierung jeweils für jedes gefahrene Szenario den Ausgang (Farbe) an. Die Ordinate/Abszisse geben jeweils einen definierten Parameter des Szenarios oder eine Metrik, die das Ergebnis beschreibt (z. B. an welcher Stelle der Strecke hat das Szenario geendet), an.

Die **Überholposition** ist definiert als die letzte Position, an der ${}_F p_{x,\text{opp}} > {}_F p_{x,\text{ego}}$. Im Falle eines Überholmanövers ist das die Position auf der Strecke, an der die beiden Fahrzeuge die Position gewechselt haben. Die Position ist ebenfalls definiert, wenn kein Überholmanöver stattgefunden hat, denn auch in diesem Fall ist ein temporärer Positionswechsel möglich. Erfolgte in der gegebenen Strecke kein Überholmanöver und kein Positionswechsel, ist der Wert der letzte gefahrene Zeitschritt. Im Falle eines zufällig generiertem Szenarios ist die Überholposition auf die jeweilige Länge der Streckensegmente normiert, um die Vergleichbarkeit zwischen den Streckenlayouts herzustellen. (0: Start von Segment 1, 1: Start von Segment 2 ...) Bei Szenarien auf einem realen Streckenlayout ist die Position in realen Streckenkoordinaten angegeben. Für das zufällige Layout definiert sich das Ende dabei durch das Ende der Strecke. Im Falle der realen Strecke wird als Limit eine volle gefahrene Runde angenommen. Die Auswirkungen dieser Annahmen werden in Abschnitt 8.2 zusammen mit den Ergebnissen der Tests diskutiert.

Die **verlorene Zeit** τ_{loss} während eines Manövers wird gemessen mittels eines zweiten Ego-Fahrzeugs, das mit dem identischem Fahrzeugmodell und -parametrisierung des Ego-Fahrzeugs fährt, aber keine Beeinflussung durch das Gegner-Fahrzeug erfährt.

Bei Überqueren der Ziellinie vom Ego-Fahrzeug endet das Szenario. Der zeitliche Unterschied wird zu diesem Zeitpunkt ermittelt, indem der Weg-Zeit-Verlauf des sich weiter vorne befindlichen Fahrzeugs auf die momentane Position des sich weiter hinten befindlichen Fahrzeugs interpoliert wird.

8.1.2. Ergebnisse

Die Ergebnisse teilen sich in drei wesentliche Teile. Im ersten Teil werden die Ereignis-Häufigkeiten für beide Algorithmen analysiert und mit den bereits beschriebenen Erwartungen verglichen. Danach folgt eine Beschreibung der mittels des Tests gefundenen Schwächen der Algorithmen, die anhand der Dichte der Position von Unfällen über einzelnen Parametern der Strecke identifiziert werden konnten. Im dritten Schritt werden diese Ergebnisse mit den Ereignis-Häufigkeiten für zwei Streckenlayouts verglichen und plausibilisiert.

Ereignis-Häufigkeiten auf zufällig generierter Strecke

Die Ergebnisse des qualitativen Vergleichs der Algorithmen anhand der Szenarien auf zufällig generierten Streckenlayouts sind in Abbildung 8-1 dargestellt. Die Abbildung zeigt die Häufigkeiten für die jeweiligen Ereignisse über dem Parameter-Gitter. Ein Wert ist dabei immer das arithmetische Mittel über der gesamten Testanzahl für eine Parametrisierung.

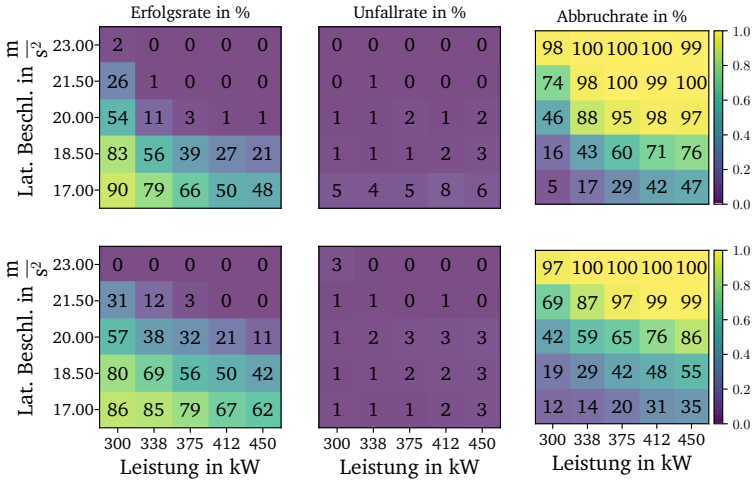


Abbildung 8-1.: Ereignisauswertung für 2000 Tests für Variationen der Parametrisierung für den Referenzalgorithmus (obere Zeile) und den LCADM-Algorithmus (untere Zeile). Variiert wurden die Parameter $a_{\text{lat,max}}$ und P_{max} mit $a_{\text{lat,max}} = D_{\text{max}}$. Die Diagramme zeigen von links nach rechts: 1) Prozentsatz an erfolgreichen Überholmanövern, 2) Prozentsatz an Kollisionen, 3) Prozentsatz an nicht erfolgreichen Überholmanövern.

In der Abbildung zeigt sich in der Abbruchrate, dass der LCADM-Algorithmus (untere Zeile) bei schnelleren Gegner-Fahrzeugen länger in der Lage ist, noch erfolgreich zu überholen. Dies ist darauf zurückzuführen, dass, wie bereits beschrieben, der Algorithmus in der Lage ist, beim Anbremsen auf die Kurve das gegnerische Fahrzeug

zu überholen. Dieser Unterschied der beiden Algorithmen begründet auch, dass der Referenzalgorithmus insgesamt bei niedrigen möglichen Querbeschleunigungen des gegnerischen Fahrzeugs öfter überholt als der LCADM-Algorithmus. Abbildung 8-2 zeigt die CDF der Überholposition für die beiden Algorithmen für $P_{\text{max}} = 300\,\text{kW}$, $a_{\text{lat,max}} = 17\,\frac{\text{m}}{\text{s}^2}$. Der LCADM-Algorithmus überholt insgesamt öfter vor der ersten Kurve bzw. in der ersten Kurve. Allerdings ist es dem Referenzalgorithmus aufgrund der sich stetig verändernden Randbedingungen möglich dem Gegner-Fahrzeug schneller zu folgen und ist dadurch in der Lage, in ca. $25\,\%$ der Fälle direkt hinter der Kurve zu überholen. Ein genauer Blick in die Daten zeigt dabei, dass durch die Wechsel zwischen den verschiedenen internen Entscheidungszuständen, der LCADM-Algorithmus Zeit verliert, wenn ein Überholmanöver nicht beim ersten Versuch erfolgreich ist. Allgemein ist die Performance der Trajektorienplanung anfällig für sich schnell ändernde Randbedingungen, insbesondere, wenn die Randbedingungen innerhalb eines kurzen Zeithorizontes nahe $n = 0$ liegen. Das begründet sich dadurch, dass die aktuelle

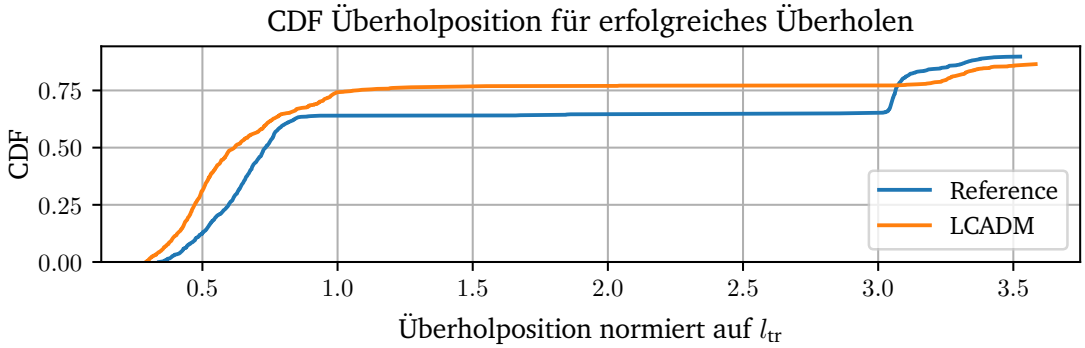


Abbildung 8-2.: CDF der Überholposition (auf Streckenlänge l_{tr} normiert) der Referenz und des LCADM-Algorithmus in zufälligem Szenario für $P_{\text{max}} = 300\,\text{kW}, a_{\text{lat,max}} = 17\,\frac{\text{m}}{\text{s}^2}, D_{\text{max}} = 17\,\frac{\text{m}}{\text{s}^2}$

Trajektorie immer auf der zuletzt berechneten Trajektorie des Ego-Fahrzeugs und des Gegner-Fahrzeugs basiert. Ändert sich eine der beiden Trajektorien signifikant, ist möglich, dass auch mehrerer Neuberechnungen notwendig sind, bis die Berechnung wieder einen stabilen Zustand erreicht. Darunter versteht sich in diesem Zusammenhang, dass bei einer Neuberechnung desselben Zeitschrittes sich die Positionen der einzelnen Zeitschritte nur in sehr kleinem Rahmen verschieben.

Unfalldichte und Überholdichte bei verschiedenen Parametrisierungen

Diese Schwäche des LCADM-Algorithmus, dass nicht erfolgreiche Überholversuche zum gesamten Scheitern des Manövers führen, zeigt sich besonders deutlich in Abbildung 8-3a mit dem Parametersatz $P_{\text{max}} = 375\,\text{kW}$, $a_{\text{lat,max}} = 18{,}5\,\frac{\text{m}}{\text{s}^2}$. Bei Szenarien mit einer Länge des ersten Segments S1 zwischen $500\,\text{m}$ und $650\,\text{m}$ ❶ liegen besonders viele Manöver, in denen der Algorithmus versucht, vor der ersten Kurve oder in der ersten Kurve zu überholen. Das Fahrzeug schafft es erfolgreich vor das Gegner-Fahrzeug zu

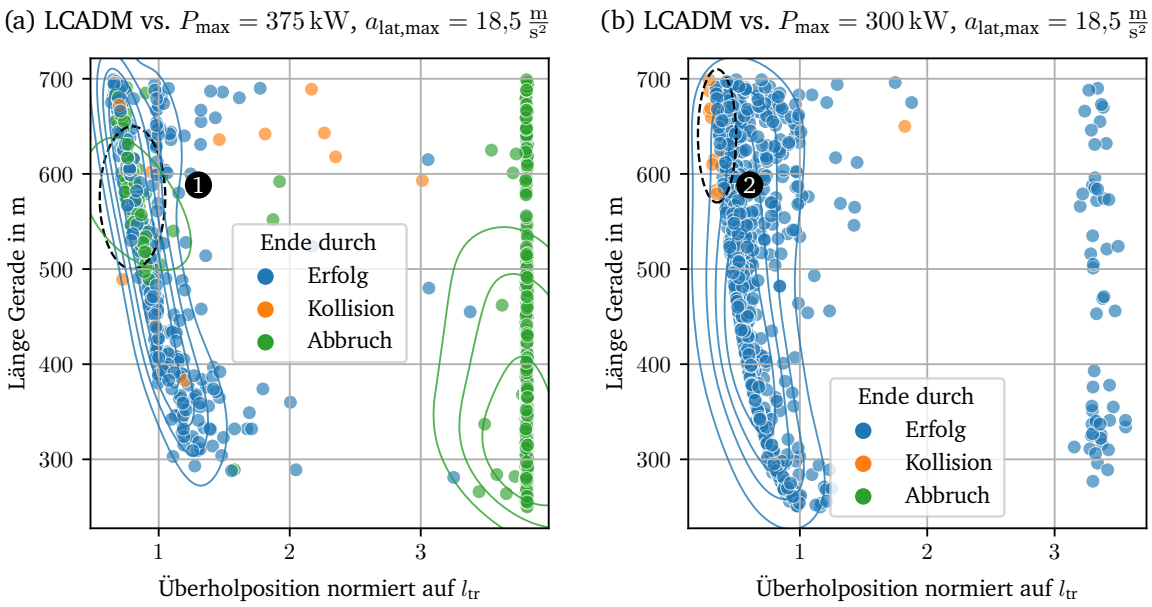


Abbildung 8-3.: (a) Ereignisauswertung LCADM-Algorithmus: Länge S1 über Überholposition (auf Streckenlänge l_{tr} normiert). Auswertung der Überholposition auch für nicht erfolgreiche Überholmanöver. In diesem Fall wird die letzte Position verwendet, an der sich das Ego-Fahrzeug vor dem gegnerischen Fahrzeug befunden hat.
(b) Ereignisauswertung LCADM-Algorithmus: Länge S1 über Überholposition (auf Streckenlänge l_{tr} normiert).
Die Linien geben die 2d-Kerndichteschätzung für jeden Szenarioausgang an.

fahren, ein erfolgreiches Überholmanöver gelingt trotzdem nicht. Die Daten zeigen, dass nahezu alle missglückten Überholmanöver in diesem Bereich auf der Kurvenaußenseite gestartet werden. Gehen beide Fahrzeuge in die Bremsphase für die erste Kurve über, hat das innen fahrende Fahrzeug einen Vorteil und das außen fahrende Fahrzeug verliert auf diese Weise Zeit und die gewonnene Position. Dieses Phänomen tritt unter der Zusatzbedingung auf, dass die Krümmung des ersten Segments S1 dasselbe Vorzeichen besitzt wie die erste Kurve. In diesem Fall liegt die Ideallinie vor der Kurve auf der Innenseite.

Die Beschränkung, dass die Überholmanöver nur für eine Länge von Segment S1 zwischen $500\,m$ und $650\,m$ scheitern, liegt an der speziellen Szenario-Entwicklung für diese Kombination der Parametrisierungen. Ist die Anlauflänge kürzer, erreicht das Ego-Fahrzeug das Gegner-Fahrzeug, wenn es bereits die Kurve auf der Außenseite anfährt. In diesem Fall bietet die Innenseite mehr Platz für ein Überholmanöver. Ist der Anlauf für das Ego-Fahrzeug länger, ist das Überholmanöver bereits vor der Kurve erfolgreich.

Ebenfalls verbunden mit der Auswahl der Seite zum Überholen des LCADM-Algorithmus, sind die Ergebnisse in Abbildung 8-3b. Sie zeigen für $P_{max} = 300\,kW$, $a_{lat,max} = 18{,}5\,\frac{m}{s^2}$ und damit einer sehr geringen longitudinalen Beschleunigung für das Gegner-Fahrzeug,

dass ein Teil der verursachten Kollisionen darauf zurückzuführen sind, dass, bevor ein Überholmanöver möglich ist, das hinten fahrende Fahrzeug mit dem vorne fahrenden Fahrzeug kollidiert ❷. Die Datenanalyse zeigt, dass hierfür ebenfalls die Seitenauswahl verantwortlich ist. Die Auswahl der Seite zum Überholen wird maßgeblich davon beeinflusst, auf welcher Seite die vorberechnete Trajektorie mehr Platz für das eigene Fahrzeug bietet. Diese vorberechnete Lösung ist sehr stabil für hohe Geschwindigkeiten, sodass auch von einem Zeitschritt zum nächsten die Position der Trajektorie für eine kurzen Zeithorizont sich nahezu nicht ändert. Anders verhält sich die Planung allerdings für niedrige Geschwindigkeiten und lange Geraden. Hier spielt in der reinen longitudinalen Beschleunigung bei einem Massenpunktmodell die genaue Richtung bzw. laterale Position auf der Strecke nur eine untergeordnete Rolle, da keine bremsenden Kräfte durch Schräglauf induziert werden und auch keine Kurve des Streckenlayouts eine eindeutige Seitenwahl bedingt. Damit ist die Pfadplanung nicht mehr eindeutig, da es keine Rolle spielt, ob das Fahrzeug in gerader Linie beschleunigt, ob es sich dabei hin und her bewegt oder ob es sich dabei am linken oder rechten Fahrbahnrand befindet. Die berechnete Lösung mit dem Modell ist damit über mehrere Berechnungen hinweg starken Veränderungen unterworfen. Das hat zur Folge, dass die Seitenwahl häufig die Seite wechselt und damit ein Ausweichen verhindert. Diese Schwäche wird damit durch die Parametrisierung des MPC-Algorithmus verursacht und tritt nur in dieser speziellen Konstellation auf. Ein Erhöhung der Kosten für die Änderung der Eingangsgrößen ist aber ebenfalls nicht praktikabel, da damit der Nach-

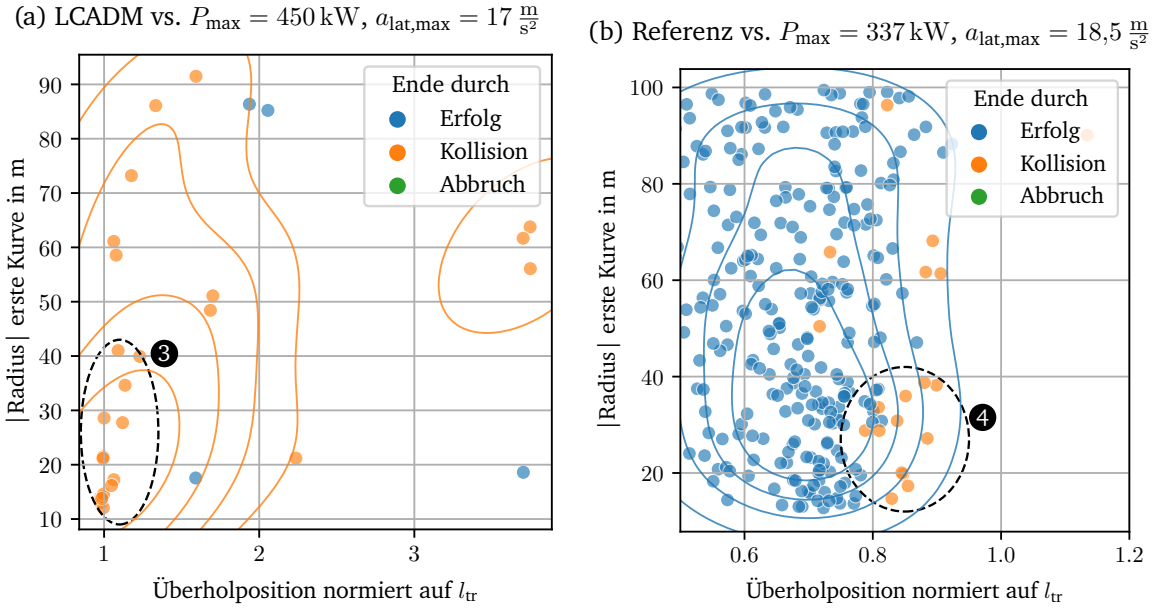


Abbildung 8-4.: Ereignisauswertung LCADM-Algorithmus: |Radius| erste Kurve über Überholposition (auf Streckenlänge l_{tr} normiert).
Die Linien geben die 2d-Kerndichteschätzung für jeden Szenarioausgang an.

teil einhergeht, dass die Agilität der Planung verloren geht. Ein mögliche Folge dessen ist, dass Kurven langsamer angefahren werden, damit die Querbeschleunigungen sich langsamer ändern.

Eine weitere Schwäche in Verbindung mit der Seitenwahl wird in Abbildung 8-4a ersichtlich. Die Abbildung zeigt eine Auswertung der Ereignisse für $P_{\text{max}} = 450\,\text{kW}$, $a_{\text{lat,max}} = 17\,\frac{\text{m}}{\text{s}^2}$. Auf der Ordinate ist hier der Betrag des Radius der ersten Kurve abgebildet.

Die Kombination aus Überlegenheit des Gegner-Fahrzeugs auf der Geraden und der Unterlegenheit in der Kurve sorgt dafür, dass es nur für sehr enge Kurven ❸ überhaupt möglich ist, dem Gegner nahezukommen. In diesem Fall erreicht das Ego-Fahrzeug genau das Gegner-Fahrzeug. Das Ego-Fahrzeug versucht zu überholen, ist aber nicht schnell genug, aber auch nicht so langsam, dass das Überholmanöver abgebrochen wird. Die Abbildung zeigt eine Abhängigkeit dieses Phänomens von dem Radius der ersten Kurve des Streckenlayouts bei der gegebenen Parametrisierung.

Eine Schwäche, die durch die Parametrisierung des MPC bedingt ist, wird in Abbildung 8-4b klar. Hier ist für den Referenzalgorithmus die Parametrisierung $P_{\text{max}} = 337\,\text{kW}$, $a_{\text{lat,max}} = 18{,}5\,\frac{\text{m}}{\text{s}^2}$ sichtbar, dass sich Kollisionen bei besonders geringen Kurvenradien der ersten Kurve ❹ häufen. Bedingt durch die Kombination aus zeitlicher Diskretisierung in Zeitschritte von $0{,}2\,\text{s}$ und eine Diskretisierung der Strecke in $1\,\text{m}$ Segmente wird hier für kleine Kurvenradien ein Aliasing-Effekt verursacht, sodass die Kurve mit zu hoher Geschwindigkeit angefahren wird. Ist das Gegner-Fahrzeug zu diesem Zeitpunkt noch nicht überholt, kommt es zur Kollision, da kein Platz für ein Ausweichmanöver zur Verfügung steht. Dasselbe Phänomen zeigt sich bei der Analyse des LCADM-Algorithmus für $P_{\text{max}} = 375\,\text{kW}$, $a_{\text{lat,max}} = 20\,\frac{\text{m}}{\text{s}^2}$.

Der Referenzalgorithmus zeigt eine Prinzip-bedingte Schwäche in Abbildung 8-5a. Obwohl die Randbedingungen des Algorithmus dazu bestimmt sind, eine Kollision des Fahrzeugs mit dem Gegner-Fahrzeug zu verhindern, kommt es bei der Parametrisierung $P_{\text{max}} = 337{,}5\,\text{kW}$, $a_{\text{lat,max}} = 23\,\frac{\text{m}}{\text{s}^2}$ am Kurveneingang häufig zu Kollisionen ❺. In den Daten wird ersichtlich, dass es immer dann dazu kommt, wenn die Fahrzeuge sich mit gleicher Geschwindigkeit nebeneinander dem Kurveneingang nähern und gleichzeitig sich das Ego-Fahrzeug auf der kurvenäußeren Position befindet. In diesem Fall kommt das gegnerische Fahrzeug dem Ego-Fahrzeug so nahe, dass die Randbedingung, die den Abstand zwischen beiden Fahrzeugen sicherstellt, außerhalb der Streckenbegrenzung gesetzt wird, dass das Optimierungsproblem nicht mehr unter Beachtung aller Randbedingungen lösbar ist. Dadurch, dass die Randbedingungen keine eindeutige Lösung mehr liefern, ist die Beachtung und Priorisierung der Randbedingungen gestört und die Randbedingung, die für die Verhinderung einer Kollision bestimmt ist, wird nicht mehr beachtet. Dadurch kollidieren beide Fahrzeuge. Dieses Verhalten zeigt sich

ebenfalls für $P_{\text{max}} = 412\,\text{kW}$, $a_{\text{lat,max}} = 18{,}5\,\frac{\text{m}}{\text{s}^2}$ in Abbildung 8-5b am Ausgang der zweiten Kurve K2 ❻.

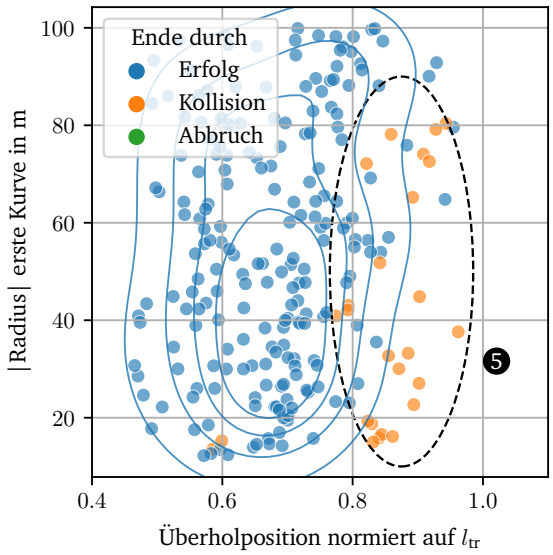
Ereignis-Häufigkeiten auf realen Streckenlayouts

Die Auswertung der Tests auf den realen Streckenlayouts der Grandprix-Strecken von Hockenheim und Barcelona-Catalunya ist in Abbildung 8-6a und Abbildung 8-6b dargestellt. Qualitativ zeigen sich ähnliche Ergebnisse wie bereits für das zufällige Szenario in Abbildung 8-1. Allerdings besonders in Bezug auf die Unfallraten beider Algorithmen zeigen sich einige Punkte, die im Folgenden hervorgehoben werden.

Für den Referenzalgorithmus zeigt sich eine auffällige Stelle mit einer besonders hohen Unfallrate von $17\,\%$ auf der Strecke Barcelona-Catalunya. Die Unfälle entsprechen dabei der Beobachtung aus Abbildung 8-5. In den Daten zeigt sich für diese Parametrisierung, dass sich die Unfälle zu $90\,\%$ auf eine einzelne Kurve konzentrieren. In dieser Kurve ergibt sich die Situation, dass die Randbedingungen des Algorithmus aus den Grenzen der Strecke gedrückt werden. Durch das unlösbare Optimierungsproblem werden die entsprechenden Randbedingungen durch den numerischen Solver ignoriert.

Ähnlich verhält es sich für den LCADM-Algorithmus. Da dieser mehr Quellen für mögliche Kollisionen aufweist, lassen sich nicht alle Unfälle bei den verschiedenen Parametrisierungen einer einzelnen spezifischen Fehlerquelle zuordnen. Die Analyse

(a) Referenz vs. $P_{\text{max}} = 337{,}5\,\text{kW}$, $a_{\text{lat,max}} = 23\,\frac{\text{m}}{\text{s}^2}$



(b) Referenz vs. $P_{\text{max}} = 412\,\text{kW}$, $a_{\text{lat,max}} = 18{,}5\,\frac{\text{m}}{\text{s}^2}$

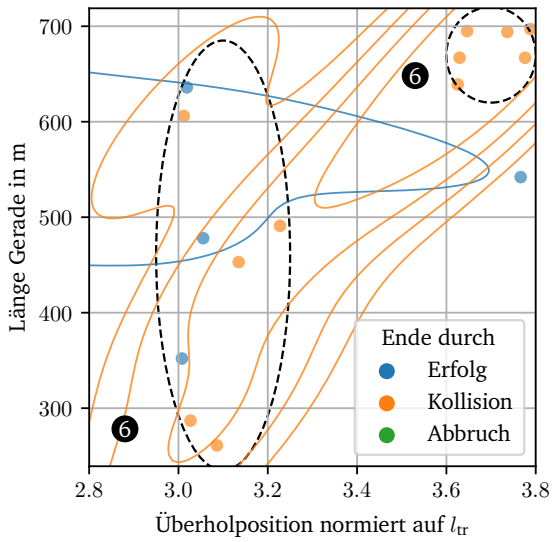


Abbildung 8-5.: (a) Ereignisauswertung Referenzalgorithmus: |Radius| der ersten Kurve über Überholposition (auf Streckenlänge l_{tr} normiert).
(b) Ereignisauswertung Referenzalgorithmus: Länge S1 über Überholposition (auf Streckenlänge l_{tr} normiert).
Die Linien geben die 2d-Kerndichteschätzung für jeden Szenarioausgang an.

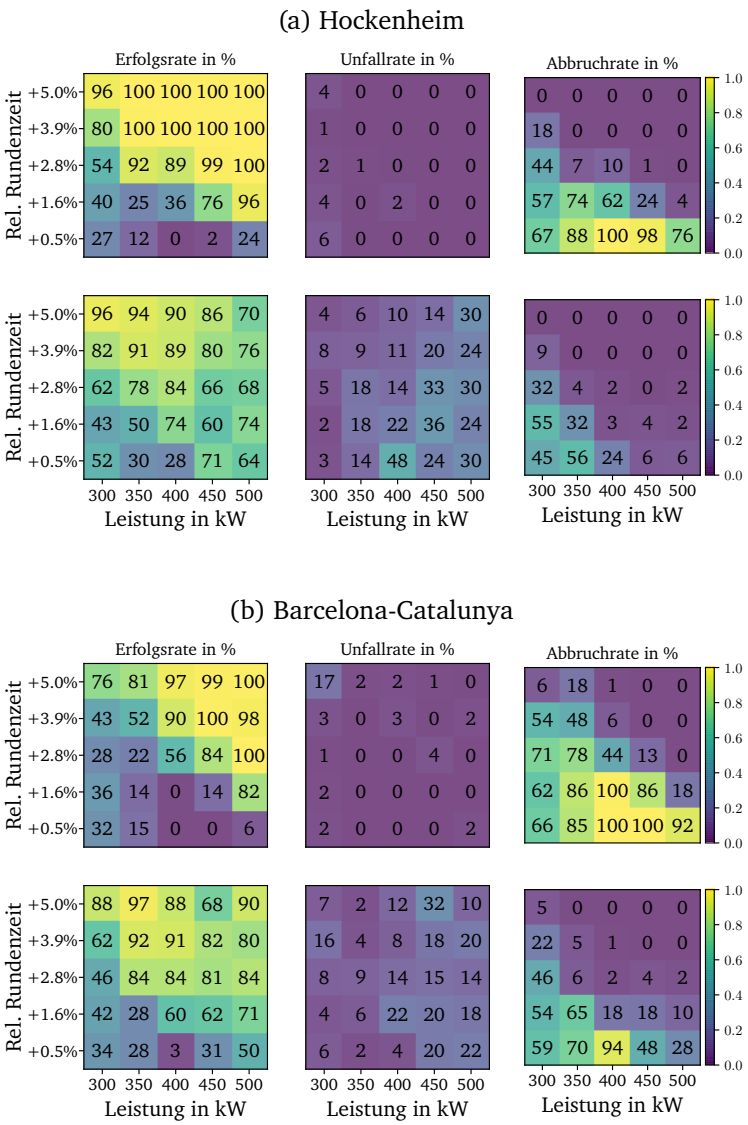


Abbildung 8-6.: Ereignisauswertung für 200 Tests für verschieden Parametervariationen für den Referenzalgorithmus (obere Zeile) und den LCADM-Algorithmus (untere Zeile). Variiert wurden die Parameter $P_{\max}$ und die relative Rundenzeit $\tau_{\text{lap,opp}}/\tau_{\text{lap,ego}}$ gemäß Abbildung 6-5. Die Diagramme zeigen von links nach rechts: 1) Prozentsatz an erfolgreichen Überholmanövern, 2) Prozentsatz an Kollisionen, 3) Prozentsatz an nicht erfolgreichen Überholmanövern. Die Linien geben die 2d-Kerndichteschätzung für jeden Szenarioausgang an.

der Daten zeigt allerdings keine Unfälle, die sich nicht mit einem der bereits aufgezählten Fehlerquellen erklären lässt. Das Problem von Aliasing-Effekten, die in sehr engen Kurven zu Kollisionen führen, trat aufgrund der verwendeten realen Strecken für beide getesteten Algorithmen nicht auf.

Der Test zeigt insbesondere den unterschiedlichen Einfluss der Fahralgorithmen auf die unterschiedlichen Arten zu testen. Während der LCADM-Algorithmus für die zufälligen Streckenlayouts mit hohen Überholraten bei schnellen Gegnern seine Stärke zeigt, verhält es sich bei dem Test auf dem realen Streckenlayout gegensätzlich, auch wenn

die Mechanismen, die die Kollisionen verursachen, identisch sind. Das spezielle Layout der Strecken in Kombination mit Gegner-Fahrzeugen, die auf der Geraden dem Ego-Fahrzeug überlegen sind, erschweren ein Überholen auf den Geraden erheblich. Durch die spezielle Aneinanderreihung von Kurven ergeben sich nur einzelne Kurven, in der der Algorithmus überhaupt noch die Gelegenheit zum Überholen besitzt. Diese Stelle fällte aber zur gleichen Zeit mit der Position zusammen, an der der Algorithmus für die Entscheidung auf einer Seite zum Überholen an seine Grenzen stößt. Diese Tatsache stützt die These, dass eine quantitative Aussage nur für eine bestimmte Parametrisierung und ein Streckenlayout möglich ist, da erst die Kombination beider Aspekte die Entwicklung einer Situation oder Kollision ausmacht.

8.1.3. Fazit

Die Tests mit unterschiedlichen Parametrisierungen auf zufälligen Strecken erlauben, die Stärken und Schwächen der einzelnen Algorithmen herauszukristallisieren. Die unterschiedlichen Parametrisierungen sorgen dafür, dass jeweils unterschiedliche Situationen aus den gleichen Randbedingungen heraus entstehen. Dadurch ist die Wahrscheinlichkeit gegenüber anderen Tests mit einem Gegner einen Fehler zu finden höher, da sich durch das zufällige Layout und die unterschiedlich parametrisierten Gegner eine Vielzahl von speziellen Situationen herausbilden. Die Anzahl der erzeugten Situationen und damit die Wahrscheinlichkeit, eine vorhandene Schwäche zu identifizieren, steigt mit der Anzahl an getesteten Varianten. Das zufällige Layout fungiert dabei als entscheidende Komponenten, da wie in den Tests gezeigt eine Abhängigkeit der Performance eines Algorithmus von der Form der Strecke besteht. Gleichzeitig steigert sich mit steigender Anzahl an Variationen auch der Rechenaufwand.

Die identifizierten Schwächen der Algorithmen sind vielfältiger Natur und sind zurückzuführen auf (a) Abhängigkeiten vom Streckenlayout, (b) die gewählte Parametrisierung des Optimierungsproblems, (c) prinzipbedingte Schwächen und (d) Inkompatibilitäten zwischen Algorithmen. Damit konnten in diesem Kontext nahezu alle Bereiche der Algorithmen als Möglichkeit für eine Schwachstelle identifiziert werden. Die Clusterung der Daten nach verschiedenen Parametern und das hohe Maß an Zufall im Design des Szenarios ermöglichen eine systematische Identifikation und Analyse von Schwachstellen in den Algorithmen. Damit ist die Methode für den Zweck der Entwicklung von Algorithmen und deren Parametrierung einsetzbar, da möglich ist, selbst kleine Parametereinflüsse und dadurch hervorgerufenen Schwächen mit der Methode zu detektieren.

Der Vergleich zwischen den Tests auf einer zufälligen Strecke mit den realen Layouts zeigt die Diskrepanz zwischen den beiden Testansätzen und bestätigt die ursprünglich festgelegte Vorgehensweise. Nicht alle Probleme, die beim Test mit einem zufällig generierten Streckenlayout auftreten, treten auch bei einem realen und festgelegten

Streckenlayout auf. Hingegen brachte die Analyse der Testergebnisse für die realen Streckenlayouts aber auch keine Erkenntnisse hinsichtlich weiterer Schwachstellen der Algorithmen. Die Wirksamkeit der in dieser Arbeit vorgestellten Methode hat sich damit durch die Anwendung bewährt.

Die Diskrepanz zwischen den Tests mit einem zufälligen Streckenlayout und denen auf den realen Layouts zeigt, dass die Möglichkeit, eine quantitative Aussage zu generieren, sich jeweils auf ein spezifisches Streckenlayout und eine Fahrzeug-Gegner-Kombination beschränkt. In jeder Kombination entstehen für verschiedene Parametrierungen unterschiedliche, aber spezifische Situationen. Eine allgemeine Aussage über die Überlegenheit eines speziellen Algorithmus lässt sich ohne die Anwendung der Ranking-Algorithmen daher in diesem Kontext nicht treffen, da es notwendig ist, auch die spezifischen Randbedingungen und die damit verbundene Komplexität eines Szenarios in dieser Aussage mit zu berücksichtigen.

8.2. Ranking der Algorithmen

Nach der qualitativen Evaluierung der Algorithmen wird in diesem Abschnitt durch Anwendung der vorgestellten Algorithmen mElo und über das maxent NG ein Ranking der Algorithmen erstellt. Die genaue Vorgehensweise zur Berechnung des maxent NG wurde von Balduzzi *et al*.[226] veröffentlicht und wurde von Ma[227] frei verfügbar implementiert. Die Algorithmen werden dabei nur für die Kategorie des Überholens beurteilt. Zusätzlich dazu wird jeweils betrachtet, welcher Algorithmus am häufigsten Kollisionen verursacht. Als Testobjekte werden die beiden bereits vorgestellten deterministischen Algorithmen als Überhol-Algorithmen genutzt. Als dynamische Objekte wird zusätzlich dazu auch das trainierte NN verwendet. Die Testbedingungen sind dabei identisch zu denen der Streckentests im vorigen Abschnitt. Der wesentliche Unterschied ist, dass jeder der Überhol-Algorithmen jeweils gegen alle anderen Verteidigungs-Algorithmen getestet wird. Diese wiederum nehmen, wie auch schon im Abschnitt zuvor nacheinander alle Parametrisierung aus dem definierten Parameter-Gitter an. Ein Test mit einer Parametrisierung besteht erneut aus 200 Einzeltests mit jeweils zufälligen Startpunkten verteilt über der Länge der Strecke.

Aufgrund der Erkenntnis, dass eine Aussage nur jeweils für ein bestimmtes Streckenlayout Gültigkeit besitzt, wird nur die Barcelona-Cataluny-Grand-Prix-Strecke betrachtet. Die Ergebnisse für die Hockenheim-Grand-Prix-Strecke und auch für das Verteidiger-Szenario finden sich in Anhang D.4.

Die Ergebnisse der Tests sind in Tabelle 8-1 dargestellt. Jeweils ein zu testender Algorithmus bildet eine Zeile für die drei Fälle (1) der diskreten Metrik (Häufigkeit des

[226] Balduzzi, D. et al.: Re-evaluating Evaluation (2018).

[227] Ma, J.: pybrium: Strategy and equilibria toolkit (2019).

Tabelle 8-1.: Payoff-Matrix und resultierendes Ranking für Algorithmen auf dem Barcelona-Catalunya Streckenlayout. Algorithmen links versuchen oben stehende Algorithmen zu überholen.

	NN	LCADM	Referenz	mElo	NG
LCADM	0.491	0.713	0.681	1624(2)	0.4399(2)
Referenz	0.802	0.529	0.826	1854(1)	0.4399(1)
NN	0.380	0.164	0.465	1331(3)	0.1202(3)
Diskreter Reward					
LCADM	0.342	0.571	0.474	1457(1)	0.3922(1)
Referenz	0.513	0.345	0.478	1434(2)	0.3364(2)
NN	0.402	0.407	0.330	1368(3)	0.2714(3)
Kontinuierlicher Reward					
LCADM	0.963	0.891	0.991	2049(2)	0.3333(1)
Referenz	0.984	0.992	1	2332(1)	0.3333(2)
NN	0.710	0.386	0.957	1689(3)	0.3333(3)
Unfälle					

Erfolgs für den Test), (2) der kontinuierlichen Metrik (Distanz bis zum Ende/Abbruch des Tests) und (3) der Anzahl an Kollisionen. Für eine einfachere Übersicht über die Ergebnisse wurden alle Tests gegen einen bestimmten Gegner-Algorithmus in einer Spalte über das arithmetische Mittel zusammengefasst. Die hinteren Spalten zeigen die errechnete Punktzahl für das jeweilige Ranking-Verfahren mit der Platzierung in Klammern.

Das erzeugte Ranking unterstreicht noch einmal die Aussage, dass die Relationen zwischen den Algorithmen intransitiv und daher nicht extrapolierbar sind. Während der trainierte NN Agent von den getesteten Varianten als dynamisches Objekt das effektivste ist, um den LCADM-Algorithmus abzuwehren, erweist sich das NN am wenigsten effektiv gegen den Referenzalgorithmus. Der Referenzalgorithmus ist bedingt durch seine Funktionsweise darauf angewiesen, dass das vorne fahrende Fahrzeug von der Ideallinie abweicht. Nur dann konvergiert die Lösung des Referenzalgorithmus schnell hin zu einem Überholmanöver. Durch die Zufälligkeit, die der NN-Agent durch sein Funktionsprinzip besitzt, erleichtert er daher das Überholen für den Referenzalgorithmus. Für den LCADM-Algorithmus verhält es sich genau gegensätzlich. Der LCADM-Algorithmus wird durch die zufälligen kleinen Bewegungen und die damit verbundenen internen Seitenwechsel gebremst.

Das Überholen durch den NN-Agenten ist allerdings unter den gegebenen Trainings-/Testbedingungen noch nicht optimal. Der Algorithmus ist in der Lage, gegen den verteidigenden NN-Agenten einen höheren kontinuierlichen Reward zu erzielen, als der LCADM-Algorithmus. Gegen den LCADM-Algorithmus erreicht er ebenfalls den zweithöchsten mittleren kontinuierlichen Reward.

Der Vergleich der Ergebnisse von mElo mit dem maxent NG zeigt, das beide Verfahren nur jeweils ein relatives Rating bieten. Die errechneten Ränge sind zwar mit Ausnahme der untersten Zeile identisch. Die Verteilung des maxent NG zeigt aber in den meisten Fällen nur einen minimalen Unterschied zwischen den Strategien an.

Die Problematik des Elo-Systems, dass die Menge der Punkte, die sich im System befindet, immer konstant bleibt, wird durch die Verwendung von mElo nicht gelöst, da das mElo Verfahren ebenfalls die Punkte, die einem Spieler zugeschrieben, immer einem anderen Spieler abgezogen werden. Das berechnete Ranking von mElo ist daher ebenfalls nur relativ zwischen den einzelnen Agenten. mElo bietet allerdings die Möglichkeit, die relativ konvergierende Skala auf einen oder mehrere definierte Tests mit einer bestimmten Parametrisierung zu normieren und auf diese Weise die Vergleichbarkeit zu erhöhen. Daneben besteht weiter die Möglichkeit einer Adaption der Ranking-Systeme von Glicko oder TrueSkill auf intransitive Relationen. Auf diese Weise ließen sich fixe Punkte innerhalb des Rankings bilden, um die Vergleichbarkeit zwischen einzelnen Durchführungen des Ranking-Algorithmus zu erhöhen.

Der Vergleich der kontinuierlichen mit der diskreten Metrik zeigt, dass selbst die Auswahl der gewählten Metrik einen Einfluss auf das Testergebnis nimmt. Während der Referenzalgorithmus bei der Anzahl der erfolgreichen Überholmanöver das beste Ergebnis erreicht, schneidet der LCADM-Algorithmus bei der kontinuierlichen Metrik besser ab. Grund dafür ist, dass der Referenzalgorithmus zwar insgesamt es häufiger schafft, erfolgreich zu überholen, der LCADM-Algorithmus dagegen, wenn er überholt, der Überholvorgang schnell zu Ende führt und der Algorithmus daher nur wenig Zeit verliert. Die verwendete Metrik bestimmt damit auch, ob Fahrweisen bevorzugt werden, die die Sicherheit in den Vordergrund stellen oder Fahrweisen, die auf einen schnellen Überholvorgang hin optimiert wurden.

Die Intransitivität der Relationen wirft ebenfalls die Frage nach der Aussagekraft eines Rankings im Allgemeinen auf. Während das Verfahren zwar keine Änderung der Rangfolge beim Hinzufügen von redundanten Tests zeigt, wird die Rangfolge wesentlich beeinflusst durch die Auswahl der Testobjekte. Eine Entfernung des LCADM-Algorithmus als Gegner-Objekt hätte an dieser Stelle bspw. zur Folge, dass für die kontinuierliche Metrik ebenfalls der Referenzalgorithmus mit Rang 1 bewertet wird. Die Auswertung der Ergebnisse im vorigen Kapitel zeigt, dass die Auswahl des Parameter-Grids ebenfalls in der Lage ist das Ergebnis zu beeinflussen, da sich viele Fehlerbilder nur in wenigen Parameter-/Strecken-Kombinationen zeigen. Ebenso verhält es sich mit der Begrenzung der Länge der Szenarien, die ebenfalls je nach der Paarung der Testobjekte das Ergebnis bzw. die Werte in der Payoff-Matrix ändern.

Die Antwort von Forschungsfrage 4 und damit der Frage nach dem Einfluss der Intransitivität auf eine Rangfolge lässt sich wie folgt zusammenfassen. Im Falle von

intransitiven Relationen sinkt die allgemeine Aussagekraft eines Rankings mit steigender Anzahl an Tests, da der Informationsgehalt der Payoff-Matrix steigt. Gleichzeitig steigt aber die Aussagekraft der Payoff-Matrix. Ein Ranking, dass die Güte eines Verhaltensalgorithmus auf eine einzelne Kennzahl reduziert, wird aus den genannten Gründen nur für den Fall, dass ein genau definierter Anwendungsfall vorliegt, als Möglichkeit für die Bewertung mehrerer Algorithmen betrachtet. Andernfalls entfernt die alleinige Betrachtung des Rankings die zusätzliche Dimension, die das mElo-Verfahren und eine Szenario-Auswahl mit breit gestreuten Parametervariationen mitbringen.

9. Fazit und Ausblick

9.1. Fazit

Der Vergleich von mehreren Verhaltensalgorithmen stellt eine komplexe Aufgabe dar. Mit steigender Komplexität und Interaktion zwischen Agenten in einem Szenario des autonomen Fahrens steigt auch die Komplexität für die Bewertung der Güte eines einzelnen Agenten. Der Usecase im Rahmen von autonomem Motorsport erschwert die Bewertung zusätzlich. Bis auf einige sicherheitstechnische Regeln, die der Vermeidung von Kollisionen dienen, existieren kaum Richtlinien als Maßstab für eine Bewertung. In Kombination mit der Anwendung von spieltheoretischen Ansätzen für den Vergleich mehrerer Agenten ist daher lediglich der Einsatz von absoluten und klar skalierten Metriken für den Erfolg möglich. Dies resultiert darin, dass für den Vergleich von Agenten nur die einfachsten Metriken genutzt werden wie der binäre Erfolg in einem Szenario oder die Ausnutzung einer Zeit- oder Streckenvorgabe für die Ausführung eines Manövers.

Die qualitative Auswertung der Algorithmen in dieser Arbeit zeigt, dass Verhaltensalgorithmen durch die Tatsache, dass sie einander in der Planung berücksichtigen, sich gegenseitig stark beeinflussen. Damit begründen sich auch die Singularitäten in der Bewertung der Algorithmen, die ein Resultat der Kombination verschiedener Randbedingungen und der Parametrisierung der Algorithmen sind. Diese Phänomene sind ohne tatsächliche Tests nicht vorherzusagen, da ein Szenario, das mit identischen Startbedingungen beginnt, in Abhängigkeit der Ausprägung der beteiligten Objekte unterschiedlichste Resultate liefert. Dadurch wird die Verwendung von mehrdimensionalen Vergleichstechniken wie mElo und maxent NG notwendig, um eine Rangfolge zu bilden. Das in dieser Arbeit vorgestellte und abstrahierte Szenario zeigt sich effektiv für die systematische Analyse von Interaktionen. Die Tatsache, dass ein Überholmanöver nur auf einer Geraden bzw. in/vor/hinter einer Kurve möglich ist, erlaubt die Abstraktion von verschiedensten Streckenlayouts in das vorgestellte, zufällig parametrisierte Layout. Der einfache Aufbau des Layouts erlaubt eine Auswertung, die die einzelnen Streckensegmente als Normierung verwendet. Damit wird die Vielfältigkeit an Kurvenkombinationen, die durch das zufällig parametrisierte Layout abgebildet wird, vergleichbar. Der hohe Abstraktionsgrad in Kombination mit der Normierung erlaubt eine einfache Identifikation von Stärken und Schwachstellen und eignet sich daher für den Einsatz als Entwicklungstool für die Entwicklung und Weiterentwicklung von Verhaltensalgorithmen. Die Tatsache, dass die gefundenen Schwachstellen bei Testung auf realen Streckenlayouts eine Untermenge der gefundenen Schwachstellen für den Test auf dem vorgestellten zufällig parametrierten Streckenlayout darstellen, bestätigt das in dieser Arbeit präsentierte Vorgehen. Je höher der Grad des

Zufalls in den durchgeführten Tests ist, desto wahrscheinlich ist es Schwachstellen, in den Verhaltensalgorithmen zu finden. Beim Test auf festen Streckenlayouts werden Schwachstellen stark polarisiert dargestellt und es ist mitunter nur eingeschränkt möglich sie zu identifizieren.

Die Analyse der quantitativen Ergebnisse in dieser Arbeit zeigt, dass der Vergleich mehrerer Agenten durch die fehlende Annahme der Transitivität erschwert wird. Dadurch ist notwendig, für einen Vergleich von Algorithmen anders als für den Vergleich von Menschen eine voll besetzte Payoff-Matrix anhand von Tests zu ermitteln. Für den Vergleich Algorithmus gegen Algorithmus steigt damit der Testaufwand quadratisch mit der Anzahl an Teilnehmern. Alternativ dazu ist möglich, ein definierten Satz an Tests mit bekannten Testbedingungen und allgemein zugänglichen Referenzobjekten zu entwickeln, die eine allgemeine Aussage ohne die Einschränkung eines einzelnen konkreten Szenarios ermöglichen.

Die in dieser Arbeit vorgestellte Methode für die Approximation fehlender Elemente in der Payoff-Matrix gibt zwar einen ersten Ansatz für eine Schätzung einzelner Testergebnisse. Dem Ansatz fehlt allerdings (a) die notwendige Robustheit, um Ergebnisse zuverlässig zu schätzen und (b) existiert keine Garantie, dass die errechneten Ergebnisse mit dem realen Erwartungswert korrelieren. Dies ist bedingt durch die komplexen Interaktionen der Agenten und der damit verbundenen intransitiven Relationen untereinander. Damit ist nicht reproduzierbar möglich, sicherzustellen, dass auch für ein perfektes Verfahren eine fehlende Relation korrekt approximiert wird, da durch die Annahme der Intransitivität unbekannt ist, ob die notwendigen Information für die Schätzung eines Eintrags tatsächlich vorhanden sind.

Die möglichst allgemein gehaltene Bewertung der Algorithmen sorgt dafür, dass ein etwaiger Vorteil von einzelnen Algorithmen minimiert wird. Dabei ist insbesondere bei der Auswahl der verfügbaren Algorithmen für Testobjekte wichtig, dass eine möglichst große Vielfalt an Funktionsweisen abgebildet wird, die auch eine hohe Übertragbarkeit auf einen realen Pool an Algorithmen besitzt. Der Vergleich der Performance von verschiedenen Algorithmen gegeneinander begründet diese Notwendigkeit, da die Performance eines Algorithmen-Paars durch komplexe Interaktionen nicht vorhersagbar ist. Ebenso zeigt der Test einzelner Algorithmen auf verschiedenen Streckenlayouts den hohen Einfluss, den die Strecke auf eine Performance-Metrik besitzt. Die Entwicklung einer definierten Benchmark-Strecke oder eines definierten Benchmark-Verhaltens-Algorithmus führt daher nicht zu dem Ziel, eine allgemeingültige Aussage über die Performance eines einzelnen Algorithmus zu generieren. Gleichzeitig steigt mit der notwendigen Vielfalt aber auch die Anzahl an Informationen, die in einer Zahl zusammengefasst werden. Die Anwendung von Ranking-Methoden aus der Spieltheorie wie mElo ist in der Lage, die intransitiven Relationen der Algorithmen abzubilden. Dem entgegen steht die Tatsache, dass ein einzelner Rang an Aussagekraft verliert, je

vielfältiger die Relationen sind, die damit abgebildet werden. Eine allgemeingültige Aussage über die Überlegenheit eines Algorithmus gegenüber einem anderen ist daher nur für einen klar identifizierten Anwendungsfall möglich und auch nur da gültig.

Mit der Forderung nach vielfältigen Vergleichsalgorithmen wächst ebenso die Notwendigkeit nach applizierbaren Standards für die Planung, die von jedem Algorithmus für einen Test anwendbar sind. Das in dieser Arbeit präsentierte Framework stellt dafür einen ersten Schritt dar, da Tests unabhängig vom Bewegungsmodell und Regelalgorithmus durchführbar sind.

Der mit $300\,\mathrm{Hz}$ rechnende vorgestellte Trajektorienplaner erlaubt erstmals einen hybriden Ansatz aus RL und einer MPC-Trajektorienplanung, der für die Aufgabe des Verhinderns von überholenden Fahrzeugen erste Erfolge zeigt. Für die Aufgabe, ein anderes Fahrzeug zu überholen, erzielt der vorgestellte Algorithmus gegen zwei von drei Algorithmen die zweitbesten Ergebnisse. Es ist anzumerken, dass für eine präzisere Definition der Parameter eines Szenarios bessere Ergebnisse erwartet werden, da für einen lernenden Algorithmus eine zu breite Vielfalt an Szenarien auch das Risiko birgt, dass eine Lernen von spezifischen Situationen erschwert wird. Im Vergleich zu anderen bereits bekannten und auf ML basierten Ansätzen ermöglicht die Kombination mit einem MPC-Trajektorienplaner eine Applikation des Verfahrens auf allgemeinen Streckenlayouts, ohne das erneute Anlernen des Algorithmus zu erfordern. Gleichzeitig ermöglicht der Einsatz des MPC den Einsatz sehr einfacher Reward-Funktionen. Es ist auf diese Weise möglich, den Lernprozess auf die Interaktion der Fahrzeuge zu beschränken, ohne die Notwendigkeit, die Fahrzeugführung im gleichen Optimierungsproblem zu lösen.

Der vorgestellte LCADM-Algorithmus bietet die Möglichkeit für aggressive Überholmanöver, die einem gegnerischen Fahrzeug aber trotzdem Raum zum weiteren Manövrieren lassen. Die bislang noch höhere Unfallrate ist dabei auf wenige Singularitäten zurückzuführen. Das Konzept als solches zeigt sich in der Analyse allerdings erfolgreich. Der vorgestellte Testansatz bietet die Möglichkeit, durch den systematischen Test nicht nur die Schwachstellen eines Algorithmus zu finden. Durch die leicht automatisierbaren Tests ist ebenfalls ein Hyperparameter-Tuning leicht möglich, da das Ranking zwischen sehr ähnlichen Algorithmen und innerhalb eines identischen Szenarios einen Anhaltspunkt für deren relative Güte bietet. Dieses Ranking in einzelnen definierten Szenarien kann ebenfalls als einfache Metrik betrachtet werden, die für den Einsatz von ML benötigt[228] werden, aber bislang nur schwer definierbar sind.

Die Verifikation der Methode in der Rennsimulation Projekt Cars 2 zeigt die Eignung des Ansatzes, die einfache Modellierung des Massenpunkts für die hochdynamische Umgebung eines Rennens zu nutzen. Der dafür entworfene Deichsel-Regler für die

[228]Vgl. Maurer, M. et al.: Autonomes Fahren (2015), S. 468.

laterale Regelung des Fahrzeugs zeigt dabei eine hohe Stabilität bei minimalem Applikationsaufwand.

9.2. Ausblick

Einige Aspekte des autonomen Rennenfahrens bleiben in dieser Arbeit allerdings noch unbehandelt und sind ein möglicher Ansatzpunkte für nachfolgende Forschung. Die Strategie eines jeden Rennteilnehmers wurde in dieser Arbeit ausgeklammert, da dafür die Betrachtung von Rennen in voller Länge mit Modellierungen notwendig ist, die auch die Degradation des Fahrzeuges mit berücksichtigen. Die daraus entstehenden Implikationen für einzelne Szenarien und die beteiligten Agenten können in der Entwicklung von fortschrittlichen Fahralgorithmen ebenfalls in Zukunft eine Rolle spielen. Mit steigender verfügbarer Rechenleistung lassen sich auf diesem Gebiet mehr und zuverlässigere Erkenntnisse erwarten.

Die Aussagekraft bezogen auf ein gesamtes Rennen ist ebenfalls ein Punkt für zukünftige Forschung, da aufgrund der hohen Komplexität die vorgestellte Methode keine Kennzahl oder Einordnung bietet, die eine Aussage über den Erfolg eines Agenten in einem ganzen Rennen ermöglicht. Diesbezüglich ist nicht nur die Extrapolation der in dieser Arbeit gewonnenen Erkenntnisse eine Herausforderung. Vielmehr steigt auch die Anzahl an Interaktionen und demnach die Anzahl an möglichen Inkompatibilitäten zwischen verschiedenen Algorithmen an.

Die notwendige Vielfalt verfügbarer dynamischer Vergleichsobjekte für einen Test von Verhaltensalgorithmen ist zum Zeitpunkt der Entstehung dieser Arbeit ebenfalls noch nicht gegeben, um eine solide Aussage über die Güte eines Algorithmus und dessen Fähigkeiten bezüglich der Interaktion mit anderen Algorithmen zu ermöglichen. Es ist möglich, dass die Entwicklung von Agenten speziell für die Evaluation weiteren Fortschritt bringt, da insbesondere die Evaluation der Fähigkeiten für die Verteidigung eine hoch entwickelte Interaktion zwischen den Agenten voraussetzt. Denn eine umfassende Aussage bezüglich der Fähigkeiten hoch entwickelter Verhaltensalgorithmen ist nur durch hoch entwickelte Testobjekte möglich.

Die Erkenntnis, dass das Bilden einer Rangfolge nur für ein spezielles Szenario Aussagekraft besitzt, bietet ebenfalls Potenzial für weitere Arbeiten. Da der autonome Motorsport als Showcase und Werbung für die Entwicklung des autonomen Fahrens betrachtet werden kann, ist aber nicht nur die Aussagekraft des technologischen Vergleichs von Interesse, sondern auch der Unterhaltungswert für die Zuschauer.

Diese Arbeit lässt ebenfalls offen, welche Auswirkungen die Diskrepanz zwischen prädiziertem Verhalten und realem Verhalten besitzt. Dies ist von Interesse, wenn die Aktionen eines gegnerischen Algorithmus nicht bekannt sind, das Ego-Fahrzeug aber trotzdem auf eine Prädiktion der anderen Objekte angewiesen ist, um die eigene

Trajektorie zu planen. Dazu gehört auch die Betrachtung und die Ausnutzung von Fahrfehlern eines anderen Algorithmus, die in dieser Arbeit ebenfalls nicht berücksichtigt wurden. Dabei bieten die Fahrfehler der anderen Fahrzeuge in realen Rennen eine der effektivsten Möglichkeiten für ein Überholmanöver.

Die qualitative Untersuchung der Algorithmen und die Probleme bei der Interaktion zeigen aber auch die Implikationen dieser Arbeit für autonome Fahrzeuge im normalen Straßenverkehr. Insbesondere der Begriff der Kolonnenstabilität, der von Winner[229] und Maurer[230] für einfache Assistenzsysteme wie der Adaptive Cruise Control beschrieben wurde, ist eine wahrscheinliche Richtung für zukünftige Forschungen. Die Herausforderung der mehrdimensionalen Planung und Prädiktion im Umfeld von anderen Fahrzeugen wächst mit der Anzahl an Fahrzeugen. Diese Arbeit zeigt die Möglichkeit auf, dass die Kenntnis der Trajektorie des anderen Fahrzeugs ebenso für Probleme in der Planung sorgt, wie die Unkenntnis darüber. Der Einsatz von spieltheoretischen Ansätzen wie die α^{α}-Rank-Methode[231] gibt erste Erkenntnisse bezüglich des Einsatzes von verschiedenen Strategien im Straßenverkehr. Sie erlaubt aber bislang keine Aussage über den Einsatz voll entwickelter Agenten oder über das Zusammenspiel von menschlichen Fahrern und autonomen Fahrzeugen.

Die Übertragung der in dieser Arbeit vorgestellten Methode auf weitere Anwendungsbereiche ist abschließend ein Ansatzpunkt für kommende Forschungen. Je vielfältiger die Anwendungsszenarien für ein autonomes System sind, desto wichtiger ist eine Vergleichsmethode, die in der Lage ist, sowohl in der Entwicklung als auch für fertige Systeme die Güte festzustellen und dabei auch nach den jeweiligen Aufgabengebieten eines Systems zu unterscheiden. Dabei ist nicht nur der relative Vergleich zwischen verschiedenen Konzepten hilfreich für System-Entwickler, auch eine genaue Analyse der Randbedingungen gewinnt mit einer steigenden Anzahl an Aufgabenbereichen eines Systems an Bedeutung. An dieser Stelle sind aussagekräftige Benchmark-Metriken notwendig, um einen Vergleich zu ermöglichen, der nicht durch eine Optimierung auf einzelne Szenarien, die den Fähigkeiten eines bestimmten Systems entsprechen, geschönt ist. Die Verantwortung liegt an der Stelle, bei dem Ersteller der Testszenarien dafür zu sorgen, dass eine Optimierung auf ein definiertes Testset einen möglichst großen Vorteil für die tatsächlich zu lösenden Aufgaben bringt. Die möglichen Anwendungsbereiche reichen dabei von autonomen Flugzeugen über Transportroboter in Fabriken bis hin zu verschiedenen Anwendungen im Sicherheits- und Assistenzbereich zur Unterstützung von Menschen.

[229] Winner, H. et al.: Handbuch Fahrerassistenzsysteme (2015), S. 872ff.

[230] Maurer, M. et al.: Autonomes Fahren (2015), S. 317.

[231] Vgl. Yang, Y. et al.: α^{α}-Rank: Practically Scaling α-Rank through Stochastic Optimisation (2020).

A. Optimierungsproblem

Das Optimierungsproblem wird über folgenden Zusammenhänge vollständig beschrieben:

$$\begin{aligned}
\text{minimize} \quad & -{}_F x_N + \sum_{n=0}^{N} (\xi_{\text{l,x}} \boldsymbol{x}_n + \boldsymbol{x}_n^T \xi_{\text{q,x}} \boldsymbol{x}_n \\
& + \xi_{\text{l,u}} \boldsymbol{u}_n + \boldsymbol{u}_n^T \xi_{\text{q,u}} \boldsymbol{u}_n + \xi_{\text{l,s}} \mathfrak{s}_n + \mathfrak{s}_n^T \xi_{\text{q,s}} \mathfrak{s}_n) && \text{(A-1)} \\
\text{subject to} \quad & \boldsymbol{x}_{n+1} = \boldsymbol{A}_n \boldsymbol{x}_n + \boldsymbol{B}_n \boldsymbol{u}_n + \boldsymbol{b}_n && \text{(A-2)} \\
& \boldsymbol{x}_0 = \boldsymbol{b}_0 && \text{(A-3)} \\
& \boldsymbol{x}_{l,n} \leq \boldsymbol{C}_n \boldsymbol{x}_n + \mathfrak{s}_n \boldsymbol{x}_n \leq \boldsymbol{x}_{u,n} && \text{(A-4)} \\
& \boldsymbol{C}_n = \begin{bmatrix} \boldsymbol{c}_{n,\text{tr}} \\ \boldsymbol{c}_{n,\text{acc},1} \\ \cdots \\ \boldsymbol{c}_{n,\text{acc},k_{\text{ac}}} \\ \boldsymbol{c}_{n,P} \\ (\boldsymbol{c}_{n,\text{opp}}) \end{bmatrix} && \text{(A-5)} \\
& \begin{bmatrix} {}_E v_{x,N} \\ {}_E v_{y,N} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} && \text{(A-6)} \\
\text{with} \quad & n \in [0, N] && \text{(A-7)}
\end{aligned}$$

Die Kostenfunktion besteht dabei (1) aus der Endbedingung, damit das Fahrzeug soviel Strecke wie möglich innerhalb des Zeithorizontes zurücklegt, (2) den linearen Kostentermen ξ_{l} und quadratischen ξ_{q} Kostenterme für die Eingangsgrößen ξ_{u}, die Zustandsgrößen ξ_{x} und die Slack-Variablen der Randbedingungen ξ_{s}.

Dabei werden folgende Parameter verwendet:

Tabelle A-1.: Parameter Optimierungsproblem

Variable	**Beschreibung**	**Wert**
A	Projizierte Stirnfläche	$2\,\text{m}^2$
$\frac{c_{w,\text{red}}}{c_w}$	Max. Reduzierung Luftwiderstand	$20\,\%$
c_w	Luftwiderstands-Beiwert	$0{,}3$
$d_{\text{SZ,co}}$	Längsseitiger Spielraum für konstante Sicherheitszone	$8\,\text{m}$
$d_{\text{SZ,po}}$	Längsseitiger Spielraum für wachsende Sicherheitszone	$20\,\text{m}$
$\mathfrak{d}t$	Länge eines Zeitschrittes	$0{,}2\,\text{s}$

Auf nächster Seite fortgesetzt

Tabelle A-1 – *Fortsetzung von letzter Seite*

Variable	**Beschreibung**	**Wert**
$\frac{F_{z,\text{red}}}{F_z}$	Max. Reduzierung Radaufstandskraft	$10\,\%$
k_{ac}	Anzahl der Beschleunigungs-Randbedingungen	9
l_{veh}	Fahrzeuglänge	$5\,\text{m}$
m	Fahrzeugmasse	$1500\,\text{kg}$
μ	Reibkoeffizient zwischen Reifen und Fahrbahn	$1{,}0$
N	Anzahl berechneter Zeitschritte	50
$\xi_{\text{l},\mathcal{J}}$	Linearer Kostenterm für Ruck	$0{,}000\,001$
$\xi_{\text{q},\mathcal{J}}$	Quadratischer Kostenterm für Ruck	$0{,}001$
$\xi_{\text{l,acc}}$	Linearer Slack-Anteil für Beschl.-Randbedingung	1
$\xi_{\text{l,SC}}$	Linearer Slack-Anteil für andere räumliche Randbedingung	30
$\xi_{\text{l,tr}}$	Linearer Slack-Anteil für Strecken-Randbedingung	40
$\xi_{\text{q,acc}}$	Quadratischer Slack-Anteil für Beschl.-Randbedingung	10
$\xi_{\text{q,SC}}$	Quadratischer Slack-Anteil für andere räumliche Randbedingung	50
$\xi_{\text{q,tr}}$	Quadratischer Slack-Anteil für Strecken-Randbedingung	100
$\tau_{\text{OT,max}}$	Max. Einflusszeit fürs Überholen	$2{,}5\,\text{s}$
$\tau_{\text{OT,min}}$	Min. Einflusszeit fürs Überholen	$1{,}0\,\text{s}$
$\tau_{\text{PD,max}}$	Max. Einflusszeit fürs Nebeneinanderfahren	$1{,}5\,\text{s}$
$\tau_{\text{Æ,max}}$	Längste Zeit unter Aerodynamik-Einfluss	$1{,}5\,\text{s}$
$\tau_{\text{Æ,min}}$	Zeit unter vollem Aerodynamik-Einfluss	$0{,}5\,\text{s}$
w_{veh}	Fahrzeugbreite	$2\,\text{m}$

B. Verwendete Algorithmen und Testobjekte

B.1. Agent mit konstanter lateraler Position

Für das Szenario, bei dem die Eignung eines Algorithmus für die Ausnutzung der Aerodynamik getestet wird, kommt ein Algorithmus zum Einsatz, der sich ohne Interaktion mit dem Ego-Fahrzeug innerhalb des Szenarios bewegt. Dieser Algorithmus nutzt die identische Modellierung wie das neuronale Netz in Abschnitt 7.6.2. Der Unterschied dabei ist, dass zum einen der Parameter für die Breite des Korridors u_{width} zu -1 gesetzt, sodass der Korridor die Breite 0 besitzt. Für jeden Test wird die Position des Korridors u_{lat} auf einen zufälligen Wert zwischen -1 und 1 gesetzt. Auf diese Weise hält das Objekt immer dieselbe laterale Position auf der Strecke. Der Parameter u_{ref} wird zu -1 gesetzt, sodass die Randbedingungen des Referenzalgorithmus keinen Einfluss auf die Trajektorie nehmen.

B.2. Agent für das Szenario zur Evaluierung der Ausnutzung des Windschattens

Die Referenz für das Szenario, in dem die Fähigkeit des Algorithmus getestet wird, den Windschatten eines anderen Fahrzeugs auszunutzen, wurde über einen einfachen Algorithmus definiert, der im Folgenden beschrieben ist. Die Idee dahinter ist, dass der Algorithmus immer direkt hinter dem anderen Fahrzeug fährt, solange es ohne Kollision möglich ist.

Dadurch teilt sich das Verhalten in zwei grundsätzliche Phasen für das Szenario. In der ersten Phase hält das Fahrzeug daher konstant die identische laterale Position zum vorausfahrenden Fahrzeug. (siehe Anhang B.1) Nähert sich das Objekt auf mehr als $20\,\mathrm{m}$, springt die Vorgabe der lateralen Position u_{lat} auf eine zufällige Seite des anderen Fahrzeugs, die für das Fahrzeug ausreichend lateralen Platz zuzüglich einer Sicherheitsmarge von $0{,}5\,\mathrm{m}$ bietet. Der Parameter F wird zu -1 gesetzt, sodass die Randbedingungen des Referenzalgorithmus keinen Einfluss auf die Trajektorie nehmen.

B.3. Algorithmus zur Schätzung einzelner Einträge der Payoff-Matrix

Wie bereits in Abschnitt 6.6.3 beschrieben, werden einzelne fehlende Einträge nach dem folgenden Algorithmus geschätzt:

Algorithmus 4 Berechnungsprozess für fehlende Elemente der Payoff-Matrix

Initialisierung aller unbekannten Werte $p_{nm,\text{est}}$ für alle n für die $\mathbb{K}_n \neq \{\}$ gilt
for Anzahl Wiederholungen mElo **do**
  Initialisierung des Rankings $\boldsymbol{r}$ auf identische Werte
  Initialisierung der Interaktionsmatrix $\mathcal{C}$ auf zufällige Werte
  for Anzahl Wiederholungen Spiele pro mElo **do**
    Ermittlung einer zufälligen Begegnung zwischen Spieler n und Aufgabe m
    if $m \notin \mathbb{K}_n$ **then**
      Ausgang der Begegnung p^*_{nm} entspricht Eintrag in der Payoff-Matrix $\boldsymbol{p}$
    else
      Ausgang der Begegnung p^*_{nm} ist zufällig 0 oder 1 mit der Gewinnwahrscheinlichkeit $p_{nm,\text{est}}$
    end if
    Updates von $\boldsymbol{r}$ und $\mathcal{C}$ gemäß Gleichungen (6-8) und (6-9)
  end for
  Update aller $p_{nm,\text{est}}$ gemäß Gleichung (6-10)
end for

Dafür werden folgende Parameter verwendet:

Tabelle B-1.: Parameter für das Schätzverfahren für einzelne Einträge der Payoff-Matrix

Variable	**Beschreibung**	**Wert**
$\eta_{\mathcal{C}}$	Update-Rate für Interaktions-Matrix	$0{,}1$
η_{est}	Update-Rate für Schätzwert	$0{,}2$
η_r	Update-Rate für Ranking	16
k	Dimension des mElo-Verfahrens	2
$n_{\text{rep,est}}$	Anzahl Durchführungen des mElo-Verfahrens und damit Anzahl Schätzungen	4000
$n_{\text{rep,mElo}}$	Anzahl Spiele pro mElo-Durchlauf	2500

Wie in Abschnitt 6.6.4 beschrieben, wurde die Methode für die Payoff-Matrix des Spiels Schere-Stein-Papier-Feuer-Wasser evaluiert. Dafür wurde jeweils eine Zeile der Matrix dupliziert und danach versucht einen Eintrag in dieser Zeile mittels des Verfahrens zu schätzen. Dafür wurde die Payoff-Matrix zusätzlich modifiziert, sodass nicht nur

Tabelle B-2.: “Schwache“ Payoff-Matrix Schere-Stein-Papier-Feuer-Wasser. Gewinnwahrscheinlichkeit: $0{,}9$

	Schere	Stein	Papier	Feuer	Wasser
Schere	0,5	0,1	0,9	0,1	0,9
Stein	0,9	0,5	0,1	0,1	0,9
Papier	0,1	0,9	0,5	0,1	0,9
Feuer	0,9	0,9	0,9	0,5	0,1
Wasser	0,1	0,1	0,1	0,9	0,5

Tabelle B-3.: Ergebnisse Schätzung Wert aus SSPFW Payoff-Matrix

Gewinnwahrscheinlichkeit	1,0	0,9	0,8	0,7	0,6
Mittelwert |Abweichung|	0,98 %	0,65 %	1,14 %	1,38 %	2,13 %
Standardabweichung |Abweichung|	0,38 %	1,19 %	1,75 %	1,85 %	1,85 %

rein binäre Einträge vorhanden sind, sondern auch “schwache“ Payoffs. “Schwach“ meint in diesem Kontext, dass ein Gewinn nicht mit einer Wahrscheinlichkeit von 1 angenommen wird, wie es für SSPFW eigentlich vorgeschrieben ist, sondern ein Wert $\in]0{,}5, 1]$, Ein Beispiel dafür ist in Tabelle B-2 abgebildet.

Um die Effektivität des Verfahrens zu testen wurde die Gewinnwahrscheinlichkeit zwischen 1 und 0,6 äquidistant variiert. Danach wurde jeweils jede Zeile und jeder Eintrag in der Zeile separat getestet und die Abweichung des geschätzten Wertes in Tabelle B-3 für jeden getesteten Wert der Gewinnwahrscheinlichkeit zusammengefasst.

C. Neuronales Netz und Trainingsbedingungen

Für eine einfachere Nachvollziehbarkeit werden im folgenden die englischen Begriffe für die Beschreibung der Komponenten des neuronalen Netzes genutzt.

C.1. Parameter für Trainingsalgorithmus

Das neuronale Netz, für den Agenten, der in Abschnitt 7.6 vorgestellt wird, wird mittels des Proximal-Policy-Optimization-Verfahrens (PPO) trainiert. Für eine genaue Beschreibung des Verfahrens und d Bedeutung der Parameter wird an dieser Stelle auf die Original-Veröffentlichung des Algorithmus verwiesen.[232] Die gewählten Parameter sind in Tabelle C-1 zusammengefasst.

Tabelle C-1.: Parameter für das genutzte Lernverfahren für das neuronale Netz

Beschreibung	**Wert**
Discount-Faktor	0,99
GAE-Lambda	0,95
Anzahl Frames pro Batch	16 384
Anzahl Frames pro Mini-Batch	4096
Anzahl Lernzyklen pro Batch	4
Lernrate	0,0002
Clipping-Parameter	0,2
Gewichtung Value-Funktion	1
Gewichtung Entropy	0,005

[232] Vgl. Schulman, J. et al.: Proximal Policy Optimization Algorithms (2017).

C.2. Aufbau des neuronalen Netzes

Das verwendete neuronale Netz mit der gewählten Actor-Critic-Struktur, für den in Abschnitt 7.6 vorgestellten Agenten, besteht aus den in Abbildung C-1 abgebildeten Layern. Die Actor-Critic-Struktur wird durch das gewählte Lernverfahren PPO vorgegeben. Für alle Convolutional Layer (CL) wird die Aktivierungsfunktion Rectified Linear Unit (ReLU) und eine Batch-Normalisierung verwendet. Dabei wird folgende Notation genutzt.

- 1-dimensonaler Convolutional Layer:
  1d-C($n_{\text{Eingangsparameter}}$, $n_{\text{Ausgangsparameter}}$, Kernelgröße)
- Fully-Connected Layer:
  FC($n_{\text{Eingangsparameter}}$, $n_{\text{Ausgangsparameter}}$)
- Long-Short Term Memory Layer:
  LSTM($n_{\text{Eingangsparameter}}$, $n_{\text{Ausgangsparameter}}$)

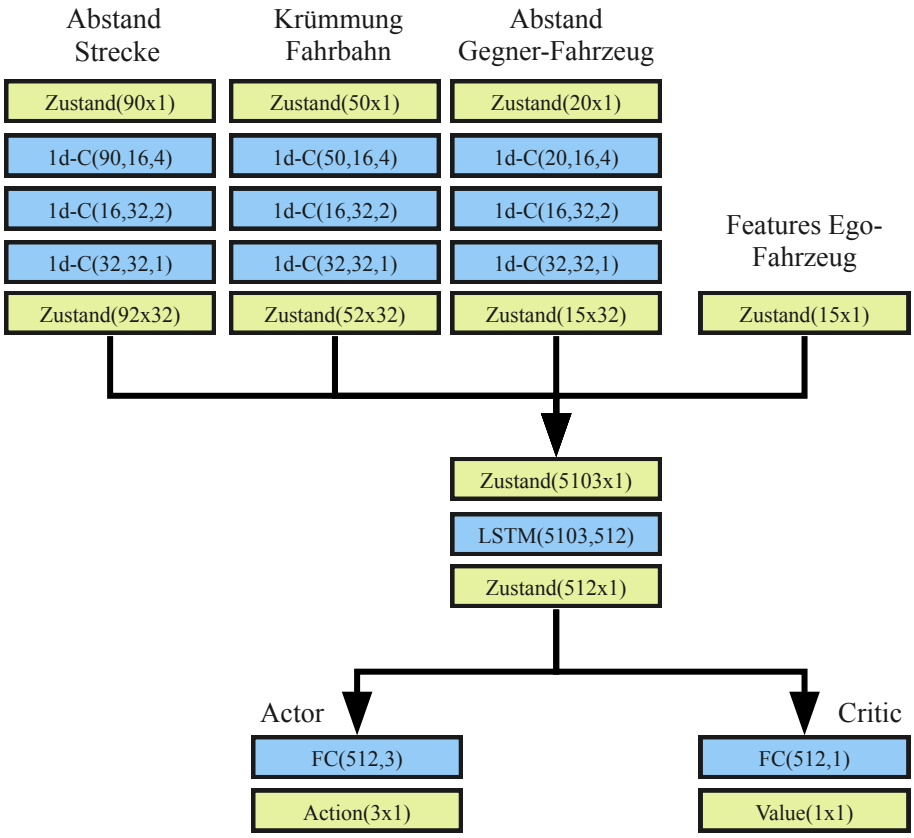


Abbildung C-1.: Aufbau des genutzten neuronalen Netzes mit Actor-Critic Struktur. LSTM: Long-Short Term Memory Layer, FC: Fully Connected Layer, 1d-C: 1-dimensonaler Convolutional Layer, blau: Layer des neuronalen Netzes, grün: Zustand

C.3. Trainingsszenarien

Sowohl für das Überholszenario als auch das Szenario, in dem es Ziel des Agenten ist, das Überholen des anderen Fahrzeugs zu verhindern, werden exemplarisch die beiden Grand-Prix-Strecken-Layouts von Hockenheim und Barcelona-Catalunya zum Training verwendet. Die Objekte, gegen die getestet wird, unterscheiden sich dabei allerdings. Für das Überholszenario wurde folgende Bedingungen gewählt.

- Strecke: Hockenheim-GP, Barcelona-Catalunya-GP
- Startpunkt: Zufällig über Streckenlänge verteilt. Ändert sich nach jedem Lernzyklus
- Startgeschwindigkeit: $10\,\frac{\mathrm{m}}{\mathrm{s}}$
- Abstand zwischen Ego und Gegner-Fahrzeug beim Start: $20\,\mathrm{m}$
- Ego-Parametrisierung: $P_{\text{max}} = 400\,\mathrm{kW}$, $a_{\text{lat,max}} = 20\,\frac{\mathrm{m}}{\mathrm{s}^2}$
- Gegner-Parametrisierung: *variierter Parameter* anhand Rundenzeit-abhängiger Parametrisierung aus Anhang D.1
- Gegner-Verhalten: *variierter Parameter*

Das Szenario, bei dem es Ziel des Agenten ist, das Überholen, des anderen Fahrzeugs zu verhindern, wurde mit folgenden Parametern durchgeführt:

- Strecke: Hockenheim-GP, Barcelona-Catalunya-GP
- Startpunkt: Zufällig über Streckenlänge verteilt. Ändert sich nach jedem Lernzyklus
- Startgeschwindigkeit: $10\,\frac{\mathrm{m}}{\mathrm{s}}$
- Abstand zwischen Ego und Gegner-Fahrzeug beim Start: $20\,\mathrm{m}$
- Ego-Parametrisierung: *variierter Parameter* anhand Rundenzeit-abhängiger Parametrisierung aus Anhang D.1
- Gegner-Parametrisierung: $P_{\text{max}} = 400\,\mathrm{kW}$, $a_{\text{lat,max}} = 20\,\frac{\mathrm{m}}{\mathrm{s}^2}$
- Gegner-Verhalten: *variierter Parameter*

D. Testparameter und -ergebnisse

D.1. Getestete Parametervariationen

Hinweis: Es wird angenommen, dass $D_{\text{max}} = a_{\text{lat,max}}$ gilt.

Wie in Abschnitt 6.3 beschrieben, werden für die beiden Grand-Prix-Strecken Barcelona-Catalunya und Hockenheim die Parameter der Fahrzeuge äquidistant in einem Gitter in Abhängigkeit von der Rundenzeit variiert. Die getesteten Parametrisierungen sind in Tabelle D-1 und Tabelle D-2 aufgelistet.

Tabelle D-1.: Rundenzeit-abhängige Parametrisierung auf Streckenlayout Hockenheim. Angegeben ist die maximal mögliche laterale Beschleunigung $a_{\text{lat,max}}$ in $\frac{\text{m}}{\text{s}^2}$.

Relative Rundenzeit in %		1,005	1,016	1,027	1,039	1,050
P_{max} in kW	300	20,5	21,5	22,7	23,9	25,4
	350	17,9	18,8	19,8	20,8	21,9
	400	16,3	17,0	17,8	18,6	19,6
	450	15,4	16,0	16,6	17,2	18,0
	500	14,8	15,3	15,8	16,4	17,1

Tabelle D-2.: Rundenzeit-abhängige Parametrisierung auf Streckenlayout Barcelona-Catalunya. Angegeben ist die maximal mögliche laterale Beschleunigung $a_{\text{lat,max}}$ in $\frac{\text{m}}{\text{s}^2}$.

Relative Rundenzeit in %		1,005	1,016	1,027	1,039	1,050
P_{max} in kW	300	19,8	20,6	21,5	22,5	23,6
	350	18,0	18,8	19,5	20,4	21,3
	400	16,9	17,5	18,2	18,9	19,7
	450	16,1	16,7	17,3	17,9	18,6
	500	15,5	16,0	16,6	17,1	17,8

D.2. Untersuchung des Einflusses unterschiedlicher Referenzobjekte

Für den Test in Abschnitt 6.1 für den Vergleich des Einflusses von unterschiedlichen Referenzobjekten wurden folgende Testbedingungen verwendet.

- Bezug: Abbildung 6-1
- Strecke: Hockenheim-GP
- Startpunkt: Zufällig über Streckenlänge verteilt
- Startgeschwindigkeit: $10\,\frac{\text{m}}{\text{s}}$
- Abstand zwischen Ego und Gegner-Fahrzeug beim Start: $20\,\text{m}$

- Anzahl Versuche: 500 pro Agentenkombination
- Ego-Parametrierung: $P_{\text{max}} = 400\,\text{kW}$, $a_{\text{lat,max}} = 20\,\frac{\text{m}}{\text{s}^2}$
- Ego-Verhalten: *variierter Parameter*
- Gegner-Parametrisierung: $P_{\text{max}} = 350\,\text{kW}$, $a_{\text{lat,max}} = 18\,\frac{\text{m}}{\text{s}^2}$
- Gegner-Verhalten: *variierter Parameter*

D.3. Untersuchung der Ausnutzung des Windschattens

Für die Untersuchung der Ausnutzung des Windschattens aus Abschnitt 6.7 werden folgende Versuchsbedingungen für die Evaluierung vorgeschlagen.

- Strecke: Zufällig generierter Kreisbogen mit Krümmung $|\kappa| \in [0, 0{,}001\,\frac{1}{\text{m}}]$
- Startgeschwindigkeit: $10\,\frac{\text{m}}{\text{s}}$
- Abstand zwischen Ego und Gegner-Fahrzeug beim Start: $20\,\text{m}$
- Ego-Parametrierung: $P_{\text{max}} = 400\,\text{kW}$, $a_{\text{lat,max}} = 20\,\frac{\text{m}}{\text{s}^2}$
- Ego-Verhalten: *variierter Parameter*
- Gegner-Parametrisierung: $P_{\text{max}} \in [250\,\text{kW}, 400\,\text{kW}]$, $a_{\text{lat,max}} = 20\,\frac{\text{m}}{\text{s}^2}$
- Gegner-Verhalten: Verhalten nach Anhang B.1

D.4. Testergebnisse Ranking

Tabelle D-3.: Payoff-Matrix und resultierendes Ranking für Algorithmen auf dem Barcelona-Catalunya-Grand-Prix-Streckenlayout in Überholszenarien

	NN	LCADM	Referenz	mElo	NG
LCADM	0.491	0.713	0.681	1624(2)	0.4399(2)
Referenz	0.802	0.529	0.826	1854(1)	0.4399(1)
NN	0.380	0.164	0.465	1331(3)	0.1202(3)
Diskreter Reward					
LCADM	0.342	0.571	0.474	1457(1)	0.3922(1)
Referenz	0.513	0.345	0.478	1434(2)	0.3364(2)
NN	0.402	0.407	0.330	1368(3)	0.2714(3)
Kontinuierlicher Reward					
LCADM	0.963	0.891	0.991	2049(2)	0.3333(1)
Referenz	0.984	0.992	1	2332(1)	0.3333(2)
NN	0.710	0.386	0.957	1689(3)	0.3333(3)
Unfälle					

Tabelle D-4.: Payoff-Matrix und resultierendes Ranking für Algorithmen auf dem Barcelona-Catalunya-Grand-Prix-Streckenlayout in Verteidigungsszenarien

	LCADM	Referenz	NN	mElo	NG
NN	0.471	0.181	0.331	1228(1)	0.3688(1)
LCADM	0.178	0.463	0.222	1183(3)	0.2819(3)
Referenz	0.311	0.174	0.492	1216(2)	0.3493(2)
Diskreter Reward					
NN	0.658	0.487	0.598	1580(3)	0.3335(1)
LCADM	0.429	0.655	0.593	1576(1)	0.3333(2)
Referenz	0.526	0.522	0.670	1588(2)	0.3332(3)
Kontinuierlicher Reward					
NN	0.963	0.984	0.710	1939(3)	0.3333(1)
LCADM	0.891	0.992	0.386	1836(1)	0.3333(2)
Referenz	0.991	1	0.957	2243(2)	0.3333(3)
Unfälle					

Tabelle D-5.: Payoff-Matrix und resultierendes Ranking für Algorithmen auf dem Hockenheim-Grand-Prix-Streckenlayout in Überholszenarien

	NN	LCADM	Referenz	mElo	NG
LCADM	0.684	0.689	0.754	1703(2)	0.4419(2)
Referenz	0.822	0.726	0.846	2009(1)	0.4549(1)
NN	0.460	0.107	0.557	1334(3)	0.1032(3)
Diskreter Reward					
LCADM	0.507	0.584	0.514	1516(1)	0.3770(1)
Referenz	0.542	0.435	0.505	1482(2)	0.3277(2)
NN	0.529	0.432	0.456	1444(3)	0.2953(3)
Kontinuierlicher Reward					
LCADM	0.924	0.822	0.981	1968(2)	0.3333(1)
Referenz	0.974	0.994	1	2286(1)	0.3333(2)
NN	0.635	0.260	0.874	1559(3)	0.3333(3)
Unfälle					

Tabelle D-6.: Payoff-Matrix und resultierendes Ranking für Algorithmen auf dem Hockenheim-Grand-Prix-Streckenlayout in Verteidigungsszenarien

	LCADM	Referenz	NN	mElo	NG
NN	0.241	0.152	0.176	1052(3)	0.3052(3)
LCADM	0.132	0.268	0.153	1056(2)	0.3175(2)
Referenz	0.227	0.154	0.317	1074(1)	0.3773(1)
Diskreter Reward					
NN	0.493	0.458	0.471	1488(3)	0.3097(2)
LCADM	0.416	0.565	0.568	1526(2)	0.3451(3)
Referenz	0.486	0.495	0.544	1529(1)	0.3451(1)
Kontinuierlicher Reward					
NN	0.924	0.974	0.635	1866(3)	0.3379(3)
LCADM	0.822	0.994	0.260	1763(1)	0.1699(1)
Referenz	0.981	1	0.874	2142(2)	0.4886(2)
Unfälle					

Literaturverzeichnis

Abdi, S.; Khosravi, H.; Sadiq, S.: Modelling Learners in Adaptive Educational Systems (2021)
Abdi, Solmaz; Khosravi, Hassan; Sadiq, Shazia: Modelling Learners in Adaptive Educational Systems: A Multivariate Glicko-based Approach, 2021

Ajanovic, Z. et al.: Search-Based Motion Planning for Performance Autonomous Driving (2019)
Ajanovic, Zlatan; Regolin, Enrico; Stettinger, Georg; Horn, Martin; Ferrara, Antonella: Search-Based Motion Planning for Performance Autonomous Driving, URL: http://arxiv.org/pdf/1907.07825v1, 2019, Zugriff 18. 07. 2019

Alrifaee, B.: Vernetzte modellbasierte prädiktive Regelung zur Kollisionsvermeidung (2017)
Alrifaee, Bassam: Vernetzte modellbasierte prädiktive Regelung zur Kollisionsvermeidung von Fahrzeugen, 2017

Alrifaee, B. et al.: Real-time Trajectory optimization for Autonomous Vehicle Racing (2018)
Alrifaee, Bassam; Maczijewski, Janis: „Real-time Trajectory optimization for Autonomous Vehicle Racing using Sequential Linearization“, in: 2018 IEEE Intelligent Vehicles Symposium (IV), 2018

Altche, F. et al.: High-speed trajectory planning for autonomous vehicles (2017)
Altche, Florent; Polack, Philip; La Fortelle, Arnaud: „High-speed trajectory planning for autonomous vehicles using a simple dynamic model“, in: 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), 2017

Arab, A. et al.: Motion planning for aggressive autonomous vehicle maneuvers (2016)
Arab, Aliasghar; Yu, Kaiyan; Yi, Jingang; Song, Dezhen: „Motion planning for aggressive autonomous vehicle maneuvers“, in: 2016 IEEE International Conference on Automation Science and Engineering (CASE.21-25 Aug. 2016, 2016

Ates, U.: Long-Term Planning with Deep Reinforcement Learning on Autonomous Drones (2020)
Ates, Ugurkan: Long-Term Planning with Deep Reinforcement Learning on Autonomous Drones, 2020

Bae, H. et al.: Multi-Robot Path Planning Method Using Reinforcement Learning (2019)
Bae, Hyansu; Kim, Gidong; Kim, Jonguk; Qian, Dianwei; Lee, Sukgyu: Multi-Robot Path Planning Method Using Reinforcement Learning, in: Applied Sciences, Vol. 9, S. 3057, 2019

Balduzzi, D. et al.: Re-evaluating Evaluation (2018)
Balduzzi, David; Tuyls, Karl; Perolat, Julien; Graepel, Thore: Re-evaluating Evaluation, in: arXiv:1806.02643 [cs, stat], 2018

Berner, C. et al.: Dota 2 with Large Scale Deep Reinforcement Learning (2019)
Berner, Christopher; Brockman, Greg; Chan, Brooke; Cheung, Vicki; Dennison, Christy; Farhi, David; Fischer, Quirin; Hashme, Shariq; Hesse, Chris; Józefowicz, Rafal; Gray, Scott; Olsson, Catherine; Pachocki, Jakub; Petrov, Michael; Salimans, Tim; Schlatter, Jeremy; Schneider, Jonas; Sidor, Szymon; Sutskever, Ilya; Tang, Jie; Wolski, Filip; Zhang, Susan: Dota 2 with Large Scale Deep Reinforcement Learning, S. 66, 2019

Bevilacqua, M. et al.: Particle swarm for path planning in a racing circuit simulation (2017)
Bevilacqua, M.; Tsourdos, A.; Starr, A.: „Particle swarm for path planning in a racing circuit simulation“, in: 2017 IEEE International Instrumentation and Measurement Technology Conference (I2MTC), 2017

Bojarski, M. et al.: End to End Learning for Self-Driving Cars (2016)
Bojarski, Mariusz; Del Testa, Davide; Dworakowski, Daniel; Firner, Bernhard; Flepp, Beat; Goyal, Prasoon; Jackel, Lawrence D.; Monfort, Mathew; Muller, Urs; Zhang, Jiakai; Zhang, Xin; Zhao, Jake; Zieba, Karol: End to End Learning for Self-Driving Cars, in: arXiv:1604.07316 [cs], 2016

Bojarski, M. et al.: End-to-End Trained Deep Neural Network Steers a Car (2017)
Bojarski, Mariusz; Yeres, Philip; Choromanska, Anna; Choromanski, Krzysztof; Firner, Bernhard; Jackel, Lawrence; Muller, Urs: Explaining How a Deep Neural Network Trained with End-to-End Learning Steers a Car, in: arXiv:1704.07911 [cs], 2017

Butyrev, L. et al.: Deep Reinforcement Learning for Motion Planning of Mobile Robots (2019)
Butyrev, Leonid; Edelhäußer, Thorsten; Mutschler, Christopher: Deep Reinforcement Learning for Motion Planning of Mobile Robots, S. 6, 2019

Cai, P. et al.: High-speed Autonomous Drifting with Deep Reinforcement Learning (2020)
Cai, Peide; Mei, Xiaodong; Tai, Lei; Sun, Yuxiang; Liu, Ming: High-speed Autonomous Drifting with Deep Reinforcement Learning, 2020

Cang Ye et al.: A fuzzy controller with supervised learning assisted RL (2003)
Cang Ye; Yung, N. H. C.; Danwei Wang: A fuzzy controller with supervised learning assisted reinforcement learning algorithm for obstacle avoidance, in: IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), Vol. 33, S. 17–27, 2003

Carpinetti, L. C. et al.: What to benchmark? (2002)
Carpinetti, Luiz C.R.; Melo, Alexandre M. de: What to benchmark? A systematic approach and cases, in: Benchmarking: An International Journal, Vol. 9, S. 244–255, 2002

Chen, J. et al.: Interpretable End-to-end Urban Autonomous Driving with Latent Deep RL (2020)
Chen, Jianyu; Li, Shengbo Eben; Tomizuka, Masayoshi: Interpretable End-to-end Urban Autonomous Driving with Latent Deep Reinforcement Learning, 2020

Christ, F. et al.: Time-optimal trajectory planning considering variable tyre-road friction (2019)
Christ, Fabian; Wischnewski, Alexander; Heilmeier, Alexander; Lohmann, Boris: Time-optimal trajectory planning for a race car considering variable tyre-road friction coefficients, in: Vehicle System Dynamics, Vol. 3, S. 1–25, 2019

Christiano, P. et al.: Deep reinforcement learning from human preferences (2017)
Christiano, Paul; Leike, Jan; Brown, Tom B.; Martic, Miljan; Legg, Shane; Amodei, Dario: Deep reinforcement learning from human preferences, 2017

Cooper, M. et al.: Stackelberg Punishment and Bully-Proofing Autonomous Vehicles (2019)
Cooper, Matt; Lee, Jun Ki; Beck, Jacob; Fishman, Joshua D.; Gillett, Michael; Papakipos, Zoë; Zhang, Aaron; Ramos, Jerome; Shah, Aansh; Littman, Michael L.: Stackelberg Punishment and Bully-Proofing Autonomous Vehicles, in: arXiv:1908.08641 [cs], 2019

Czarnecki, K.: Operational World Model Ontology for Automated Driving Systems - Part 2 (2018)
Czarnecki, Krzysztof: Operational World Model Ontology for Automated Driving Systems - Part 2: Road Users, Animals, Other Obstacles, and Environmental Conditions, 2018

Dominy, R. G.: The Influence of Slipstreaming on the Performance of a Grand Prix Racing Car (1990)
Dominy, R. G.: The Influence of Slipstreaming on the Performance of a Grand Prix Racing Car, in: Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering, Vol. 204, S. 35–40, 1990

Dong, H.; Ding, Z.; Zhang, S.: Deep RL: Fundamentals, Research and Applications (2020)
Dong, Hao; Ding, Zihan; Zhang, Shanghang: Deep Reinforcement Learning: Fundamentals, Research and Applications, Springer Singapore, 2020

Doubek, F. et al.: What makes a good driver on public roads and race tracks? (2020)
Doubek, Fabian; Salzmann, Falk; Winter, Joost: What makes a good driver on public roads and race tracks? An Interview Study, URL: https://www.researchgate.net/publication/343651540_What_makes_a_good_driver_on_public_roads_and_race_tracks_An_Interview_Study, 2020

Drew, S. A. W.: From knowledge to action (1997)
Drew, Stephen A. W.: From knowledge to action: the impact of benchmarking on organizational performance, in: Long Range Planning, Vol. 30, S. 427–441, 1997

Duan, J. et al.: A Generalized Model for Multidimensional Intransitivity (2017)
Duan, Jiuding; Li, Jiyi; Baba, Yukino; Kashima, Hisashi: „A Generalized Model for Multidimensional Intransitivity“, in: 2017

Džijan, I. et al.: Aerodynamic characteristics of two slipstreaming race cars (2021)
Džijan, Ivo; Pašić, Aleksandar; Buljac, Andrija; Kozmar, Hrvoje: Aerodynamic characteristics of two slipstreaming race cars, in: Journal of Mechanical Science and Technology, Vol. 35, S. 179–186, 2021

Ebtekar, A. et al.: An Elo-like System for Massive Multiplayer Competitions (2021)
Ebtekar, Aram; Liu, Paul: An Elo-like System for Massive Multiplayer Competitions, in: arXiv:2101.00400 [cs, stat], 2021

Elo, A. E.: The Rating of Chessplayers, Past and Present (1978)
Elo, Arpad E.: The Rating of Chessplayers, Past and Present, Arco Pub., 1978

Everitt, T. et al.: Reinforcement Learning with a Corrupted Reward Channel (2017)
Everitt, Tom; Krakovna, Victoria; Orseau, Laurent; Hutter, Marcus; Legg, Shane: Reinforcement Learning with a Corrupted Reward Channel, in: arXiv:1705.08417 [cs, stat], 2017

Facchinei, F. et al.: Generalized Nash equilibrium problems (2007)
Facchinei, Francisco; Kanzow, Christian: Generalized Nash equilibrium problems, in: 4OR, Vol. 5, S. 173–210, 2007

Fan, T. et al.: Getting Robots Unfrozen and Unlost in Dense Pedestrian Crowds (2018)
Fan, Tingxiang; Cheng, Xinjing; Pan, Jia; Long, Pinxin; Liu, Wenxi; Yang, Ruigang; Manocha, Dinesh: Getting Robots Unfrozen and Unlost in Dense Pedestrian Crowds, in: arXiv:1810.00352 [cs], 2018

FIA: FIA WEC - Les Mans 2020 News (2021)
FIA: 2020 Le Mans - FIA World Endurance Championship, URL: https://www.fiawec.com/en/race/show/4643, 2021, Zugriff 01. 04. 2021

FIA: Appendix H - Supervision of the road and emergency services (2021)
FIA: Appendix H - Recommendations for the supervision of the road and emergency services, URL: https://www.fia.com/sites/default/files/appendix_h_2021_published_09_03_2021.pdf, 2021

FIA: Appendix L - Driver's licenses, equipment and conduct (2021)
FIA: Appendix L - International Drivers' licences, medical examinations, driver's equipment and conduct, URL: https://www.fia.com/sites/default/files/appendix_l_2021_publie_le_08_juillet_2021.pdf, 2021

FIA: Formula 1 Sporting Regulations Issue 5 (2021)
FIA: Formula 1 Sporting Regulations Issue 5, URL: https://www.fia.com/sites/default/files/2021_formula_1_sporting_regulations_-_iss_5_-_2020-12-16.pdf, 2021

FIA: Formula 1 Sporting Regulations Issue 9 (2021)
FIA: Formula 1 Sporting Regulations Issue 9, URL: https://www.fia.com/sites/default/files/2021_formula_1_sporting_regulations_-_iss_9_-_2021-05-17.pdf, 2021

FIA: Formula 1 Technical Regulations Issue 9 (2021)
FIA: Formula 1 Technical Regulations Issue 9, URL: https://www.fia.com/sites/default/files/2021_formula_1_technical_regulations_-_iss_9_-_2021-03-05.pdf, 2021

FIA: Formula e Sporting Regulations (2021)
FIA: Formula e Sporting Regulations, URL: https://www.fia.com/sites/default/files/2020-2021_fia_fe_sporting_regulations_s7_marked-up_060521_v1.pdf, 2021

FIA: List of FIA licensed circuits (2020)
FIA: List of FIA licensed circuits, URL: https://www.fia.com/sites/default/files/list_of_fia_licenced_circuits_0.pdf, 2020

FIA: Technical Regulations for Grand Touring Cars (2020)
FIA: Technical Regulations for Grand Touring Cars, URL: https://www.fia.com/sites/default/files/2021_lmgte_wmsc_2020.12.16_-_published_2020.12.04.pdf, 2020

Frank, J. et al.: Reinforcement learning in the presence of rare events (2008)
Frank, Jordan; Mannor, Shie; Precup, Doina: „Reinforcement learning in the presence of rare events", in: 2008

Frego, M. et al.: Trajectory planning for car-like vehicles: A modular approach (2016)
Frego, Marco; Bevilacqua, Paolo; Bertolazzi, Enrico; Biral, Francesco; Fontanelli, Daniele; Palopoli, Luigi: „Trajectory planning for car-like vehicles: A modular approach", in: 2016 IEEE 55th Conference on Decision and Control (CDC.ARIA Resort & Casino, December 12-14, 2016, Las Vegas, USA, 2016

Frison, G. et al.: HPIPM: a high-performance quadratic programming framework (2020)
Frison, Gianluca; Diehl, Moritz: HPIPM: a high-performance quadratic programming framework for model predictive control, URL: https://arxiv.org/pdf/2003.02547.pdf, 2020

Frison, G. et al.: BLASFEO (2018)
Frison, Gianluca; Kouzoupis, Dimitris; Sartor, Tommaso; Zanelli, Andrea; Diehl, Moritz: BLASFEO: Basic Linear Algebra Subroutines for Embedded Optimization, in: ACM Trans. Math. Softw. Vol. 44, 42:1–42:30, 2018

Frömmig, L.: Grundkurs Rennwagentechnik (2019)
Frömmig, Lars: Grundkurs Rennwagentechnik: Einführung in das Zusammenwirken von Reifen, Fahrwerk, Aerodynamik, Differenzialsperren und Rahmen, Springer Fachmedien Wiesbaden, 2019

Fuchs, F. et al.: Super-Human Performance in Gran Turismo Sport Using Deep RL (2020)
Fuchs, Florian; Song, Yunlong; Kaufmann, Elia; Scaramuzza, Davide; Duerr, Peter: Super-Human Performance in Gran Turismo Sport Using Deep Reinforcement Learning, 2020

Garlick, S. et al.: Real-Time Trajectory Planning Using Machine Learning (2021)
Garlick, Sam; Bradley, Andrew: Real-Time Optimal Trajectory Planning for Autonomous Vehicles and Lap Time Simulation Using Machine Learning, in: arXiv:2102.02315 [cs], 2021

Gerdts, M. et al.: Motion Planning in Driving Scenarios with Communicating Vehicles (2018)
Gerdts, Matthias; Martens, Björn: Optimization-based Motion Planning in Virtual Driving Scenarios with Application to Communicating Autonomous Vehicles, 2018

Glickman, M.: Example of the Glicko-2 system (2013)
Glickman, Mark: Example of the Glicko-2 system, S. 6, 2013

Grasso, G. M. et al.: A flexible environment for autonomous driving interaction testing (2020)
Grasso, G. M.; d'Italia, G.; Battiato, S.: „A flexible virtual environment for autonomous driving agent-human interaction testing", in: 2020 AEIT International Conference of Electrical and Electronic Technologies for Automotive (AEIT AUTOMOTIVE), 2020

Gu, S. et al.: Continuous Deep Q-Learning with Model-based Acceleration (2016)
Gu, Shixiang; Lillicrap, Timothy; Sutskever, Ilya; Levine, Sergey: Continuous Deep Q-Learning with Model-based Acceleration, 2016

Gutjahr, B. et al.: Lateral Vehicle Trajectory Optimization Using Linear MPC (2016)
Gutjahr, Benjamin; Groll, Lutz; Werling, Moritz: Lateral Vehicle Trajectory Optimization Using Constrained Linear Time-Varying MPC, in: IEEE Trans. Intell. Transport. Syst. S. 1–10, 2016

Haarnoja, T. et al.: Soft Actor-Critic (2018)
Haarnoja, Tuomas; Zhou, Aurick; Abbeel, Pieter; Levine, Sergey: „Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor“, in: International Conference on Machine Learning, 2018

Hart, P. E. et al.: A Formal Basis for the Heuristic Determination of Minimum Cost Paths (1968)
Hart, P. E.; Nilsson, N. J.; Raphael, B.: A Formal Basis for the Heuristic Determination of Minimum Cost Paths, in: IEEE Transactions on Systems Science and Cybernetics, Vol. 4, S. 100–107, 1968

He, H. et al.: Opponent Modeling in Deep Reinforcement Learning (2016)
He, He; Boyd-Graber, Jordan; Kwok, Kevin; Daumé III, Hal: Opponent Modeling in Deep Reinforcement Learning, in: arXiv:1609.05559 [cs], 2016

Heinrich, J. et al.: Deep Learning from Self-Play in Imperfect-Information Games (2016)
Heinrich, Johannes; Silver, David: Deep Reinforcement Learning from Self-Play in Imperfect-Information Games, in: arXiv:1603.01121 [cs], 2016

Herbrich, R. et al.: TrueSkill : A Bayesian Skill Rating System (2007)
Herbrich, Ralf; Minka, Tom; Graepel, Thore: „TrueSkill : A Bayesian Skill Rating System“, in: 2007

Herman, J. et al.: Learn-to-Race (2021)
Herman, James; Francis, Jonathan; Ganju, Siddha; Chen, Bingqing; Koul, Anirudh; Gupta, Abhinav; Skabelkin, Alexey; Zhukov, Ivan; Kumskoy, Max; Nyberg, Eric: Learn-to-Race: A Multimodal Control Environment for Autonomous Racing, in: arXiv:2103.11575 [cs], 2021

Hermansdorfer, L. et al.: Benchmarking of a software stack for autonomous racing (2020)
Hermansdorfer, Leonhard; Betz, Johannes; Lienkamp, Markus: Benchmarking of a software stack for autonomous racing against a professional human race driver, 2020

Hochreiter, S. et al.: Long Short-term Memory (1997)
Hochreiter, Sepp; Schmidhuber, Jürgen: Long Short-term Memory, in: Neural computation, Vol. 9, S. 1735–80, 1997

Huang, H.-H. et al.: Learning overtaking and blocking skills in simulated car racing (2015)
Huang, Han-Hsien; Wang, Tsaipei: „Learning overtaking and blocking skills in simulated car racing“, in: 2015 IEEE Conference on Computational Intelligence and Games (CIG), 2015

Hubmann, C. et al.: Planning With Interaction and Uncertain Prediction (2018)
Hubmann, Constantin; Schulz, Jens; Becker, Marvin; Althoff, Daniel; Stiller, Christoph: Automated Driving in Uncertain Environments: Planning With Interaction and Uncertain Maneuver Prediction, in: IEEE Trans. Intell. Veh. Vol. 3, S. 5–17, 2018

Huppler, K.: The Art of Building a Good Benchmark (2009)
Huppler, Karl: „The Art of Building a Good Benchmark“, in: 2009

Inoue, H. et al.: Intelligent Driving System for Safer Automobiles (2017)
Inoue, Hideo; Raksincharoensak, Pongsathorn; Inoue, Shintaro: Intelligent Driving System for Safer Automobiles, in: Journal of Information Processing, Vol. 25, S. 32–43, 2017

Jacobsen, J.-H. et al.: Shortcuts: Neural Networks Love To Cheat (2020)
Jacobsen, Jörn-Henrik; Geirhos, Robert; Michaelis, Claudio: Shortcuts: Neural Networks Love To Cheat, in: The Gradient, 2020

Junietz, P. M.: Dissertation, Microscopic and Macroscopic Risk Metrics for Safety Validation (2019)
Junietz, Philipp Matthias: Microscopic and Macroscopic Risk Metrics for the Safety Validation of Automated Driving, Dissertation Technische Universität, 2019

Kaiser, Ł. et al.: Learning to Remember Rare Events (2017)
Kaiser, Łukasz; Nachum, Ofir; Roy, Aurko; Bengio, Samy: „Learning to Remember Rare Events“, in: 2017

Kala, R. et al.: Multi-Level Planning for Semi-autonomous Vehicles in Traffic Scenarios (2013)
Kala, Rahul; Warwick, Kevin: Multi-Level Planning for Semi-autonomous Vehicles in Traffic Scenarios Based on Separation Maximization, in: J Intell Robot Syst, Vol. 72, S. 559–590, 2013

Karakovskiy, S. et al.: The Mario AI Benchmark and Competitions (2012)
Karakovskiy, S.; Togelius, J.: The Mario AI Benchmark and Competitions, in: IEEE Transactions on Computational Intelligence and AI in Games, Vol. 4, S. 55–67, 2012

Katz, J.: Aerodynamics of race cars (2006)
Katz, Joseph: Aerodynamics of race cars, in: Annu. Rev. Fluid Mech, Vol. 38, S. 27–63, 2006

Klimenko, A. Y.: Intransitivity in Theory and in the Real World (2015)
Klimenko, Alexander Y.: Intransitivity in Theory and in the Real World, in: Entropy, Vol. 17, S. 4364–4412, 2015

Kloeser, D. et al.: NMPC for Racing Using a Singularity-Free Model with Obstacle Avoidance (2020)
Kloeser, Daniel; Schoels, Tobias; Sartor, Tommaso; Zanelli, Andrea; Frison, Gianluca; Diehl, Moritz: NMPC for Racing Using a Singularity-Free Path-Parametric Model with Obstacle Avoidance, URL: https://cdn.syscop.de/publications/Kloeser2020.pdf, 2020

Kloock, M. et al.: Networked Model Predictive Vehicle Race Control (2019)
Kloock, Maximilian; Scheffe, Patrick; Botz, Lukas; Maczijewski, Janis; Alrifaee, Bassam; Kowalewski, Stefan: „Networked Model Predictive Vehicle Race Control“, in: 2019 IEEE Intelligent Transportation Systems Conference (ITSC), 2019

Knox, W. B. et al.: Reward (Mis)design for Autonomous Driving (2021)
Knox, W. Bradley; Allievi, Alessandro; Banzhaf, Holger; Schmitt, Felix; Stone, Peter: Reward (Mis)design for Autonomous Driving, in: arXiv:2104.13906 [cs], 2021

Lavalle, S. M.: Rapidly-Exploring Random Trees (1998)
Lavalle, Steven M.: Rapidly-Exploring Random Trees: A New Tool for Path Planning, 1998

Lazaridis, D.: An R package for multidimensional Elo ratings (2021)
Lazaridis, David: An R package for multidimensional Elo ratings, URL: https://github.com/dclaz/mELO, 2021, Zugriff 26. 05. 2021

Lee, C.-S. et al.: Human vs. Computer Go: Review and Prospect (2016)
Lee, Chang-Shing; Wang, Mei-Hui; Yen, Shi-Jim; Wei, Ting-Han; Wu, I-Chen; Chou, Ping-Chiang; Chou, Chun-Hsun; Wang, Ming-Wan; Yang, Tai-Hsiung; Academy, Haifong Weiqi: Human vs. Computer Go: Review and Prospect, S. 6, 2016

Lei, X. et al.: Dynamic Path Planning of Unknown Environment Based on Deep RL (2018)
Lei, Xiaoyun; Zhang, Zhian; Dong, Peifang: Dynamic Path Planning of Unknown Environment Based on Deep Reinforcement Learning, in: Journal of Robotics, 2018

Li, D. et al.: RL and Deep Learning based Lateral Control for Autonomous Driving (2018)
Li, Dong; Zhao, Dongbin; Zhang, Qichao; Chen, Yaran: Reinforcement Learning and Deep Learning based Lateral Control for Autonomous Driving, S. 14, 2018

Likmeta, A. et al.: Combining RL with rule-based controllers for autonomous driving (2020)
Likmeta, Amarildo; Metelli, Alberto Maria; Tirinzoni, Andrea; Giol, Riccardo; Restelli, Marcello; Romano, Danilo: Combining reinforcement learning with rule-based controllers for transparent and general decision-making in autonomous driving, in: Robotics and Autonomous Systems, S. 103568, 2020

Limited, M. R.: Formula One Race Strategy (2021)
Limited, McLaren Racing: Formula One Race Strategy, URL: https://www.raeng.org.uk/publications/other/14-car-racing, 2021

Liniger, A. et al.: Optimization-Based Autonomous Racing of 1:43 Scale RC Cars (2015)
Liniger, Alexander; Domahidi, Alexander; Morari, Manfred: Optimization-Based Autonomous Racing of 1:43 Scale RC Cars, in: Optim. Control Appl. Meth. Vol. 36, S. 628–647, 2015

Liniger, A. et al.: A Non-Cooperative Game Approach to Autonomous Racing (2017)
Liniger, Alexander; Lygeros, John: A Non-Cooperative Game Approach to Autonomous Racing, S. 14, 2017

Loiacono, D. et al.: Learning to overtake in TORCS using simple reinforcement learning (2010)
Loiacono, D.; Prete, A.; Lanzi, P. L.; Cardamone, L.: „Learning to overtake in TORCS using simple reinforcement learning“, in: IEEE Congress on Evolutionary Computation, 2010

Ma, J.: pybrium: Strategy and equilibria toolkit (2019)
Ma, Jerry: pybrium: Strategy and equilibria toolkit, URL: https://github.com/jma127/pybrium, 2019

Maiza, A.: Reinforcement Learning for Formula 1 Race Strategy (2020)
Maiza, Ashref: Reinforcement Learning for Formula 1 Race Strategy. Medium, URL: https://towardsdatascience.com/reinforcement-learning-for-formula-1-race-strategy-7f29c966472a, 2020, Zugriff 07. 05. 2021

Maurer, M.; Gerdes, J. C.; Lenz, B.; Winner, H.: Autonomes Fahren (2015)
Maurer, Markus; Gerdes, J. Christian; Lenz, Barbara; Winner, Hermann: Autonomes Fahren, Springer Berlin Heidelberg, 2015

McCloskey, M. et al.: Catastrophic Interference in Connectionist Networks (1989)
McCloskey, Michael; Cohen, Neal J.: Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem, in: Bower, Gordon H. (Hrsg.): Psychology of Learning and Motivation, Academic Press, 1989

Michael Romanidis: Codemasters F1 2019 Australia World Record (2019)
Michael Romanidis: Codemasters F1 2019 Australia World Record, URL: https://www.youtube.com/watch?v=njJLfs8oyB4, 2019, Zugriff 05. 05. 2021

Minton, E. A.; Kahle, L. R.: Belief systems, religion, and behavioral economics (2014)
Minton, Elizabeth A; Kahle, Lynn R: Belief systems, religion, and behavioral economics: marketing in multicultural environments, 2014

Mirchevska, B. et al.: High-level Decision Making for Safe Lane Changing using RL (2018)
Mirchevska, Branka; Pek, Christian; Werling, Moritz; Althoff, Matthias; Boedecker, Joschka: „High-level Decision Making for Safe and Reasonable Autonomous Lane Changing using Reinforcement Learning“, in: 2018 IEEE Intelligent Transportation Systems Conference.November 4-7, Maui, Hawaii, 2018

Nagabandi, A. et al.: Learning in Dynamic, Real-World Environments through Meta-RL (2018)
Nagabandi, Anusha; Clavera, Ignasi; Liu, Simin; Fearing, Ronald S.; Abbeel, Pieter; Levine, Sergey; Finn, Chelsea: „Learning to Adapt in Dynamic, Real-World Environments through Meta-Reinforcement Learning“, in: International Conference on Learning Representations, 2018

Niu, J. et al.: Two-Stage Safe Reinforcement Learning for High-Speed Autonomous Racing (2020)
Niu, J.; Hu, Y.; Jin, B.; Han, Y.; Li, X.: „Two-Stage Safe Reinforcement Learning for High-Speed Autonomous Racing“, in: 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2020

Nolte, M. et al.: Towards a skill- and ability-based development process for road vehicles (2017)
Nolte, M.; Bagschik, G.; Jatzkowski, I.; Stolte, T.; Reschka, A.; Maurer, M.: „Towards a skill- and ability-based development process for self-aware automated road vehicles“, in: 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), 2017

Omidshafiei, S. et al.: α-Rank: Multi-Agent Evaluation by Evolution (2019)
Omidshafiei, Shayegan; Papadimitriou, Christos; Piliouras, Georgios; Tuyls, Karl; Rowland, Mark; Lespiau, Jean-Baptiste; Czarnecki, Wojciech M.; Lanctot, Marc; Perolat, Julien; Munos, Remi: α-Rank: Multi-Agent Evaluation by Evolution, in: Sci Rep, Vol. 9, 2019

Onieva, E. et al.: Overtaking opponents with blocking strategies using fuzzy logic (2010)
Onieva, E.; Cardamone, L.; Loiacono, D.; Lanzi, P. L.: „Overtaking opponents with blocking strategies using fuzzy logic“, in: Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games, 2010

Osiński, B. et al.: CARLA Real Traffic Scenarios - novel training ground and benchmark (2020)
Osiński, Błażej; Miłoś, Piotr; Jakubowski, Adam; Zięcina, Paweł; Martyniak, Michał; Galias, Christopher; Breuer, Antonia; Homoceanu, Silviu; Michalewski, Henryk: CARLA Real Traffic Scenarios – novel training ground and benchmark for autonomous driving, in: arXiv:2012.11329 [cs], 2020

Owen, G.: Spieltheorie (1968)
Owen, Guillermo: Zwei-Personen-Nullsummenspiele, in: Owen, Guillermo (Hrsg.): Spieltheorie, Springer, 1968

Pacejka, H. B. et al.: The Magic Formula Tyre Model (1992)
Pacejka, Hans B.; Bakker, Egbert: The Magic Formula Tyre Model, in: Vehicle System Dynamics, Vol. 21, S. 1–18, 1992

Paden, B. et al.: Survey of Motion Planning and Control Techniques for Urban Vehicles (2016)
Paden, Brian; Cap, Michal; Yong, Sze Zheng; Yershov, Dmitry; Frazzoli, Emilio: A Survey of Motion Planning and Control Techniques for Self-driving Urban Vehicles, 2016

Plessen, M. G. et al.: Trajectory Planning Under Vehicle Dimension Constraints Using SLP (2017)
Plessen, Mogens Graf; Lima, Pedro F.; Martensson, Jonas; Bemporad, Alberto; Wahlberg, Bo: Trajectory Planning Under Vehicle Dimension Constraints Using Sequential Linear Programming, S. 7, 2017

Prisco, A. et al.: A multidimensional ELO model for matching learning objects (2018)
Prisco, André; Penna, Rafael; Evandro; Botelho, Silvia; Tonin, Neilor; Bez, Jean: „A multidimensional ELO model for matching learning objects“, in: 2018 IEEE Frontiers in Education Conference (FIE), 2018

Qiao, Z. et al.: Automatically Generated Curriculum based RL for Autonomous Vehicles (2018)
Qiao, Zhiqian; Muelling, Katharina; Dolan, John M.; Palanisamy, Praveen; Mudalige, Priyantha: „Automatically Generated Curriculum based Reinforcement Learning for Autonomous Vehicles in Urban Environment“, in: 2018 IEEE Intelligent Vehicles Symposium (IV), 2018

Rabe, M. et al.: Verifikation und Validierung - Definitionen (2008)
Rabe, Markus; Spiekermann, Sven; Wenzel, Sigrid: Definitionen, in: (Hrsg.): Verifikation und Validierung für die Simulation in Produktion und Logistik: Vorgehensmodelle und Techniken, Springer, 2008

Rakelly, K. et al.: Efficient Off-Policy Meta-Reinforcement Learning via Probabilistic Variables (2019)
Rakelly, Kate; Zhou, Aurick; Finn, Chelsea; Levine, Sergey; Quillen, Deirdre: „Efficient Off-Policy Meta-RL via Probabilistic Context Variables“, in: International Conference on Machine Learning, 2019

Ratcliff, R.: Connectionist models of recognition memory (1990)
Ratcliff, R.: Connectionist models of recognition memory: constraints imposed by learning and forgetting functions, in: Psychol Rev, Vol. 97, S. 285–308, 1990

Rizano, T. et al.: Global path planning for competitive robotic cars (2013)
Rizano, Tizar; Fontanelli, Daniele; Palopoli, Luigi; Pallottino, Lucia; Salaris, Paolo: „Global path planning for competitive robotic cars“, in: 2013 IEEE 52nd Annual Conference on Decision and Control (CDC.10-13 Dec. 2013, Florence, Italy, 2013

Saavedra, R. H. et al.: Analysis of benchmark characteristics and -performance (1996)
Saavedra, Rafael H.; Smith, Alan J.: Analysis of benchmark characteristics and benchmark performance prediction, in: ACM Trans. Comput. Syst. Vol. 14, S. 344–384, 1996

Sander, R.: Emergent Autonomous Racing Via Multi-Agent Proximal Policy Optimization (2020)
Sander, Ryan: Emergent Autonomous Racing Via Multi-Agent Proximal Policy Optimization, 2020

Schulman, J. et al.: Proximal Policy Optimization Algorithms (2017)
Schulman, John; Wolski, Filip; Dhariwal, Prafulla; Radford, Alec; Klimov, Oleg: Proximal Policy Optimization Algorithms, in: arXiv:1707.06347 [cs], 2017

Segers, J.: Analysis techniques for racecar data acquisition (2014)
Segers, Jörge: Analysis techniques for racecar data acquisition, Second Edition. Auflage, SAE International, 2014

Silver, D. et al.: A general RL algorithm that masters chess and Go through self-play (2018)
Silver, David; Hubert, Thomas; Schrittwieser, Julian; Antonoglou, Ioannis; Lai, Matthew; Guez, Arthur; Lanctot, Marc; Sifre, Laurent; Kumaran, Dharshan; Graepel, Thore; Lillicrap, Timothy; Simonyan, Karen; Hassabis, Demis: A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play, in: Science (New York, N.Y.) Vol. 362, S. 1140–1144, 2018

Simaan, M. et al.: On the Stackelberg strategy in nonzero-sum games (1973)
Simaan, M.; Cruz, J. B.: On the Stackelberg strategy in nonzero-sum games, in: J Optim Theory Appl, Vol. 11, S. 533–555, 1973

Sippel, M. et al.: Result Video: Two Behavior Algorithms Driving the Racecar (2020)
Sippel, Marco; Winner, Hermann: Result Video: Two Behavior Algorithms Driving the Racecar, URL: https://doi.org/10.25534/tudatalib-344, 2020

Sippel, M. et al.: Source Code: Sequential Linearized Motion Planner for Racecar Behavior (2020)
Sippel, Marco; Winner, Hermann: Source Code: Sequential Linearized Motion Planner for Racecar Behavior, URL: https://doi.org/10.25534/tudatalib-353, 2020

Song, Y. et al.: Autonomous Overtaking in Gran Turismo Sport Using Curriculum RL (2021)
Song, Yunlong; Lin, HaoChih; Kaufmann, Elia; Duerr, Peter; Scaramuzza, Davide: Autonomous Overtaking in Gran Turismo Sport Using Curriculum Reinforcement Learning, in: arXiv:2103.14666 [cs], 2021

Stahl, T. et al.: An Open-Source Scenario Architect for Autonomous Vehicles (2020)
Stahl, Tim; Betz, Johannes: An Open-Source Scenario Architect for Autonomous Vehicles, S. 8, 2020

Stahl, T. et al.: Multilayer Graph-Based Trajectory Planning for Race Vehicles (2019)
Stahl, Tim; Wischnewski, Alexander; Betz, Johannes; Lienkamp, Markus: „Multilayer Graph-Based Trajectory Planning for Race Vehicles in Dynamic Scenarios“, in: 2019 IEEE Intelligent Transportation Systems Conference (ITSC), 2019

Stone, P.: Encyclopedia of Machine Learning: Reinforcement Learning (2010)
Stone, Peter: Reinforcement Learning, in: Sammut, Claude; Webb, Geoffrey I. (Hrsg.): Encyclopedia of Machine Learning, Springer US, 2010

Subosits, J. et al.: Real-time Trajectory Replanning for Autonomous Driving (2019)
Subosits, John; Gerdes, J. Christian: From the Racetrack to the Road: Real-time Trajectory Replanning for Autonomous Driving, in: IEEE Trans. Intell. Veh. S. 1, 2019

Timbers, F. et al.: Approximate exploitability (2020)
Timbers, Finbarr; Lockhart, Edward; Lanctot, Marc; Schmid, Martin; Schrittwieser, Julian; Hubert, Thomas; Bowling, Michael: Approximate exploitability: Learning a best response in large games, in: arXiv:2004.09677 [cs, stat], 2020

Tobin, J. et al.: Domain Randomization for Transferring Deep NN to the Real World (2017)
Tobin, Josh; Fong, Rachel; Ray, Alex; Schneider, Jonas; Zaremba, Wojciech; Abbeel, Pieter: Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World, in: arXiv:1703.06907 [cs], 2017

Trautman, P. et al.: Unfreezing the robot (2010)
Trautman, P.; Krause, A.: „Unfreezing the robot: Navigation in dense, interacting crowds“, in: 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2010

Tremayne, D.: The concise encyclopedia of Formula One (2000)
Tremayne, David: The concise encyclopedia of Formula One, Bath, England : Parragon Pub., 2000

Trzesniowski, M.: Rennwagentechnik - Gesamtfahrzeug (2019)
Trzesniowski, Michael: Gesamtfahrzeug, Springer Fachmedien Wiesbaden, 2019

Trzesniowski, M.; Eder, P.: Rennwagentechnik - Datenanalyse, Abstimmung (2017)
Trzesniowski, Michael; Eder, Philipp: Datenanalyse, Abstimmung und Entwicklung, Springer Fachmedien Wiesbaden, 2017

Ulbrich, S.; Menzel, T.; Reschka, A.; Schuldt, F.; Maurer, M.: Begriffsdefinitionen für das automatisierte Fahren (2015)
Ulbrich, Simon; Menzel, Till; Reschka, Andreas; Schuldt, Fabian; Maurer, Markus: Definition der Begriffe Szene, Situation und Szenario für das automatisierte Fahren, 2015

Veček, N. et al.: A Comparison between Ratings for Ranking Evolutionary Algorithms (2014)
Veček, Niki; Črepinšek, Matej; Mernik, Marjan; Hrnčič, Dejan: A Comparison between Different Chess Rating Systems for Ranking Evolutionary Algorithms, in: 2014 Federated Conference on Computer Science and Information Systems, FedCSIS 2014, S. 511–518, 2014

Vinyals, O. et al.: AlphaStar: Mastering the Real-Time Strategy Game StarCraft II (2019)
Vinyals, Oriol; Babuschkin, Igor; Chung, Junyoung; Mathieu, Michael; Jaderberg, Max; Czarnecki, Wojtek; Dudzik, Andrew; Huang, Aja; Georgiev, Petko; Powell, Richard; Ewalds, Timo; Horgan, Dan; Kroiss, Manuel; Danihelka, Ivo; Agapiou, John; Oh, Junhyuk; Dalibard, Valentin; Choi, David; Sifre, Laurent; Sulsky, Yury; Vezhnevets, Sasha; Molloy, James; Cai, Trevor; Budden, David; Paine, Tom; Gulcehre, Caglar; Wang, Ziyu; Pfaff, Tobias; Pohlen, Toby; Yogatama, Dani; Cohen, Julia; McKinney, Katrina; Smith, Oliver; Schaul, Tom; Lillicrap, Timothy; Apps, Chris; Kavukcuoglu, Koray; Hassabis, Demis; Silver, David: AlphaStar: Mastering the Real-Time Strategy Game StarCraft II, 2019

Vinyals, O. et al.: Grandmaster level in StarCraft II using multi-agent RL (2019)
Vinyals, Oriol; Babuschkin, Igor; Czarnecki, Wojciech M.; Mathieu, Michaël; Dudzik, Andrew; Chung, Junyoung; Choi, David H.; Powell, Richard; Ewalds, Timo; Georgiev, Petko; Oh, Junhyuk; Horgan, Dan; Kroiss, Manuel; Danihelka, Ivo; Huang, Aja; Sifre, Laurent; Cai, Trevor; Agapiou, John P.; Jaderberg, Max; Vezhnevets, Alexander S.; Leblond, Rémi; Pohlen, Tobias; Dalibard, Valentin; Budden, David; Sulsky, Yury; Molloy, James; Paine, Tom L.; Gulcehre, Caglar; Wang, Ziyu; Pfaff, Tobias; Wu, Yuhuai; Ring, Roman; Yogatama, Dani; Wünsch, Dario; McKinney, Katrina; Smith, Oliver; Schaul, Tom; Lillicrap, Timothy; Kavukcuoglu, Koray; Hassabis, Demis; Apps, Chris; Silver, David: Grandmaster level in StarCraft II using multi-agent reinforcement learning, in: Nature, Vol. 575, S. 350–354, 2019

Wagener, N. et al.: An Online Learning Approach to Model Predictive Control (2019)
Wagener, Nolan; Cheng, Ching-An; Sacks, Jacob; Boots, Byron: An Online Learning Approach to Model Predictive Control, 2019

Wang, P. et al.: Driving behavior decision making based on a benefit evaluation model (2020)
Wang, Pengwei; Gao, Song; Li, Liang; Cheng, Shuo; Zhao, Hailan: Research on driving behavior decision making system of autonomous driving vehicle based on benefit evaluation model, in: AoT, Vol. 53, S. 21–36, 2020

Wei, J. et al.: A behavioral planning framework for autonomous driving (2014)
Wei, Junqing; Snider, Jarrod M.; Gu, Tianyu; Dolan, John M.; Litkouhi, Bakhtiar: „A behavioral planning framework for autonomous driving“, in: 2014 IEEE Intelligent Vehicles Symposium Proceedings, 2014

Weiss, T. et al.: DeepRacing: Parameterized Trajectories for Autonomous Racing (2020)
Weiss, Trent; Behl, Madhur: DeepRacing: Parameterized Trajectories for Autonomous Racing, 2020

Werling, M. et al.: Optimal trajectories for time-critical street scenarios (2012)
Werling, Moritz; Kammel, Sören; Ziegler, Julius; Gröll, Lutz: Optimal trajectories for time-critical street scenarios using discretized terminal manifolds, in: The International Journal of Robotics Research, Vol. 31, S. 346–359, 2012

Werling, M. et al.: Optimal trajectory generation for dynamic street scenarios (2010)
Werling, Moritz; Ziegler, Julius; Kammel, Sören; Thrun, Sebastian: „Optimal trajectory generation for dynamic street scenarios in a Frenét Frame“, in: 2010 IEEE International Conference on Robotics and Automation, 2010

Wiewiora, E.: Reward Shaping (2010)
Wiewiora, Eric: Reward Shaping, in: Encyclopedia of Machine Learning, S. 863–865, 2010

Winner, H.; Hakuli, S.; Lotz, F.; Singer, C.: Handbuch Fahrerassistenzsysteme (2015)
Winner, Hermann; Hakuli, Stephan; Lotz, Felix; Singer, Christina: Handbuch Fahrerassistenzsysteme: Grundlagen, Komponenten und Systeme für aktive Sicherheit und Komfort, ATZ/MTZ-Fachbuch, 3. Auflage, Springer Vieweg, 2015

Wymann, B.: Forenbeitrag zum Schadensmodell in TORCS (2012)
Wymann, Bernhard: TORCS - The Open Racing Car Simulator / Where damage is distributed. sourceforge.net, URL: https://sourceforge.net/p/torcs/mailman/message/29758884/, 2012, Zugriff 29. 07. 2021

Wymann, B.: Torcs FAQs (2012)
Wymann, Bernhard: Torcs FAQs, URL: http://torcs.sourceforge.net/?name=Sections&op=viewarticle&artid=30#c1_1, 2012, Zugriff 29. 07. 2021

Yang, Y. et al.: α^α-Rank: Practically Scaling α-Rank through Stochastic Optimisation (2020)
Yang, Yaodong; Tutunov, Rasul; Sakulwongtana, Phu; Ammar, Haitham Bou: α^α-Rank: Practically Scaling α-Rank through Stochastic Optimisation, in: arXiv:1909.11628 [cs], 2020

Yi, B.: Integrated Planning and Control for Collision Avoidance Systems (2018)
Yi, Boliang: Integrated Planning and Control for Collision Avoidance Systems, Schriftenreihe / Institut für Mess- und Regelungstechnik, Karlsruher Institut für Technologie, Vol. 39, KIT Scientific Publishing, 2018

Zhang, J.: Gradient Descent based Optimization Algorithms for Deep Learning (2019)
Zhang, Jiawei: Gradient Descent based Optimization Algorithms for Deep Learning Models Training, in: arXiv:1903.03614 [cs, stat], 2019

Zhou, M. et al.: Scalable Multi-Agent RL Training School for Autonomous Driving (2020)
Zhou, Ming; Luo, Jun; Villela, Julian; Yang, Yaodong; Rusu, David; Miao, Jiayu; Zhang, Weinan; Alban, Montgomery; Fadakar, Iman; Chen, Zheng; Huang, Aurora Chongxi; Wen, Ying; Hassanzadeh, Kimia; Graves, Daniel; Chen, Dong; Zhu, Zhengbang; Nguyen, Nhat; Elsayed, Mohamed; Shao, Kun; Ahilan, Sanjeevan; Zhang, Baokuan; Wu, Jiannan; Fu, Zhengang; Rezaee, Kasra; Yadmellat, Peyman; Rohani, Mohsen; Nieves, Nicolas Perez; Ni, Yihan; Banijamali, Seyedershad; Rivers, Alexander Cowen; Tian, Zheng; Palenicek, Daniel; Ammar, Haitham bou; Zhang, Hongbo; Liu, Wulong; Hao, Jianye; Wang, Jun: SMARTS: Scalable Multi-Agent Reinforcement Learning Training School for Autonomous Driving, S. 20, 2020

Ziegler, J. et al.: Navigating car-like robots using an obstacle sensitive cost function (2008)
Ziegler, Julius; Werling, Moritz: „Navigating car-like robots in unstructured environments using an obstacle sensitive cost function“, in: 2008 IEEE Intelligent Vehicles Symposium.Eindhoven University of Technology, Eindhoven, The Netherlands, June 4-6, 2008, 2008

Zubača, J. et al.: Smooth Reference Line Generation for a Race Track with Gates (2020)
Zubača, J.; Stolz, M.; Watzenig, D.: „Smooth Reference Line Generation for a Race Track with Gates based on Defined Borders“, in: 2020 IEEE Intelligent Vehicles Symposium (IV), 2020

Eigene Publikationen

Sippel, Marco; Winner, Hermann: Planning Trajectories Using an Extended Sequential Linearization Algorithm, ATZ Fachtagung Automatisiertes Fahren 2020, 13.-14.10.2020, Wiesbaden, 2020.

Sippel, Marco; Winner, Hermann: Vorrichtung zur Abschätzung einer Fahrzeit bis zu Zusammentreffen, Patent, *DE102020109875A1*, 2021.

Sippel, Marco; Winner, Hermann: Vorrichtung zur automatischen Ermittlung eines kritischen Zustands für ein Fahrzeug, Patent, *DE102020109878A1*, 2021.

Betreute Abschlussarbeiten

Miao, Fan: Entwicklung und Analyse von Systempartitionierungen im Kontext eines kommunikativen Assistenzsystems für die Rennstreckenanwendung.
Master-Thesis Nr. 724/18

Schaller, Maximilian: Entwicklung und Integration eines virtuellen Fahrermodells.
Bachelor-Thesis Nr. 1342/19

Berger, Jan: Methodische Entwicklung eines Trajektorien-Reglers für die Anwendung in einer Rennsimulation.
Master-Thesis Nr. 750/19

Scholl, Karl: Konzeptionierung und Konstruktion eines Lenkaktors für ein automatisiertes Formula Student Fahrzeug.
Bachelor-Thesis Nr. 1343/19

Fröhlich, Kevin: Konzeptionierung eine Antriebsstrangs for den Formula Student Electric Rennwagen des TU Darmstadt Racing Team e.V..
Bachelor-Thesis Nr. 1344/19

Kara, Mehdi: Entwicklung von Lernstrategien für Online Fahrzeugmodelle.
Master-Thesis Nr. 794/20