



Duale Hochschule Baden-Württemberg
Mannheim

Studienarbeit

**Entwicklung, Implementierung und Validierung eines
lokalen Pfadplanungsalgorithmus für den autonomen
Rennsport**

Studiengang Elektrotechnik

Studienrichtung Automation

Verfasser:	Finn Rasmus Schäfer
Matrikelnummer:	9523059
Kurs:	TEL19AT2
Studiengangsleiter:	Walther Berthold
Studienarbeitsbetreuerin:	Dr. Leila Mekacher

Kurzfassung

Die Vorliegende Arbeit beschäftigt sich mit einem Algorithmus für die Bewegungsplanung eines autonomen Formula Student Rennwagens. Sie stellt die Weiterentwicklung eines nicht optimierenden Ansatzes dar. Die Grundlage des entwickelten Algorithmus stellt dabei ein multidimensionaler gerichteter Graph dar. Über gewichtete Kostenfunktionen wird innerhalb der Arbeit auf einen möglichst krümmungsneutralen Pfad geschlossen. Dem Pfad wird über die Kombinatorik von maximalem, bremsendem und beschleunigendem Geschwindigkeitsprofil nach quasi-state-steady Ansatz ein resultierendes Geschwindigkeitsprofil zugewiesen wird. Der entwickelte Algorithmus wird über visuelle und fahrdynamische Simulation ausgewertet und anschließend bewertet. Ergebnis der Arbeit stellt einen verbesserten Algorithmus dar, dessen aktuellen Schwachstellen ebenfalls aufgezeigt werden.

Abstract

This thesis deals with an algorithm for motion planning of an autonomous Formula Student racing car. It represents the further development of a non-optimizing approach. The basis of the developed algorithm is a multidimensional directed graph. Within the work, weighted cost functions are used to infer a path that is as curvature-neutral as possible, to which a resulting velocity profile is assigned via the combinatorics of maximum, braking and accelerating velocity profiles according to a quasi-state-steady approach. The developed algorithm is evaluated via visual and vehicle dynamics simulation and then evaluated. Result of the work represents an improved algorithm whose current weaknesses are also pointed out.

Eigenständigkeitserklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Ort, Datum, Unterschrift: _____

Inhaltsverzeichnis

1. Einleitung	1
1.1. Motivation	3
1.2. Problemformulierung und Lösungsansatz	4
1.3. Gliederung	5
2. Theoretische Grundlagen	7
2.1. Grundlagen der Pfadplanung	7
2.2. Gerichtete Graphen	9
2.3. Kostenfunktionen	10
2.4. Geschwindigkeitsprofil	12
3. Konzept	15
3.1. Eingangs- und Ausgangsgrößen	15
3.2. Planning Normalen und Nodes	17
3.3. Kostenfunktion CM-22x	18
3.4. Optimierungsansatz	19
4. Implementierung	20
4.1. Aufbau und Ablauf eines Bewegungsplanungszyklus	20
4.2. Graph	21
4.3. Kostenfunktion	24
4.4. Pfad	26
4.5. Geschwindigkeitsprofil	26
5. Validation und Ergebnisse	29
5.1. Simulation	29
5.2. Simulationsdurchläufe	30
5.2.1. Hockenheimring	31
5.2.2. Melbourne	35
5.2.3. Berlin	38
5.3. Simulationsergebnisse	41
6. Zusammenfassung und Ausblick	42
A. Bildanhang Kapitel 4	47
B. Bildanhang Kapitel 5	54

Abkürzungsverzeichnis

AS Autonomous System

CV Combustion Vehicle

CURE Cooperative University Racecar Engineering

DNF Did Not Finish

DNS Did Not Start

EV Electrical Vehicle

EVA Electrical Vehicle Autonomous

FS Formula Student

FSG Formula Student Germany

LIDAR Light Detection and Ranging

Orga Organisation

ROS Robotic Operating System

RRT Rapidly-Exploring-Random-Tree

SLAM Simultaneous Localization and Mapping

TUM Technische Universität München

Abbildungsverzeichnis

1.	Abbildung des Track Layouts für die Skid Pad Disziplin	2
2.	Übersicht über das autonome Gesamtsystem und die zugehörigen Arbeitsbereiche des AS-Teams von Cure Mannheim e.V.	3
3.	CAD Modell des Fahrzeugs CM-22x ohne Darstellung der AS-spezifischen Sensoren	5
4.	Beispielhafte Darstellung eines Voronoi-Diagramms als Grundlage eines Pfadplanungsmodul	8
5.	Darstellung eines Potentialfelddiagramms, welches einen Beispielhaften Pfad durch verschiedene Hindernisse darstellt	8
6.	Bildbeispiel für einen in der Formula Student Anwendung findenden rapidly exploring random tree-Ansatzes.	9
7.	Darstellungsformen von Graphen. Der linke Teil der Abbildung zeigt einen gerichteten Graphen mit zwei Knoten und einer Kante. Der rechte Teil zeigt einen multidimensionalen Graphen mit drei Schichten, sechs Knoten und acht Kanten	10
8.	Beispielhafte Darstellung eines multidimensionalen gerichteten Graphen anhand eines zufälligen Streckenabschnitts	10
9.	Visualisierung einer linearen Regression mit Koeffizientenbestimmung durch Kostenfunktionen	13
10.	Symbolische Darstellung eines GGV-Diagramms, welches laterale und longitudinale Geschwindigkeitswerte kombiniert darstellt	14
11.	Aufgaben- und Ablaufübersicht über einen vollständigen AS-Zyklus . .	16
12.	Übersicht über die direkten Schnittstellen des Submoduls Motion Planning inklusive zugehöriger und übergebender Parameter	17
13.	Darstellung des Ablaufes der lokalen Trajektoriengenerierung für den fertigen Algorithmus	21
14.	Klassendiagramm des Geschriebenen Bewegungsplanungsalgorithmus zur Beispielhaften erklärung des Ablaufs	22
15.	Darstellung des genauer interpolierten Splines, welcher die Algorithmik verwendet, die zu ca. doppelter Berechnungszeit führt	24
16.	Berechneter Spline mit der ca. doppelt so schnellen Berechnungsmethode mittels numerischem Ansatz	24
17.	Darstellung der Implementierung der Planning Nodes am Beispiel einer Referenzkurve.	25
18.	Visualisierung der räumlichen Darstellung des implementierten geometrischen Graphen. In der Abbildung sind nur die Planning Nodes eingeblendet. Es ist eine variierte Start Node mit mehreren Endpunkten zu erkennen .	25
19.	Darstellung des implementierten geometrischen Graphen. In dieser Abbildung sind neben den Planning Nodes auch die Edges eingeblendet. Die Abbildung stellt die Grundlage für die Berechnung des kostengünstigsten Graphen dar	25
20.	Visualisierung der Kombinatorik des Geschwindigkeitsprofils	27

21.	Darstellung einer GGV-Map für den autonomen Rennwagen CM-22x von Cure Mannheim e.V.	27
22.	Abbildung der linearisierten GGV-Map auf der Basis der, in Abbildung 21 dargestellten, idealen GGV-Map für den Motor des Rennwagens CM-22x	27
23.	Darstellung einer Ideallinie mit Geschwindigkeitsprofil am Beispiel der ehemaligen Formel1 Strecke des Hockenheimrings mit Initial- und Endgeschwindigkeit von 0 Meter pro Sekunde	28
24.	Darstellung der Simulation zu Grunde liegenden Modelle und ihre Abhängigkeiten	30
25.	Darstellung eines Plots für die durch den in dieser Arbeit entwickelten Algorithmus zur Berechnung der Ideallinie anhand des Beispiels des Hockenheimrings	31
26.	Visualisierung des Geschwindigkeitsprofils in Abhängigkeit der errechneten Ideallinie am Beispiel des Hockenheimrings mit Initial- und Endgeschwindigkeit von 0 Metern pro Sekunde	32
27.	Visualisierung des ungeglätteten Pfades, der besonders in Kurven zu Welligkeit neigt	32
28.	Darstellung des angedeuteten Kurvenschneidens durch den berechneten idealen Pfad für den Hockenheimring	32
29.	Darstellen des Fahrzeugverhalten durch Regelung auf die errechnete Ideallinie am Beispiel des Hockenheim Rings	33
30.	Darstellung des lateralen Offset für ausgewählte Regelansätze für die Validierung der errechneten Ideallinie des Hockenheim Rings	33
31.	Gegenüberstellung einer geregelten Fahrzeugtrajektorie mit wenig lateraler Abweichung und einer Trajektorie mit großer lateraler Abweichung am Beispiel der für den Hockenheimring errechneten Ideallinien	34
32.	Darstellung der durch den Algorithmus berechneten idealen Trajektorie für die Formel1 Strecke von Melbourne, vor ihrem Umbau 2022	35
33.	Darstellung des Geschwindigkeitsprofils für die errechnete Ideallinie der Formel1 Strecke Melbourne mit Initial- und Endgeschwindigkeit von 0 Metern pro Sekunde	35
34.	Darstellung des zu glättenden Pfades für die Kurve 1 der Formel 1 Strecke Melbourne	35
35.	Darstellung des Fahrzeugverhaltens für zwei Regleransätze auf Basis der errechneten Trajektorie für die Formel1 Strecke Melbourne vor ihrem Umbau	36
36.	Darstellung des Simulierten Fahrzeugverlaufs für zwei beispielhafte Kurven der Formel1 Strecke Melbourne	36
37.	Darstellung der lateralen Abweichung im Vergleich von errechneter Trajektorie und geregeltem Fahrzeug für die Formel1 Strecke vor ihrem Umbau	37
38.	Darstellung der errechneten Ideallinie für die Rennstrecke von Berlin. Besonderes Augenmerk liegt auf der Welligkeit des Pfades	38
39.	Darstellung des durch die Welligkeit des Pfades beeinflussten Geschwindigkeitsprofil mit nahezu sprunghaften Geschwindigkeitswerten für die Rennstrecke in Berlin	39

40.	Visualisierung der simulierten Fahrdynamik für den Rennwagen CM-22x am Beispiel der errechneten Referenztrajektorie für die Rennstrecke in Berlin	39
41.	Darstellung des simulierten Kurvenverlaufs für den Rennwagen CM-22x für ausgewählte Streckenabschnitte auf der Rennstrecke Berlin. Zu sehen ist neben der Referenztrajektorie ebenfalls das Verhalten bei Regelung mit dem Standard Pure Pursuite Controller und einem von CURE weiterentwickelten Controller	40
42.	Darstellung der lateralen Regelabweichung bei errechneter Ideallinie für die Rennstrecke von Berlin	40

Tabellenverzeichnis

1.	Übersicht der vorhandenen Formula Student Driverless Disziplinen	4
2.	Beispielmatrix mit 4 Einträgen für das Verwenden der Normalengleichung zum Bestimmen der Gewichtungsfaktoren von Kostenfunktionen	12
3.	Darstellung der gemessenen Runtime Zeiten, bei gleichen Interpolierungsko- ordinaten. Dargestellt werden ausgewählte Testdaten zum Vergleich der Berechnungsschnelligkeit .	23

1. Einleitung

Im Rahmen des internationalen Konstruktionswettbewerbs, der Formula Student, erstellt der Verein Cooperative University Racecar Engineering (CURE) e.V. einen Rennwagen mit Elektroantrieb. Dieser Rennwagen (CM-22x, zu sehen in Abbildung 3), soll in der Lage sein, sowohl mit, als auch ohne Fahrer, bei den verschiedenen Disziplinen der weltweit stattfindenden Events teilzunehmen. Die Events finden in den verschiedenen Ländern einmal pro Saison statt. Es gibt unter anderem Events in Deutschland, Ungarn, Tschechien, aber auch im außereuropäischen Ausland zum Beispiel in Indien, Brasilien und weiteren Ländern [1]. Für jede Formula Student Saison muss ein neuer Rennwagen entwickelt werden. Dieser kann sowohl als Verbrenner (Combustion Vehicle (CV)), als auch mit Elektroantrieb (Electrical Vehicle (EV)) entwickelt werden. Die Einzelteile der Autos müssen von den Studenten selbst konstruiert, gefertigt und getestet werden. Für die Konstruktion, teilweise ebenfalls für die Tests, gibt es Regeln. Diese Regeln stellen hauptsächlich Konstruktionsvorschriften für die Entwicklung dar und dienen somit der Sicherheit, teilweise sind diese aber auch spezifisch für die Disziplinen der Formula Student. Die Regeln unterscheiden sich dabei von Event zu Event. Um diese Konstruktionsaufgabe zu stemmen, besitzt der Verein eine unternehmensähnliche Struktur. Diese Struktur besteht aus der Projektleitung-Technik, der Projektleitung-Organisation und den zugehörigen Sub-Teams Technik und Organisation (Orga). In der Technik gibt es folgende Sub-Teams:

- *Suspension*: Hauptaufgabe des Suspension Teams ist die Fahrzeugkinematik. Außerdem werden hier unter anderem die Bremsen, die Lenkung, die Radträger und das Feder-Dämpfer System entwickelt.
- *Chassis & Composites*: Dieses Sub-Team entwickelt den Rahmen und die Rahmenanbindungen, hierzu zählen unter anderem der Sitz, das Lenkrad, die den Fahrer schützende Crash-Box in der Nase des Wagens, sowie die Felgen.
- *Electronics*: Das Electronics-Team entwickelt die Platinen des Autos und verkabelt verschiedene Sensoren und Aktoren. Außerdem schließt das Team diese an den Can-Bus an. Die Anschlüsse sind dabei in drei verschiedene logische Bereiche gegliedert, die Peripherie, den Akku und den Motor. Das Electronics-Team ist außerdem für die Bremslichter, die Visualisierung der verschiedenen autonomen Stati und zum Beispiel den Kabelbaum verantwortlich.
- *Powertrain*: Zu Powertrain gehört vor allem der Motor, sowie dessen Anbringung. Außerdem beschäftigt sich das Team mit der Unterbringung der Electronics Platinen, des Getriebes und dessen Aufhängung, sowie die Kühlung ihrer Verantwortungsbereiche (Motor und Inverter).
- *Accumulator*: Hier wird sowohl der Hoch-, als auch der Niederspannungsakku entwickelt. Dabei ist der Hochvoltakku hauptsächlich für die Versorgung des Motors verantwortlich und besitzt 600 Volt. Der Niederspannungsakku ist für die Spannungsversorgung der verschiedenen Sensoren und Aktoren gedacht und besitzt eine

Spannung von 24 Volt. Teil der Akkuentwicklung ist außerdem das Sicherheitssystem, dazu gehören zum Beispiel das Brandmeldesystem und die Kühlung.

- *Aerodynamics:* In diesem Team werden alle aerodynamischen Teile des Rennwagens entwickelt. Dazu gehören sämtliche Flügelformen, die Nase und der Unterboden.
- *Autonomous System:* Das Autonomous System (AS) befasst sich mit allen nötigen Bereichen für das autonome Fahren, dazu zählt die Umgebungswahrnehmung, das Lokalisieren und Kartieren des Wagens und der Umgebung, die Pfadplanung und der damit verbundenen Regelung.

Diese Arbeit beschäftigt sich mit der Pfadoptimierung für das autonome Fahren und ist somit innerhalb des Projektes in das AS-Team einzuordnen. Durch die Einführung eines Hybriden Wettbewerbs in der Formula Student Germany (FSG) Saison 2022 ist die Wichtigkeit des AS-Teams gestiegen. Erstmals in der Historie eines Formula Student Events ist es nur möglich, die volle Eventpunktzahl zu erreichen, wenn zwei der normalen Events (mit Fahrer) autonom abgeschlossen werden. Diese beiden Dynamic-Disziplinen sind die Acceleration und das Skidpad. Bei der Acceleration muss der Rennwagen eine Distanz von 75 Metern schnellstmöglich absolvieren und dann innerhalb eines vorgegebenen Haltebereichs sicher zum Stehen kommen. Aus der gemessenen Zeit, auf die mögliche Strafen aufaddiert werden, wird die Gesamtpunktzahl errechnet [2]. Das Skidpad stellt eine liegende Acht dar (siehe Abbildung 1), dort werden nach dem Einfahren in den Kurs zwei Runden gestoppt. Aus dem Mittelwert der beiden gemessenen Runden, zuzüglich Strafen, erschließt sich nach Berechnungsvorschrift die Gesamtpunktzahl [3]. Für die FSG ergibt sich daraus ein möglicher Punktvorteil von 150 Punkten gegenüber Teams, die nicht in der Lage sind, Disziplinen autonom abzuschließen.

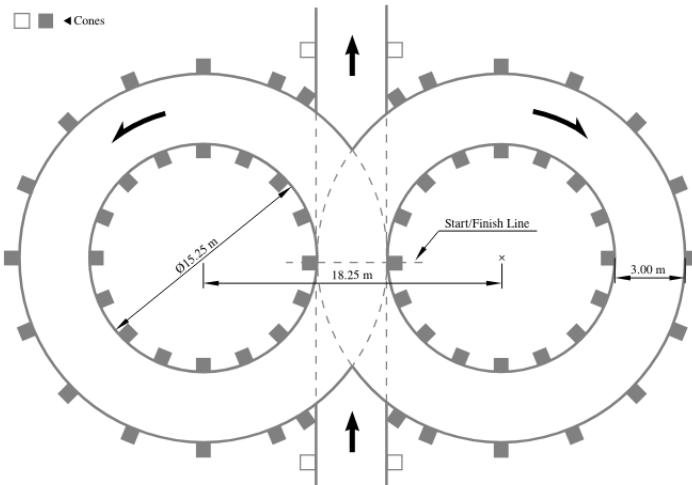


Figure 1: Abbildung des Track Layouts für die Skid Pad Disziplin Bildquelle: [3]

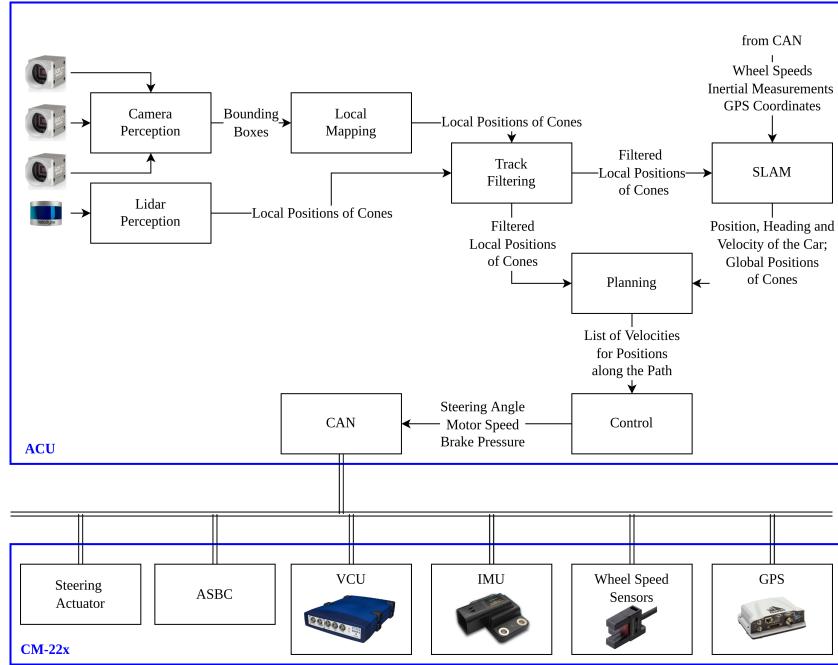


Figure 2: Übersicht über das autonome Gesamtsystem und die zugehörigen Arbeitsbereiche des AS-Teams von Cure Mannheim e.V. Bildquelle: Cure Mannheim e.V.

Damit der Rennwagen autonom durch den Kurs fahren kann, bedarf es auch im AS-Team einer weiteren Untergliederung. Dabei gibt es die oben bereits angesprochenen Bereiche der Umgebungswahrnehmung (Perception) mittels mehrerer Kameras und einem Light Detection and Ranging (LIDAR)-Sensor. Die Daten der Perception werden in der Lokalisation und Kartierung, dem Mapping aufgegriffen. Innerhalb des Mappings wird mit dem Simultaneous Localization and Mapping (SLAM)-Algorithmus eine globale Karte erstellt. Durch Verwendung des Algorithmus werden die einzelnen Fehler der Sensoren verrechnet und eine möglichst genaue Karte erstellt. Im Themenbereich Pfadplanung (Planning) liegt neben der Pfadoptimierung ebenfalls die Filterung der kartierten Strecke, das Track Filtering. Das Track Filtering soll die Informationen der Karte noch einmal so filtern, dass der Logik des Fahrzeuges danach bewusst ist, in welchen Bereichen es sich fortbewegen darf. Mit dem zweiten Teil des Plannings, der Pfadoptimierung, beschäftigt sich diese Studienarbeit. In der Pfadoptimierung muss eine Trajektorie geplant werden und an den letzten Themenschwerpunkt des AS-Team, der Regelung (Control), übergeben werden. Eine Visuelle Übersicht über das AS-Team ist der Abbildung 2 zu entnehmen. Auf die genauen Inhalte, sowie die genaue Aufgabe dieses Themenbereichs wird in den folgenden Kapiteln eingegangen.

1.1. Motivation

In der letztjährigen coronabedingten Doppelsaison 2020/21 entwickelte CURÉ den ersten autonomen Rennwagen der Vereinsgeschichte. Dieser Rennwagen war in der Lage alle

Disziplinen, abgesehen des Skid Pad, zu absolvieren. Durch den Grundgedanken „Make it work - Then make it better“ des AS-Teams war das Ziel der Saison ein autonom fahrendes Auto. In der diesjährigen Saison 2022 soll das letztjährige Konzept nun verbessert werden. Dazu gehören verschiedene Ansätze, wie zum Beispiel die Einführung eines LIDARs, der Wechsel von einer auf drei Kameras und der Wechsel von der letztjährigen Fahrzeugsteuerung zu einer Fahrzeugregelung. Das Hauptziel der aktuellen Saison ist das Abschließen aller Disziplinen und das Verbessern der Zeiten, die aus der letzten Saison bereits existieren:

Table 1: Übersicht der vorhandenen Formula Student Driverless Disziplinen [4]

Disziplin	Erklärung	Status
Accelaration	Track: Beschleunigung über 75 Meter, sicher stehen bleiben	finished (+Penalty): 15,347 s
Skid Pad	Track: liegende acht, siehe Abbildung 1	Did Not Start (DNS)
Autocross	Track: 200...500 Meter lange Runde	finished (+Penalty): 52,573 s
Trackdrive	zehn Runden Autocross Track	Did Not Finish (DNF)

Um bessere Zeiten erzielen zu können, muss vor allem der Bereich Planning überarbeitet werden. Hier liegt, zusammen mit dem Unterpunkt Control, das größte Potential gegenüber dem letztjährigen System. Dieses Verbesserungspotential wird in den Folgenden Kapiteln erschlossen und ausgearbeitet.

1.2. Problemformulierung und Lösungsansatz

Der Ausgangspunkt dieser Arbeit stellt die Pfadplanung des Rennwagens der Saison 2020/21 dar. Das Modell der letzten Saison, Electrical Vehicle Autonomous (EVA), besaß eine reine Pfadplanung. Über eine Deaunly Triangulation wurde die Mittellinie des vorliegenden Tracks ermittelt und an die Regelungseinheit des Wagens übergeben. Die Nachteile dieser Herangehensweise waren, dass der Pfad nicht auf die Fahrzeugparameter ausgelegt war. Der Rennwagen wurde demnach nie an der höchstmöglichen Belastungsgrenze gefahren. Außerdem stellte dieser Pfad, ohne Geschwindigkeitsinformationen, nicht das zeitliche Optimum dar. Dieses zeitliche Optimum ist jedoch besonders wichtig für die Bepunktung und somit die Platzierung bei den dynamischen Disziplinen. Um dies im diesjährigen Fahrzeug zu ändern, soll eine lokale Pfadoptimierung entwickelt und implementiert werden. Das Ziel ist die Entwicklung eines Algorithmus, der mit Hilfe von Kostenfunktionen den zeitlich optimierten Pfad mit Geschwindigkeit an die Regelungseinheit übergeben kann. Diese Bahnkurve soll als Trajektorie definiert werden. Die Trajektorie benötigt dabei immer eine Anfangs- und eine Endbedingung. Die Trajektorie und ihre Anfangs- und Endbedingungen sollen innerhalb dieser Arbeit ergründet werden. Die Planung soll über Knotenpunkte (Planning Nodes) auf Normalen der Mittellinie des Tracks erfolgen. Durch die Verbindung der einzelnen Planning Nodes entsteht ein gerichteter Graph. Die zeiteffizienteste Verbindung der Planning Nodes, die Raceline, soll über die Kostenfunktionen erschlossen werden. Durch den neuen Pfadop-



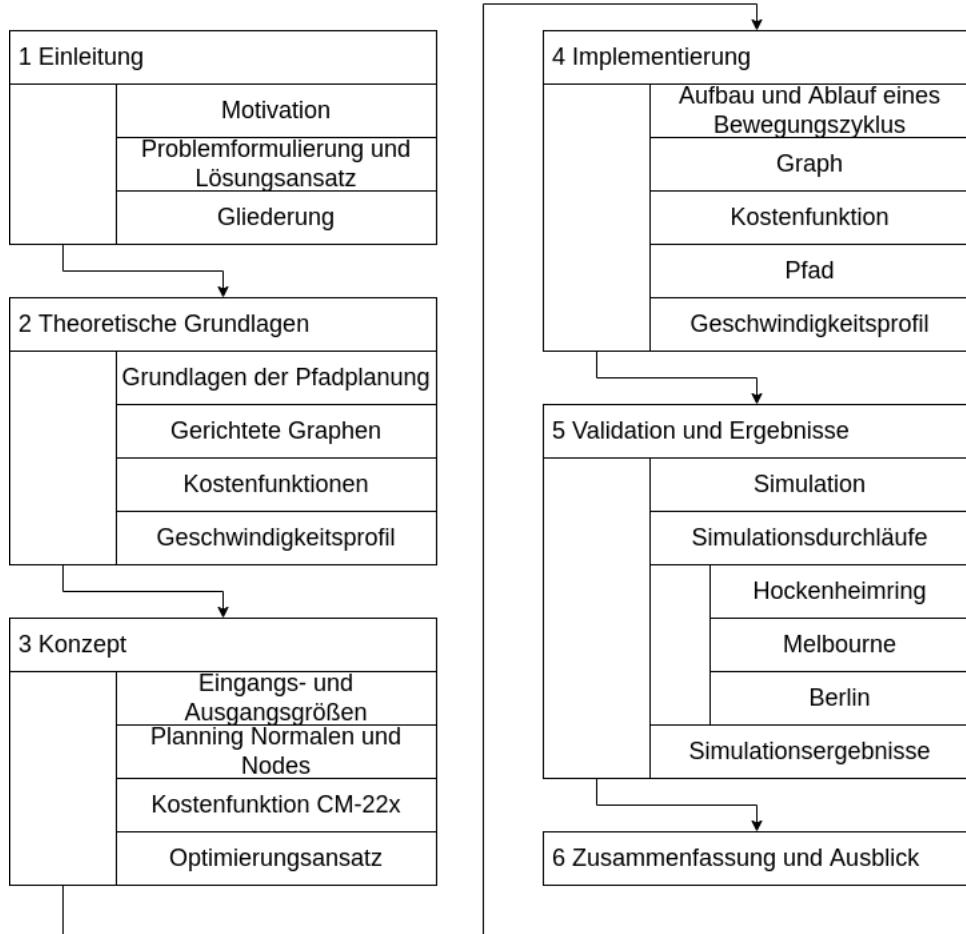
Figure 3: CAD Modell des Fahrzeuges CM-22x ohne Darstellung der AS-spezifischen Sensoren

timierungsansatz mittels Bewegungsplanung werden die Probleme von EVAs alter Pfadplanung behoben. Für diese Arbeit entehen damit folgende Problemstellungen:

- Die Erstellung einer Kostenfunktion mit Gütefaktor
- Das Bestimmen der Trajektorie
- Das Implementieren des Fahrzeugmodells für die zeiteffiziente Pfadoptimierung
- Die Validierung der Ergebnisse durch Simulation

1.3. Gliederung

Die Arbeit ist in sechs Kapitel unterteilt. Im Ersten Kapitel, der Einleitung wird auf die Grundlagen der Formula Student bzw. der Formula Student Driverless, sowie die grundlegende Themenstellung und Motivation eingegangen. Die Kapitel „Theoretische Grundlagen“ & „Konzept“ befassen sich mit der grundlegenden Theorie für den entwickelten Algorithmus. Während im Kapitel 2 auf die verschiedenen betrachteten Pfadplausionsmöglichkeiten, sowie gerichtete Graphen, Kostenfunktionen und die Grundlagen eines Geschwindigkeitsprofils eingegangen wird, wird im Kapitel „Konzept“ mit begründeten Quellennachweisen auf die Entscheidung für das diesjährige Konzept eingegangen. Im Kapitel vier „Implementierung“ wird vorgestellt auf welcher Basis das vorgestellte Projekt umgesetzt wurde. Hierbei wird auf den allgemeinen Ablauf eines Programmzykluses und die grundlegende Implementierung eingegangen. Das Kapitel fünf „Validation und Ergebnisse“ stellt die dieser Arbeit zu Grunde liegende SImulation vor. Dabei ist diese in eine visuelle und eine dynamische Simulation zu unterscheiden. Bei der visuellen Simulation wird allein die Trajektorie betrachtet und bewertet. Bei der



dynamischen Simulation wird auf das durch klassische Regler und ein implementiertes Fahrzeugmodell entstehende Fahrzugverhalten eingegangen. Die durchgeföhre Simulation sowie ihre Ergebnisse werden in den Unterkapiteln am Beispiel der Formel1 Strecke des Hockenheimrings, der Formel1 Strecke in Melbourne und einer Rennstrecke in Berlin dargestellt und Diskutiert. Im letzten Kapitel, dem Kapitel „Zusammenfassung und Ausblick“ werden noch einmal die Ergebnisse zusammengefasst und weitere Entwicklungsmöglichkeiten - so wie Ziele - für die diesjährige Saison ausgegeben.

2. Theoretische Grundlagen

Innerhalb dieses Kapitel wird auf die theoretischen Grundlagen für die diesjährige Bewegungsplanung eingegangen. Im ersten Teil des Kapitel werden die verschiedenen Pfadplanungsansätze betrachtet und begründet, weshalb sich für den gewählten Ansatz entschieden wurde. Im Anschluss daran wird auf die Eigenschaft des gerichteten Graphen eingegangen. Dieser wird als Basis für den im Konzept verwendeten Ansatz erklärt. Gegen Ende des Kapitels wird auf Kostenfunktionen eingegangen. Diese sollen dazu verwendet werden über Gewichtung bestimmter Parameter den für den Anwendungsfall am Besten passensten Ansatz zu verwenden. Abgeschlossen wird das Kapitel mit einen kurzen Einblick in die Erstellung eines Geschwindigkeitsprofils für einen Referenzpfad.

2.1. Grundlagen der Pfadplanung

Für die Pfadplanung gibt es verschiedene Ansätze. Im Folgenden wird ein Teil dieser Ansätze verglichen und erläutert, weshalb der gewählte Ansatz verwendet wurde. Die zu betrachtenden Ansätze sind dabei die Folgenden:

- Potentialfeld-Verfahren
- Voronoi-Diagramm
- Rapidly-Exploring-Random-Tree-Ansatz
- Multidimensionale Graphansätze

Die Potentialfeldmethode verwendet positives und negatives Potential. Dabei könnte das Auto ein Potential besitzen, dass vom Ziel angezogen wird und Hindernisse ein Potential, das den Rennwagen abstößt. Hierraus ergibt sich ein Gradientenfeld. Auf dieses Gradientenfeld kann dann ein Gitter gelegt werden, in dem auch Start und Zielpunkt eingetragen sind. Der Pfad von Start zu Ziel ergibt sich über die Unterscheidung der Gitter. Dieses Verfahren bietet folglich einen sicheren Pfad durch den Kurs, wird in der Regel aber nur zur Optimierung eines bereits vorhandenen Pfades verwendet [5], [6]. Da im Anwendungsfall der Formula Student kein Pfad bekannt ist, und die Rechenzeit und zugehörige Algorithmen klein gehalten werden soll, wird dieser Ansatz keine Anwendung finden.

Voronoi Diagramme stellen einen Mix aus zellbasierten und graphbasierten Planungsmethoden dar. Grundlage dieses Diagramms bzw. dessen Auswertung sind geometrische Algorithmen niedriger Komplexität. Das Diagramm nutzt die Voronoi Zellen, hierfür werden verschiedene geometrische Formen um einen Datensatz von Punkten (im Fall der Formula Student zum Beispiel Hüttchen) gespannt. Innerhalb einer solchen Zelle kann dann der kürzeste Weg in die nächste Zelle bestimmt werden [7]. Eine sehr ähnlicher Ansatz, der Ansatz der Delaunay Triangulation war die Grundlage der letzjährigen Pfadplaung [8]. Hierbei wurde jedoch kein idealer Pfad bestimmt, sondern die Mittellinie. Der entstandene Ansatz stellt dabei vor allem eine Verbindung des Kartieren und des Planens dar. Durch die Abgrenzung des Trackfiltering, dem Filtern der erstellten

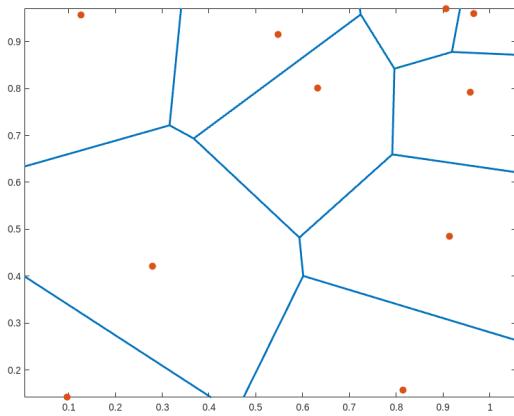


Figure 4: Beispielhafte Darstellung eines Voronoi-Diagramms als Grundlage eines Pfadplanungsmodul Bildquelle: Eigene Darstellung

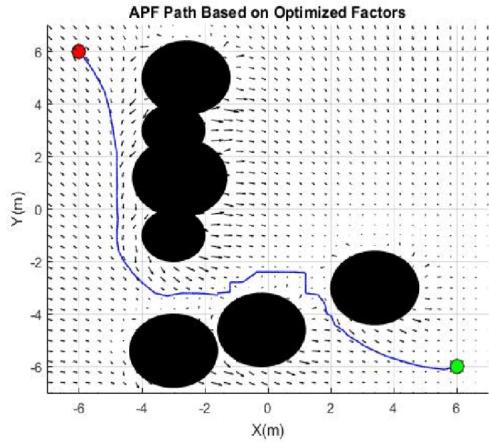


Figure 5: Darstellung eines Potentialfelddiagramms, welches einen beispielhaften Pfad durch verschiedene Hindernisse darstellt Bildquelle: ResearchGate

Karte und der Bewegungsplanung, aber auch durch die nicht gegebene Zeitoptimierung, die von enormer Bedeutung im autonomen Rennsport ist, wurde dieser Ansatz für die Saison 2022 verworfen.

Die vermutlich bekanntesten Pfadplandungs bzw. Pfadfindungsalgorithmen sind die Rapidly-Exploring-Random-Tree (RRT)-Ansätze. Bei diesen Algorithmen werden für eine bestimmte Zeit lang verschiedene Pfade generiert und dann der Pfad verwendet, der den Gütekriterien am ehesten entspricht. Ein Vorteile des RRT-Ansatzes ist die schnelle Zielfindung. Diese Zielfindung ist dabei in der Regel jedoch erst einmal nicht optimal. Mit steigender Laufzeit wächst die Wahrscheinlichkeit einen optimalen Pfad zu finden, da mit steigender Laufzeit mehr und mehr Zweige des Baumes den Zielpunkt erreichen. Je größer die zurückzulegende Strecke ist, desto größer wird automatisch auch die Laufzeit des Suchbaums. Die meisten RRT-Algorithmen dienen als Pfadplanungsmodul, da nie wirklich sicher gestellt werden kann, dass das Optimum gefunden wurde. Das bedeutet, der durch den Suchbaum gefundene Pfad wird nocheinmal, beispielsweise durch das Gradientenverfahren, optimiert [9], [10]. Gerade zu Beginn der Formula Student Driverless wurden vor allem RRT-Algorithmen verwendet [11], [12], [13]. Seit der Zunahme von implementierten SLAM Algorithmen und denen sich daraus ergebenden globalen Lösungsmöglichkeiten, ist die Beliebtheit jedoch zurückgegangen. Aus einzelnen Analysen, beispielhaft des DHBW Engineering [14], wird genauer auf diese Thematik eingegangen. Auf Grundlage der Analysen wurde der RRT-Ansatz ebenfalls verworfen.

Durch das in den letzten Jahren wachsende Interesse an autonomen Rennsport, unter anderem durch das Roborace und die Indy Autonomous Challenge, sind verschiedene Forschungsansätze entwickelt worden. Diese Ansätze basieren zum Teil auf gerichteten

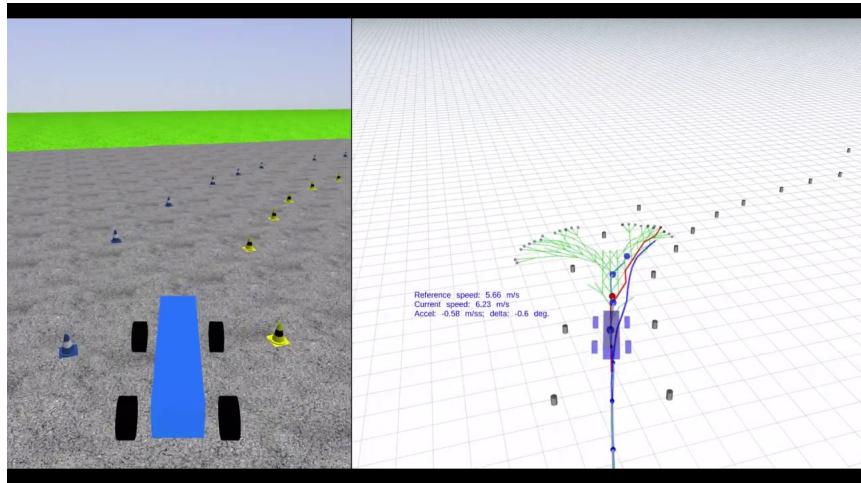


Figure 6: Bildbeispiel für einen in der Formula Student Anwendung findenden rapidly exploring random tree-Ansatzes. Bildquelle: Maxim Yastremsky

multidimensionalen Graphen mit Kostenfunktionen. Im Verein Cure Mannheim e.V. wurde auf Basis der Erfolge dieser Algorithmen beschlossen die Implementierung dieses offen zugänglichen Forschungsansatzes auf den Formula Student Anwendungsfall anzupassen. Um das Konzept, welches im Kapitel 3 Konzept festgehalten ist, vollständig verstehen zu können, wird nun auf die zugehörigen theoretischen Grundlagen eingegangen.

2.2. Gerichtete Graphen

Graphen dienen der Abstraktion hauptsächlich technischer Probleme. Knoten und Kanten werden dazu verwendet Informationen logisch miteinander zu verknüpfen und somit eine Struktur für die Problemstellung zu schaffen. Ein Knoten stellt dabei i.d.R. eine Information oder einen Zustand dar. Durch die Kanten können Zustandsübergänge oder Verbindungen dargestellt werden. Wird ein Graph aufgespannt bedeutet das, dass verschiedene Nodes mittels Kanten verbunden werden. Ist ein Graph gerichtet, sind die Kanten mit Richtungen versehen und können somit als Pfeile interpretiert werden. Der Zustand kann sich dann nur in die durch die Kante gegebene Richtung ändern. Beispielhaft ist dies zu sehen in Abbildung 7. Dort kann sich vom Zustand der *Node A* nur in den Zustand der *Node B* bewegen werden. Ein multidimensionaler Graph besitzt mehrere Schichten. In jeder Schicht kann es dabei beliebig viele Knoten geben die durch beliebig viele Kanten verbunden sein können (siehe Abbildung 7). Graphen lassen sich mathematisch gut beschreiben und dienen oftmals als Grundlage für verschiedene Algorithmen. Sie ermöglichen das endliche Darstellen eines Zustandsraumes und bieten verschiedene Ansätze zum effektiven Verarbeiten von Daten. In der Pfadplanung werden Graphen vor allem innerhalb der RRT-Ansätze verwendet, bei CURE sollen sie mittels eines anderen Ansatzes, der in Kapitel 3 Konzept vorgestellt wird verwendet werden. Der in dieser Arbeit verwendete gerichtete Graph stellt einen geometrischen gerichteten



Figure 7: Darstellungsformen von Graphen. Der linke Teil der Abbildung zeigt einen gerichteten Graphen mit zwei Knoten und einer Kante. Der rechte Teil zeigt einen multidimensionalen Graphen mit drei Schichten, sechs Knoten und acht Kanten

Graphen dar. Dieser wird verwendet um sinnvolle räumliche Beziehungen herstellen zu können. Der geometrisch gerichtete Graph kann dabei dazu verwendet werden, die Bewegungsrichtung innerhalb des Graphbaums vorzugeben. Bei einseitig gerichteter Verbindung durch eine Kante/Edge, kann sich innerhalb des Baumes auch nur in diese karthesische Richtung bewegen werden. Die Edges können somit auch Teil der Geometrie des Graphen sein und mehr als nur eine logische Verbindung darstellen. Über die Ausrichtungsinformationen, welche beispielsweise aus der Tangente abgeleitet werden können, kann eine Edge interpoliert werden und eine möglichst realistische räumliche Verbindung der Nodes darstellen. Durch die Verbindung von räumlicher und graph-basierter Darstellung können weitaus mehr Informationen über die Verbindungen und Verbindungsmöglichkeiten generiert, gespeichert und logisch verknüpft werden. Eine Verdeutlichung der eben vorgestellten Thematik ist Abbildung 8 zu entnehmen. Da sich in dieser Arbeit keiner weiteren Graphalgorithmen als dieser bedacht wird, wird diese hier auch nicht weiter betrachtet.

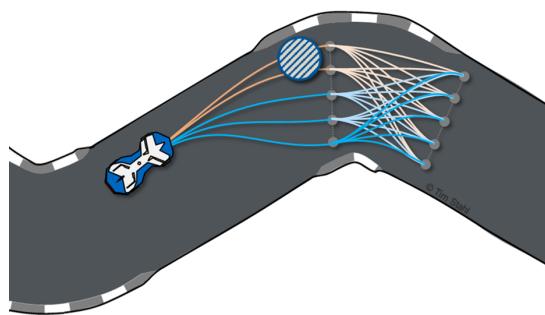


Figure 8: Beispielhafte Darstellung eines multidimensionalen gerichteten Graphen anhand eines zufälligen Streckenabschnitts. Bildquelle: TUM Motorsports

2.3. Kostenfunktionen

Kostenfunktionen mit Gütefaktor sind Funktionen, die über bestimmte Parameter den Kosten zu einem Datensatz berechnen. In der Technik werden diese Parameter in der Regel über den Ansatz des Maschinellen Lernens bestimmt. Um eine Kostenfunktion

aufzustellen, werden verschiedene Testdaten aufgenommen und die eingehenden Variablen gewichtet addiert. Um Kostenfunktionen beispielhaft zu erklären wird im Folgenden auf das Beispiel der linearen Regression nach [15] eingegangen. In [15] wird zum bestimmen der Kostenfunktion zu allererst eine Hypothese aufgestellt, im Beispiel der linearen Regression ist dies die Folgende lineare Funktion:

$$h_{\omega}(x) = \omega_1 x + \omega_0 \quad (1)$$

In dieser Gleichung sind die Parameter ω_0 , was dem Anfangswert der linearen Funktion entspricht und ω_1 , der Steigung der Funktion. Als Kosten wird der Abstand zum eigentlichen Wert der bestimmt. Die aufgestellte Gleichung sieht demnach wie folgt aus:

$$c(\omega_0, \omega_1) = \frac{1}{2m} \sum_{i=1}^m (h_{\omega}(x^{(i)}) - y^{(i)})^2 \quad (2)$$

Ziel ist es die Summe $c(\omega_0, \omega_1)$ zu minimieren und folglich eine lineare Funktion zu erlangen, die in Summe den kleinsten Abstand zu den gegebenen Daten besitzt. Da sich die Funktion je nach Parameterwahl jeder Zeit verändert und die Abstände zu den einzelnen Datenpunkten damit immer verschieden sind, müssen die Faktoren ω_0 und ω_1 genau gewählt werden. Dies kann entweder durch raten geschehen, was eine sehr unge nauie Methode darstellt, oder aber durch den Ansatz des Maschinellen Lernens (beispielhaft durch das Gradientenverfahren oder aber durch Lösen der Normalgleichung). Das Gradientenverfahren zum Finden von Parametern muss dabei jedoch nicht immer zum Ziel führen. Gerade bei steigender Anzahl von Variablen kann es bei falscher Wahl der Lernrate geschehen, dass das Gradientenverfahren in einem lokalen Maxima oder Minima stagniert. Bei zu großer Lernrate ist es dabei auch möglich, dass das Verfahren nie in einem Maxima oder Minima endet, sondern divergiert. Bei zu kleiner Lernrate, kann es geschehen, dass sich die Funktion nur asymptotisch an den idealwert annähert, da er die Schritte nach Berechnungsvorschrift zu klein werden. Die vereinfachte Berechnungsvorschrift für einen gesuchten Parameter sieht nach [15] wie folgt aus:

$$\omega_1 := \omega_1 - \alpha \frac{d}{d\omega_1} c(\omega_1) \quad (3)$$

In dieser Gleichung stellt α die bereits angesprochene Lernrate dar. Alternativ zum Gradientenverfahren, kann auch das analytische Vorgehen mittels Normalengleichung verwendet werden. Um die Normalengleichung auf ein Kostenproblem anzuwenden bedarf es aber an strukturierten Testdaten. Daraus wird aus allen so genannten Features eine Matrix erstellt, im oben stehenden Beispiel der linearen Regression wären das zwei x Zustände ($x_0 \hat{=} 1$ und x_1 geht aus den Testdaten hervor). Die daraus resultierende Matrix X ist eine $m \times 2$ -Matrix (mit $m = \text{Anzahl Testdaten}$). Der Ausgang wird durch einen m -dimensionalen Vektor \vec{y} beschrieben

Table 2: Beispielmatrix mit 4 Einträgen für das Verwenden der Normalengleichung zum Bestimmen der Gewichtungsfaktoren von Kostenfunktionen

x_0	x_1	y
1	250	70
1	300	74
1	275	69
1	290	81

$$X = \begin{bmatrix} 1 & 2 \\ 1 & 3 \\ 1 & 5 \\ 1 & 7 \end{bmatrix}, \quad y = \begin{pmatrix} 2 \\ 3 \\ 5 \\ 4 \end{pmatrix}$$

Die Kostenvariable ω für die Funktion der linearen Regression ($c(\omega) = a\omega + b$) berechnet sich nach [15] mit folgendem Schema:

$$\omega = (X^T X)^{-1} X^T y \quad (4)$$

Eine Berechnung mittels Matlab liefert:

$$\omega = \begin{pmatrix} 1,6271 \\ 0,4407 \end{pmatrix}$$

Und den darausfolgenden Plot in Abbildung 9. Dabei spiegelt der blaue Graph die lineare Regression mit den in Matlab bestimmten Parametern wieder und der rote Graph eine lineare Regression mit ähnlich gewählten Parametern. Es ist deutlich zu erkennen, dass der rote Graph in der Summe eine größere Abweichung zu den gegebenen schwarzen Punkten besitzt, als der blaue Graph.

Sowohl für die lineare Regression, als auch Kostenfunktionen an sich, können dabei jeder Zeit mehr als ein Parameter verwendet werden. Es ist ebenfalls eine Gewichtung einzelner Parameter möglich. Je nach Datenlage wird für die Parameterbestimmung entweder das Gradientenverfahren oder die Normalengleichung verwendet. Nach [15] wird das Gradientenverfahren vor allem verwendet, wenn es viele Parameter n (im englischen auch als Feature bekannt) gibt. Die Definition von viele ist hierbei im Größenbereich $\geq 10^6$. Die Normalengleichung wird verwendet, wenn die Anzahl der Parameter im Vergleich relativ gering ist (*ca.* < 1000). Dies hat den Hintergrund, dass die Berechnungsdauer des Ausdrucks $(X^T X)^{-1}$ mit steigender Anzahl n ebenfalls steigt (Komplexität $O(n^3)$).

2.4. Geschwindigkeitsprofil

Das Geschwindigkeitsprofil vollendet die Trajektoriendefinition. Durch Zuweisen von Geschwindigkeiten für spezifische Kurvenpunkte entsteht dabei eine Raum-Zeit-Kurve,

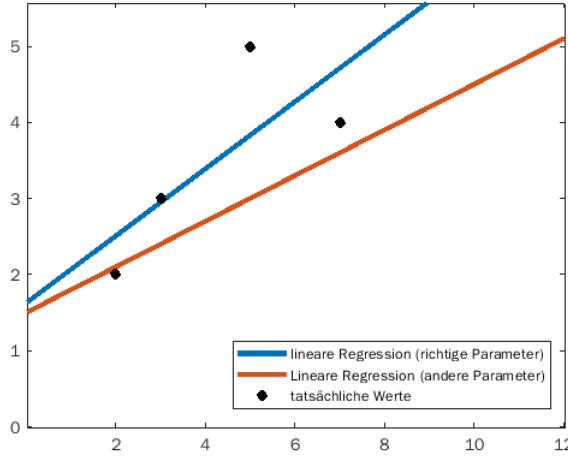


Figure 9: Visualisierung einer linearen Regression mit Koeffizientenbestimmung durch Kostenfunktionen

eine Trajektorie also. Ein Geschwindigkeitsprofil kann dabei verschieden aufgestellt werden. Es soll dazu beitragen die Rundenzeit weiterhin zu minimieren. Auf Basis der Fahrzeugparameter soll der ideale räumliche Verlauf durch bestmögliche Geschwindigkeit nocheinmal zeitlich optimiert werden. Grundlage des Geschwindigkeitsprofils ist ein gegebener Pfad, in der Regel die Ideallinie, sowie eine so genannte GGV-Map (siehe Abbildung 10). Dieses Diagramm beschreibt die Kombination der maximalen lateralen und longitudinal möglichen Beschleunigungen. Diese Beschreibung erfolgt dabei über den gesamten Arbeitsbereich des Fahrzeugs. Durch diese Modellierung wird das, im Rennsport wichtige Herausbeschleunigen und Einbremsen an den Belastungsgrenzen des Fahrzeuges in Kurven ermöglicht. Wie in [8] beschrieben, lässt sich ein resultierendes Geschwindigkeitsprofil über die Kombinatorik von Maximalgeschwindigkeit, Start- und Endgeschwindigkeit beschreiben. Das Geschwindigkeitsprofil für die Startgeschwindigkeit stellt dabei das Profil für die Beschleunigung dar. Das Geschwindigkeitsprofil, welches die Endgeschwindigkeit berücksichtigt, stellt das nötige Bremsprofil, das der negative Beschleunigung, dar. Die Maximalgeschwindigkeit für einen bestimmten Punkt lässt sich dabei aus der GGV-Map gegebenen maximalen Querbeschleunigung bestimmen. Grundlage dafür ist ein *quasi-state-steady* Ansatz nach [16]. Dieser Ansatz setzt voraus, dass die Änderungen zeitlich gesehen so gering sind, dass sie als konstant angenommen werden können. Die Maximalgeschwindigkeit in einem Scheitelpunkt, weil dort nur Querbeschleunigung auftritt, lässt sich beispielsweise wie folgt bestimmen:

$$v_{max} = \sqrt{\frac{a_{y,max}}{\rho}} \quad (5)$$

Anhand der vorhandenen Scheitelpunktgeschwindigkeiten können dann die restlichen Punkte abgeleitet werden. Die Berechnung dieser Geschwindigkeiten ist [8] und [17] zu entnehmen.

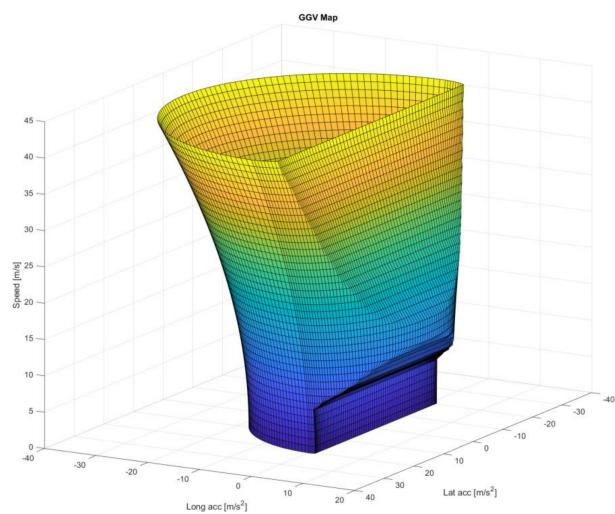


Figure 10: Symbolische Darstellung eines GGV-Diagramms, welches laterale und longitudinale Geschwindigkeitswerte kombiniert darstellt Bildquelle: Andrei-Cristian Pridie

3. Konzept

Das Konzept der diesjährigen Bewegungsplanung basiert auf den implementierten Forschungsergebnissen von Technische Universität München (TUM) Motorsport. Über mehrere Jahre hinweg wurden verschiedene Ansätze verfolgt und der im Folgenden vorgestellte vereinfacht. Die zugehörigen Dokumentationen befinden sich in [18], [19], [20], [21] und [22]. Der von der TUM entwickelte Ansatz wurde dabei in mehreren autonomen Rennsportklassen - unter anderem dem Roborace und der Indy Autonomous Challenge - verwendet. Da sich die Anwendungsfälle der Formula Student und dieser Rennklassen stark unterscheiden, wird im Folgenden der für Cure entwickelte Ansatz vorgestellt. Im Gegensatz zum eigentlichen Anwendungsbereich der TUM ist bei der Formula Student nicht zu jedem Zeitpunkt die gesamte Strecke bekannt. Daher muss für den Usecase der Formula Student ein Ansatz für die Referenzlinie geschaffen werden. Für den Wagen CM-22x soll dies die interpolierte Mittelline des bereits erkannten Tracks sein. Die Mittellinie wird dabei in den vorrangehenden Submodulen des autonomen Systems bestimmt und an den Planning-Algorithmus übergeben. Die Übergabe erfolgt dabei in Form von Punkten, um den Datentransfer so gering wie möglich zu halten. Nach der Interpolierung wird die Mittelline in den Frénet-Space transferiert. Dieser spannt die Raumkurve anhand der Kurventangenten und Normalen auf. Auf die Normalen, im Weiteren Verlauf als Planning Normalen bezeichnet, werden Punkte gesetzt. Diese Punkte werden von nun an als Planning Nodes bezeichnet. Diese Planning Nodes dienen als Knotenpunkte für einen gerichteten multidimensionalen Graphen. Dieser Graph soll dazu verwendet werden, die kostengünstigste Verbindung von Start- und Endpunkt zu bestimmen. Die kostengünstigste Verbindung soll in der Folge in eine zeitlich optimierte Trajektorie umgerechnet werden. Im letzten Schritt wird ein Geschwindigkeitsprofil für die vorliegende Trajektorie bestimmt und die einzelnen Raumpunkte und zugehörigen Geschwindigkeiten an die Regelung des Fahrzeuges übergeben. Das genaue Vorgehen wird dabei in den folgenden Unterkapiteln beschrieben.

3.1. Eingangs- und Ausgangsgrößen

Die Eingangsgrößen für das Submodul Planning und folglich auch die modellbasierte Pfadplanung sind die gefilterten Streckendaten aus dem Bereich Trackfiltering. Das Submodul bekommt die, durch die Kameras und den Lidarsensor aufgenommenen Daten. Diese werden dann durch den SLAM-Algorithmus in eine Karte verrechnet. Ausgangsgröße des Algorithmus ist die gefiltert reale Strecke. Nach dieser Filterung wird die Mittellinie (in Punktform) und die rechte und linke Streckenbegrenzung (ebenfalls in Punktform) über das Robotic Operating System (ROS) zur Verfügung gestellt. Eine variable Eingangsgröße stellen weiterhin Daten des Mapping bereit. Über die ROS-Mapping Node sollen dabei die aktuelle Geschwindigkeit und die aktuelle Position übermittelt werden. Die Position ist dabei wichtig für das Setzen der Start Position. Die Geschwindigkeit ist wiederum wichtig für das Berechnen des Geschwindigkeitsprofils mit Anfangswert. Aufgabe des Plannings ist dann, aus diesen Informationen die bereits beleuchtete Trajektorie mit Geschwindigkeitsprofil zu erstellen.

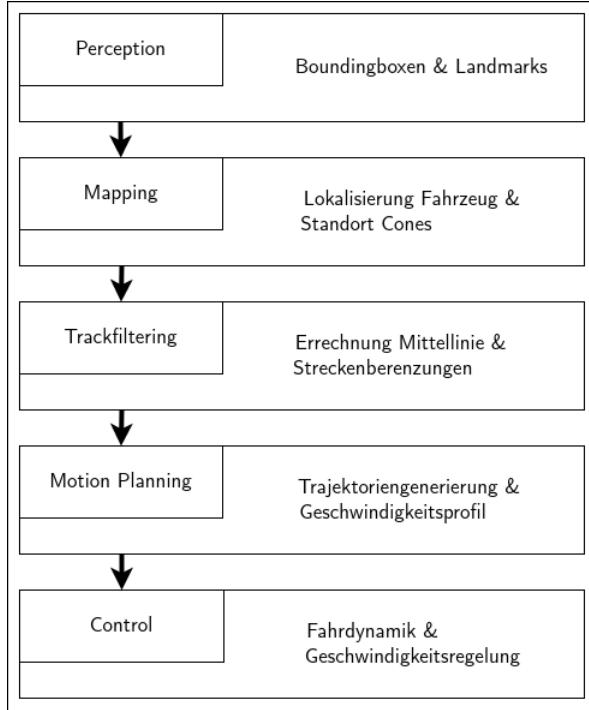


Figure 11: Aufgaben- und Ablaufübersicht über einen vollständigen AS-Zyklus

Ausgangsgröße des Algorithmuses ist eine Trajektorie. Diese Trajektorie wird punktweise übergeben. Jeder Punkt beinhaltet dabei nicht nur die karthesischen Koordinaten, sondern auch die diesem Punkt zugeordnete Geschwindigkeit. Da der Motion Planning Code ebenfalls eine ROS-Node darstellt, erfolgt auch hier die Übergabe mittels ROS. Die Motion Planning Node publischt dabei den individuell erstellten Datentyp an den Subscriber, die Regelungseinheit. Für den ersten Stand der Saison 2022 wird ein klassisches Regelungssystem bestehend aus einem Fahrdynamikregler und einem Geschwindigkeitsregler angenommen. Die karthesischen Koordinaten entsprechen demnach der Eingangsgröße des Fahrdynamikreglers und die Geschwindigkeitsangabe entspricht der Eingangsgröße des Geschwindigkeitsreglers. Die Regelung stellt das letzte Glied im Prozessfluss des autonomen Systems bei CURE dar. Bildlich dargestellt ist der Prozessfluss in Abbildung 11. Die direkten Schnittstellen und die zugehörigen physikalischen Größen sind in Abbildung 12 zu sehen. Für die Pfadplanung soll es keine direkte Variierungsmöglichkeit der Parameter im User Interface geben. Bei Bedarf sollen lediglich verschiedene Logging Informationen ausgegeben werden können. Welche dieser Informationen das genau sein werden, soll einheitlich im AS-Team zu einem späteren Zeitpunkt der Saison festgelegt werden. Es sollen jedoch keine Parameter kurzfristig variiert werden können, diese sollen durch ausgiebige Tests validiert werden und zum Zeitpunkt der Events dem Ideal entsprechen. Grundlage dieser Entscheidung ist vor allen Dingen die statische Disziplin des Engineering Design, in dem unter anderem genau diese Validierung und nicht Variierung als Grundlage für gute Bepunktung dienen.

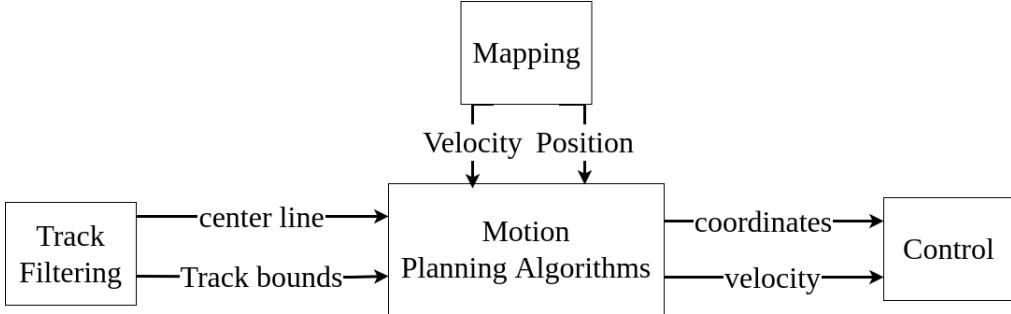


Figure 12: Übersicht über die direkten Schnittstellen des Submoduls Motion Planning inklusive zugehöriger und übergebender Parameter

3.2. Planning Normalen und Nodes

Wie in der Einleitung des Kapitels bereits beschrieben, sind die Planning Normalen die Normalen der Referenzlinie (der Mittellinie des Tracks). Hierfür wird eine Spline-Interpolierung durchgeführt. Aus den interpolierten Punkten wird über die Berechnungsvorschrift in (6) und (7) die Tangente und die Normale in Abhängigkeit des Ortsvektors \vec{r} bestimmt.

$$\vec{T} = \frac{1}{|\vec{r}|} \cdot \vec{r} \quad (6)$$

$$\vec{N} = \frac{1}{|\vec{T}|} \cdot \vec{T} \quad (7)$$

Für den Bewegungsplanungsalgorithmus werden die Planning Normalen in einem festen Punktabstand gesetzt, dies hat zur Folge, dass die Abstände auf einer Geraden (einem Kurvenabschnitt mit vergleichsweise geringer Krümmung) größer sind, als die Abstände in einer, oder mehreren Kurven (Kurvenabschnitt mit vergleichsweise größerer Krümmung). Über die Planning Nodes (Punkte auf den Normalen) wird dann, nach der Bestimmung des optimalen Graphen, die zeitoptimale Trajektorie gebildet. Die Planning Nodes berechnen sich aus der Berechnungsvorschrift in (8). Mit der Berechnungsvorschrift und dem entsprechenden Nodefaktor kann jede Node in Abhängigkeit von der Bogenlänge s der Kurve errechnet werden [18].

$$\vec{p}(s, k_{Nodefaktor}) = \vec{r}(s) + k_{Nodefaktor} \cdot \vec{N}(s) \quad (8)$$

In jeder Node sollen die Informationen über die Verbindung zur vorherigen Node gespeichert werden. Dazu gehören die karthesischen Koordinaten der Node, die Ausrichtung des Fahrzeugs im Endpunkt, sowie die Krümmung der Verbindung. Erfasst werden diese Zustandsdaten in einem Zustandsvektor $x(s) = [x(s), y(s), \theta(s), \kappa(s)]$. Die Berechnung der Ausrichtung θ ist in Gleichung 9 dargestellt und berechnet sich aus den Tangentenvektoren im jeweiligen Punkt. Die Krümmung berechnet sich nach [23] wie folgt:

$$\theta = \arctan\left(\frac{y}{x}\right) - \frac{\pi}{2} \quad (9)$$

$$\kappa = \frac{\dot{x}\ddot{y} - \ddot{x}\dot{y}}{(\dot{x}^2 + \dot{y}^2)^{\frac{3}{2}}} \quad (10)$$

3.3. Kostenfunktion CM-22x

Die Kostenfunktion basiert auf dem gerichteten multidimensionalen Graphen, der als Knotenpunkte die Planning Nodes verwendet. Über die gewichtete Kostenfunktion sollen dabei die durch den Usecase gegebenen Parameter ausgewertet werden. Daraus kann der kostengünstigste Graph erschlossen werden. Der Ansatz aus [18] bezieht sich dabei auf den autonomen Rennsport mit Überholmanövern. Die Umgebung ist vollkommen bekannt und stellt eine bekannte Formel- oder IndyCar-Rennstrecke dar. Die Parameter für den Kosten einer Node nach [18] (bzw. der Verbindung zweier Nodes), ist dabei die linear gewichtete Summe der Multiplikation der Abschnittslänge (s mit dem Gewichtungsfaktor ω_{len}) und der Summe der Durchschnittskrümmung (mit dem Gewichtungskoeffizienten ω_{range}), dem Krümmungsbereich (mit dem Gewichtungsfaktor ω_{κ_d}) und der lateralen Verschiebung ($d_{\Delta_{lat}}$ mit dem Gewichtungsfaktor ω_{offset}), zur gegebenen Ideallinie. Mathematisch errechnet sich der Kosten wie folgt:

$$cost = s(\omega_{len} + \omega_{\kappa_d} \bar{\kappa}^2 + \omega_{range} \Delta \kappa^2 + \omega_{offset} |d_{\Delta_{lat}}|) \quad (11)$$

Diese Kostenfunktion ist vor allem für autonome Überholvorgänge gedacht, da in der Regel der Ideallinie gefolgt wird. Eine Abweichung zur vorher gegebenen Ideallinie entsteht nur, wenn diese durch ein dynamisches Objekt, beispielsweise einen anderen Rennwagen, blockiert ist. Die Funktion ist damit darauf ausgelegt möglichst schnell wieder auf den idealen Pfad zurückzukehren. Im Fall der Formula Student ergeben sich zwei große Unterschiede. Zum einen ist die gegebene Referenzlinie nicht die Ideallinie, folglich soll auch nicht möglichst schnell auf die Referenzlinie zurückgekehrt werden. Zum Anderen erfolgt durch die unbekannte Strecke kein Wissen über den Besten Endzustand des Fahrzeugs. Die Konsequenz aus dieser Erkenntnis ist, dass die Kostenfunktion zum einen in der Parameterwahl, als auch in der allgemeinen Summe geändert werden muss. Eine zusätzliche Bedingung, mit dem entsprechendem Gewichtungsfaktor, ist eine extra Endbedingung. Am Ende jeder lokal geplanten Trajektorie soll das Fahrzeug in der Lage sein, die Kurve mit dem geringsten Kurvenradius zu fahren. Außerdem muss innerhalb der Simulation des Algorithmus ermittelt werden, ob eine Parameteränderung der gleichbleibenden Krümmungskosten durchgeführt werden muss. Die Erweiterung der Kostenfunktion soll dabei wie folgt aussehen:

$$cost = s(\omega_{len} + \omega_{\kappa_d} \bar{\kappa}^2 + \omega_{range} \Delta \kappa^2 + \omega_{offset} |d_{\Delta_{lat}}| + \omega_{Endbedingung} * c_{end}) \quad (12)$$

Für die Bestimmung des kostengünstigsten Graphen muss die Summenbildung über die einzeln berechneten Kosten gebildet werden. Um eine solche Summe bilden zu können, muss eine Bezeichnungsvorschrift für den Node-Inhalt eingeführt werden. So soll der Kosten im allgemeinen als c bezeichnet werden und zwei Indizes (i, j) besitzen. Die

beiden Indizes sollen dabei einmal für den Layer und einmal für den Nodeindex selbst stehen. Eine Node $c_{i,j} = c_{1,2}$ entspräche demnach dem Kosten der 2. Node auf dem ersten Layer, wobei ein Layer einer Normalen entspricht. Für den Kosten eines einzelnen Graphen (m) mit den Nodekosten c_n ergibt sich die nachfolgende Summe:

$$cost_m = \sum_{n=0}^N c_n \quad (13)$$

Aus allen berechneten Graphen ($m = 0 \dots \text{Anzahl Graphen}$) kann dann auf den minimalen Kosten geschlossen werden. Der kostengünstigste Graph entspricht bei richtiger Parameterwahl der Kostengewichtungsparameter auch der Ideallinie. Der Graph soll sich durch möglichst geringe Krümmungsänderungen auszeichnen. Dabei ist jedoch nicht nur der gewichtete Mittelwert zu berücksichtigen, sondern ebenfalls die Spanne, in der sich die Krümmung bewegt. Große Krümmungsänderungen stehen dabei nämlich in direktem Bezug zu großen Geschwindigkeitsänderungen.

3.4. Optimierungsansatz

Aus Vergleichen unterschiedlicher Formula Student Teams geht hervor, dass eine zeitliche Optimierung der Rundenzeit i.d.R. mit einer Krümmungsoptimierung einhergeht [14], [8]. Ziel einer Krümmungsoptimierung ist das Konstanthalten der Krümmung, da diese in direkter Verbindung mit der Geschwindigkeit steht. Große Krümmungsänderungen, die zum Beispiel bei der Optimierung auf den kürzesten Weg entstehen, haben drastische Geschwindigkeitsänderungen und somit auch langsameren Rundenzeiten zur Folge. Für den diesjährigen Ansatz soll dabei ein Mittlweg gewählt werden. Durch den Pfad, der sich aus dem kostengünstigsten Graphen ergibt, soll ein möglichst konstantes und ideales Geschwindigkeitsprofil erstellt werden. Durch Gewichtung der bereits im vorangehenden Abschnitt beleuchteten Parameter, ist ein möglichst krümmungskonstanter Pfad zu erwarten. Aus einem Pfad mit konstanter Krümmung lässt sich dann wiederum ein möglichst konstantes und ideales Geschwindigkeitsprofil ableiten. Dieses Geschwindigkeitsprofil ist ermöglicht das Umsetzen der Manöver des Hineinbremsen und Herausbeschleunigen aus einer Kurve. Auf Basis dieser Überlegungen wird eine möglichst effiziente Rundenzeit erwartet. Durch die Bestimmung des kostengünstigsten Graphen wird demnach kein direkter nach zum Beispiel [5] definierter Optimierungsansatz verwendet. Es wurde sich bewusst gegen eine solche Herangehensweise entschieden, um einen sicheren Fortschritt im Bereich der Pfadplanung bzw. der Bewegungsplanung zu erreichen und dem im Kapitel 1 angesprochenem Motto des AS-Teams zu folgen.

4. Implementierung

Innerhalb dieses Kapitels wird genauer auf die Details der Implementierung eingegangen. Zu Beginn des Kapitels wird eine Übersicht über den allgemeinen Ablauf des Algorithmus und seiner Funktionen gegeben. In der Folge wird sich mit der Implementierung der einzelnen Schritte beginnend mit dem vorgestellten Graphansatz befasst. In der Folge wird auf das Einbinden der Pfadfindung und der Kostenfunktionen eingegangen. Zum Abschluss des Kapitels wird die Implementierung des Geschwindigkeitsprofils thematisiert. Die Algorithmik wurde in der Programmiersprache Python verfasst. Diese Sprache dient ebenfalls als Grundlage für das Gesamtsystem. Die Implementierung bedient sich dennoch verschiedener Klassen, die C++ Bibliotheken verwenden. Eine dieser Bibliotheken ist *igraph*. Diese stellt verschiedene Funktionen für Graphen der Pfadplanung bereit. Sie lässt sich dabei leicht um eigene Graphentypen erweitern. Als Grundlage dient dabei eine selbstgeschriebene Basisklasse. Auch für das Finden des kostengünstigsten Pfades stellt die Bibliothek eine Klasse bereit. Die Kostenfunktionen und das Geschwindigkeitsprofil bedienen sich keiner Klassen der *igraph* Bibliothek.

4.1. Aufbau und Ablauf eines Bewegungsplanungszyklus

Ein Zyklus der Bewegungsplanung besteht aus dem Verarbeiten der Input Daten zu einer an Control weitergebbaren Trajektorie. Dabei muss unterschieden werden zwischen erstmaligem Durchlaufen des Codes, da hier der Graph von Grund auf neu implementiert werden muss und den Fall, dass bereits eine Trajektorie und somit auch ein Graph existiert. Für den Fall des ersten Durchlaufens des Algorithmus, beginnt der Zyklus mit dem Initialisieren aller wichtigen Objekte. Dazu zählt das Laden der Kostenparameter und dem Dictionary, welches alle Pfade zu wichtigen Dateien enthält. Aber auch das Initialisieren der *Motion_planning_helpers* Klasse (im Folgenden als Helpers bezeichnet). Im ersten Schritt wird über die *center2spline*-Funktion der Helpers Klasse die Mittellinie interpoliert sowie die Krümmung, Normalen, Tangenten und die Ausrichtung in den Punkten errechnet. Diese Informationen stellen die grundlegenden vor allem aber wichtigsten Informationen für die Graphfunktionen dar. Über die Funktion *normals2layer* werden die errechneten Daten umgeformt, um diese sinnvoll an den Graphen übergeben zu können. In der Folge der Umstrukturierung wird auch der Graph erstellt. Anhand der errechneten Parameter werden dem Graphen Nodes und Layer hinzugefügt, sowie die Startposition und Ausrichtung festgelegt. Der Rechenintensivste und essentiellste Teil des Algorithmus beginnt mit dem Aufspannen des Edge bzw. Node Skeleton. Gerade bei längeren Streckenverläufen steigt die Laufzeit des Algorithmus exponentiell, da die Anzahl der Edges ebenfalls exponentiell zur Länge des Tracks steigt. Nach dem Bestimmen des grundlegenden Skeletts des Graphen, werden die Einzelkosten der Edges bestimmt. Vor der Berechnung des Geschwindigkeitsprofils wird über die *igraph*-Funktion *get_shortest_paths* der kostengünstigste Pfad bestimmt. Die kathesischen Koordinaten dienen als Eingangsgröße für die Bestimmung des Geschwindigkeitsprofils. Der Algorithmus übermittelt die errechnete Trajektorie an Control und startet mit den neuen Input Daten von vorn. Ab dem zweiten Durchlauf kann aber, da bereits

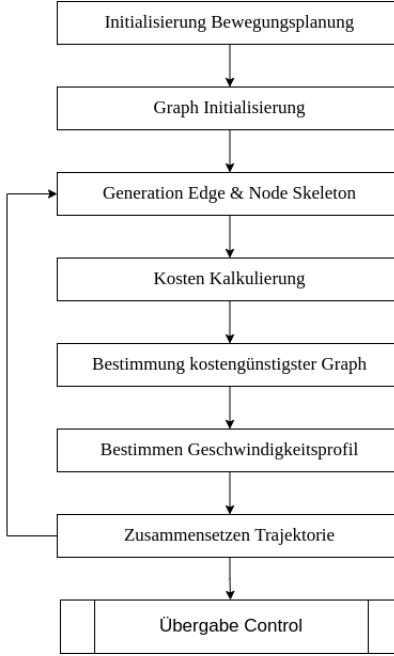


Figure 13: Darstellung des Ablaufes der lokalen Trajektoriengenerierung für den fertigen Algorithmus

ein Graph existiert, auf das erneute Initialisieren des Graphen verzichtet und gleich mit dem Erstellen des neuen Skeleton begonnen werden. In den Abbildungen 13 & 14 sind Ablauf und vorhandene Funktionen in Darstellung des Klassen- und des vereinfachten Ablaufdiagramms dargestellt.

4.2. Graph

Für das Verwenden der bereits vorgestellten Graphansätze, muss die Bibliothek *igraph* um eine Basisklasse, folglich um einen eigenen Graphtypen, erweitert werden. Innerhalb des Klassenkonstruktors kann dabei die Grundlage geschaffen werden. Eingangsgöße des Graphen sind die karthesischen Koordinaten die durch das Aufspannen der Planning Normalen in einer Hilfsbibliothek entstehen (zu sehen in Abbildung 17). Bei der ersten Implementierung des Graphen wird dieser lediglich als gerichtet definiert. Der Graph besteht aus Layern, Nodes und Edges. Die Nodes beinhalten dabei neben den karthesischen Koordinaten außerdem noch die Ausrichtung im jeweiligen karthesischen Punkt. Die Edges beinhalten neben Start und End Node, Start und End Layer, die Splinekoordinaten eben dieser Verbindung und falls bereits bestimmt, auch die Kosten für die jeweilige Verbindung (die Edges sind in Abbildung 19 dargestellt). Durch Aufsummieren der einzelnen Edgekosten kann dann auf die Kosten einzelner Baumzweige geschlossen werden. Die Splinekoordinaten der Edgeverbindungen werden mit einer größeren Schrittbreite als die Interpolierung der Mittellinie vorgenommen. Gerade, weil die Genauigkeit der Interpolierung der Mittellinie wichtig für die Planning Normalen und somit auch den letztendlichen Pfad sind, wird hier eine engere Schrittbreite verwendet.

<i>motion_planning</i>	<i>Motion_planning_helpers</i>
config: ConfigParser path_directory: dict w_curv_avg: float w_curv_peak: float w_length: float w_offset: float	add_full_layer(graph, layer, layer_pos, psi): None center2spline(points, factor, node_dist): np.array normals2layer(node_ctr, layer, normals, psi): np.array
<i>Velocity_profile</i>	<i>GraphBase</i>
Trajectory : np.array Curvature: np.array Velocity_accel: np.array Velocity_break: np.array Velocity_max: np.array Velocity: np.array a_x_max: float a_y_max: float v_start: float v_end: float v_max: float	layer: int node_points_layer: dict add_edge(start_layer, start_node, end_layer, end_node): None add_full_layer(graph, layer, layer_pos, psi): None add_node(layer, node_number, pos, psi): None add_nodes2layer(layer, layer_pos, nodes_pro_layer, psi): None gen_edge_splines(graph, start_layer, start_node, end_layer, \ end_node, psi_start, psi_end, step_size): np.array get_edge(start_layer, start_node, end_layer, end_node): tuple get_edges(): list get_layer_info(layer): tuple get_nodes(pos): list get_node_info(layer, node_number): list plot_graph(graph, bestx, besty, nodes_best): None remove_edge(start_layer, start_node, end_layer, end_node): None search_graph(start_node, end_node, max_solutions): tuple set_start_pos(pos_s, psi, vel_s): None

Figure 14: Klassendiagramm des Geschriebenen Bewegungsplanungsalgorithmus zur Beispielhaften erklärung des Ablaufs

Die engere Schrittbreite ist genauer als die der vielen Edges, dafür aber auch zeitintensiver. Da die Edges nicht den finalen Pfad darstellen und lediglich zur Kostenbestimmung verwendet werden sollen, ist hier ein Trade off zwischen Genauigkeit und Zeitaufwand zu tollerieren. Die numerische Interpolation der Edges ist durch die gewählte Schrittbreite dabei in etwa doppelt so schnell, wie das Interpolieren der Mittellinie (siehe Tabelle 3).

Table 3: Darstellung der gemessenen Runtime Zeiten, bei gleichen Interpolierungscoordinaten. Dargestellt werden ausgewählte Testdaten zum Vergleich der Berechnungsschnelligkeit

Mittellinie	Graph-Edges
0.003666 s	0.002021 s
0.002023 s	0.001155 s
0.003991 s	0.001734 s
0.002077 s	0.001555 s

Die in der Tabelle dargestellten Werte dienen jedoch nicht als qualitativer Richtwert für die Berechnungszeit im Fahrzeug, da die vorliegende Auslastung, aber auch die vorhandene Computing Power des Fahrzeugs, zum Zeitpunkt der Implementierung nicht nachgestellt werden kann. Der qualitative Unterschied ist vor allem bei Splines mit großer Krümmung zu erkennen. Dies ist beispielhaft in Abbildung 15 & 16 dargestellt. Die leichten Verformungen im markierten Bereich könnten beim Aufspannen der Normale bereits zu größeren Abweichungen im Pfad führen.

Ein Layer beinhaltet, beim aktuellen Stand der Implementierung, in der Regel 12 Nodes. Alle folgenden Definitionen, wie das Hinzufügen von Layern, Nodes oder aber Edges wird über Funktionen implementiert. Die Funktionen lassen sich in fünf Kategorien unterscheiden:

- Planning Nodes
- Edges
- Layer
- Graph Search
- Plotting

Die Methoden, die im Zusammenhang mit den Planning Nodes stehen, dienen vor allem verschiedenen Implementierungsmethoden - zum Beispiel die Unterscheidung zwischen dem Hinzufügen einer einzelnen Node und vieler Verschiedener - aber auch um Informationen über spezifische oder aber alle Nodes zu bekommen. Dies soll während der Testphase auf dem neun Fahrzeug CM-22x vor allem zum Loggen von Informationen genutzt werden. Das Gleiche gilt für die Edges und Layer. Hier dienen die Funktionen ebenfalls

nur für die Implementierung neuer Elemente und zur Datengenerierung. Bei den Graph Search und Plotting Funktionen ist dies nicht der Fall. Innerhalb der Graph Search Funktionen wird der kostengünstigste Pfad über die gespeicherten Kosteninformationen in den Edges bestimmt. Das Bestimmen des kostengünstigsten Graphen wird dabei über eine bereits in *igraph* vorhandene Funktion realisiert. Die Funktion *get_shortest_paths()* nimmt die Start Node und die End Node und für den Gewichtungsfaktor die Berechneten Kosten der Edges entgegen. Ausgabe der Funktion sind die karthesischen Koordinaten der ermittelten Raumkurve sowie die zugehörigen Nodes. Für diesen Pfad wird in der Folge ein Geschwindigkeitsprofil erstellt. Die sich daraus ergebende Trajektorie soll dann mittels ROS an Control übergeben werden. Das Plotting dient als visuelle Vorstufe der Simulation, hier wird der gerichtete Graph im zweidimensionalen Raum visualisiert. Beispielhaft ist dies in Abbildung 18 zu sehen. Mittels verschiedener Testdaten soll hier innerhalb der Testphase eine visuelle Überprüfung der Trajektorie durchgeführt werden. Dabei soll vor allem die Form und die Position der Trajektorie überprüft werden. Diese Art von Test stellt einen Zusatzsicherheitsfaktor dar, der durch Unit- und Systemtests im Laufe der Saison erweitert werden soll. Detailliertere Informationen und Plots sind dem Kapitel 5 zu entnehmen.

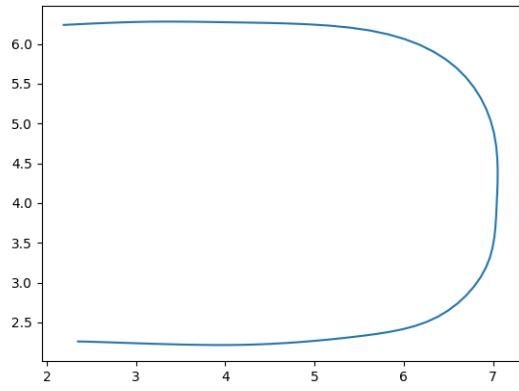


Figure 15: Darstellung des genauer interpolierten Splines, welcher die Algorithmik verwendet, die zu ca. doppelter Berechnungszeit führt

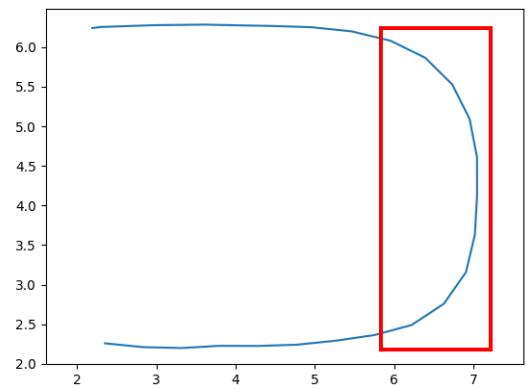


Figure 16: Berechneter Spline mit der ca. doppelt so schnellen Berechnungsmethode mittels numerischem Ansatz

4.3. Kostenfunktion

Ausgangspunkt der Kostenfunktion, bzw. die daraus resultierenden Kostenfunktionsparameter, sind die von der TUM bestimmten Größen. Innerhalb der Testphase bzw. der ausgiebigen Simulation und dem Testen auf dem neuen Rennwagen, sollen diese Parameter weiter angepasst werden. Die Funktionen sollen bei der ersten Initialisierung

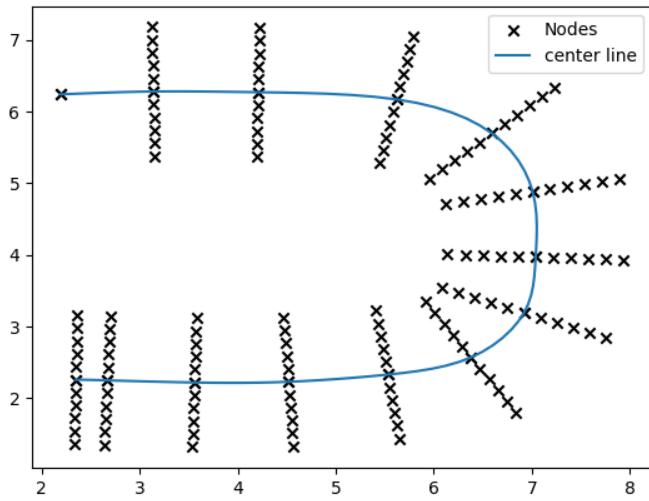


Figure 17: Darstellung der Implementierung der Planning Nodes am Beispiel einer Referenzkurve.

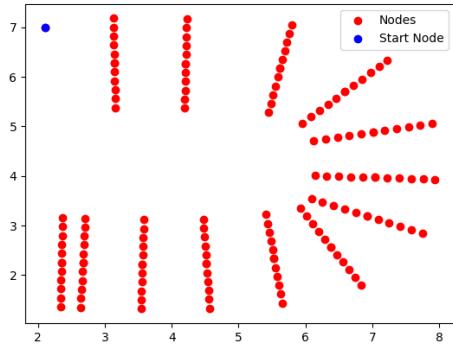


Figure 18: Visualisierung der räumlichen Darstellung des implementierten geometrischen Graphen. In der Abbildung sind nur die Planning Nodes eingeblendet. Es ist eine variierte Start Node mit mehreren Endpunkten zu erkennen

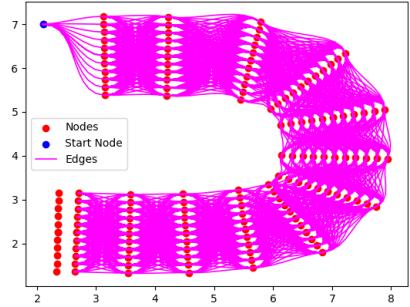


Figure 19: Darstellung des implementierten geometrischen Graphen. In dieser Abbildung sind neben den Planning Nodes auch die Edges eingeblendet. Die Abbildung stellt die Grundlage für die Berechnung des kostengünstigsten Graphen dar

des Codes mittels eines *.ini*-File eingebunden werden. Innerhalb dieser Datei sind die im Kapitel 3.3 aufgeführten Parameter definiert. Durch die Implementierung innerhalb einer *.ini* Datei sind die Parameter nach Neubestimmung leicht zu Ändern. Dabei ist

es jedoch nicht möglich, die Parameter händisch durch das User Interface am Fahrzeug zu ändern. Außerdem ist das Initialisieren mittels ConfigParser-Klasse einfach umzusetzen. Die Berechnung der verschiedenen Kosten erfolgt dabei über die in den Edges gespeicherten Informationen. Dafür wird für jede Edge unter anderem die effektive Länge berechnet, um die im Grundlagenkapitel beschriebenen Kosten zu normieren. Das Ergebnis der Kostenberechnung stellt einen einfachen float Wert dar. Dieser Floatwert kann durch die Aufsummierung verschiedener gerichtet verbundener Edges zu einem Baum zusammengesetzt werden. Die Bestimmung der genauen Kostenparameter ist als Teil des Testing für die neue Saison angedacht.

4.4. Pfad

Als Grundlage der Implementierung dient die Klasse *motion_planning_helpers*. Innerhalb dieser Klasse wird unter anderem die Mittellinie des Tracks interpoliert. Über die Interpolation werden mit einem Abstandsfaktor neue Punkte bestimmt. Mit Hilfe der neuberechneten Punkte kann dann die Tangente und die Normale nach den Gleichungen 6 & 7 berechnet werden. Der Faktor für den Abstand der Nodes (in Gleichung 8 als $\kappa_{Nodefaktor}$ bezeichnet) und der Faktor für den Abstand der Planning Normalen kann dabei variabel festgelegt werden. Die Wahl der beiden Faktoren soll dabei durch die Simulation validiert werden. Diese Faktoren haben direkten Einfluss auf die Laufzeit und somit auch auf die Publish Time des Algorithmus und sollen demnach so gering wie möglich gehalten werden. Für die Bestimmung des kostengünstigsten Pfades wurde eine Funktion geschrieben, die den in Abhängigkeit der Edge-Länge gewichteten Kosten bestimmt. Die Kosten werden dabei in den Edges gespeichert. Die letztendliche Pfadauswahl wird über Angabe der Gewichtung durch die eine Funktion der Klasse *igraph* ausgeführt. Der in Summe günstigste Zweig des Baumes wird dabei als kostengünstiger Pfad bestimmt. Dieser bestimmte Pfad dient als Eingangsgröße zur Bestimmung des Geschwindigkeitsprofils und somit zum Zusammensetzen der Trajektorie.

4.5. Geschwindigkeitsprofil

Am in der letzten Saison geschaffenen, jedoch noch nicht verwendeten Geschwindigkeitsprofils, wurden zu dieser Saison hauptsächlich Parameteränderungen durchgeführt. Für das Geschwindigkeitsprofil dient eine vereinfachte GGV-Map des Rennwagens CM-22x. Aus Betrachtungen der letzten Saison und einer bereits vorliegenden GGV-Map für den diesjährigen Motor (siehe Abbildung 21), kann auf eine linearisierte zylindrische GGV-Map mit einem Durchmesser von 2g geschlossen werden (siehe Abbildung 22). Die gegebene Maximalgeschwindigkeit wurde durch Modellierung und Simulierung des diesjährigen Fahrzeugs auf circa 30,5 Meter pro Sekunde geschlossen. Das Geschwindigkeitsprofil lässt sich dabei variabel implementieren. Das bedeutet, dass das Vorgeben einer variablen Start- und Endgeschwindigkeit möglich ist. Ein beispielhaftes, kombiniertes Geschwindigkeitsprofil ist in Abbildung 23 dargestellt. In Abbildung 20 ist die Kombinatorik des beschleunigenden, bremsenden und maximal möglichen Geschwindigkeitsprofils, sowie die Wechselwirkung von Krümmung und

Geschwindigkeit visualisiert. Das resultierende Geschwindigkeitsprofil ergibt sich dabei aus der Überlagerung der im Grundlagenteil vorgestellten Geschwindigkeitsprofile. In der Abbildung setzt sich das bremsende Geschwindigkeitsprofil (orange), das beschleunigende Geschwindigkeitsprofil (grün), das maximale Geschwindigkeitsprofil (blau) und das resultierende Geschwindigkeitsprofil (rot) zu erkennen.

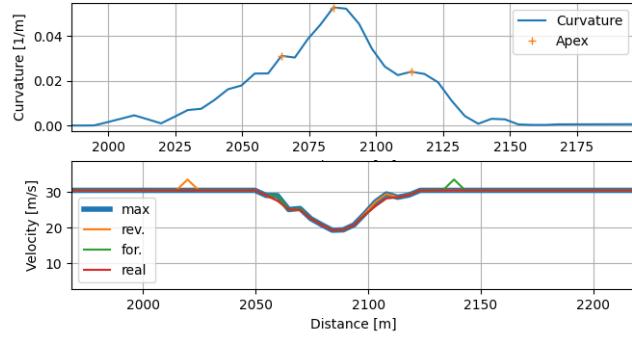


Figure 20: Visualisierung der Kombinatorik des Geschwindigkeitsprofils

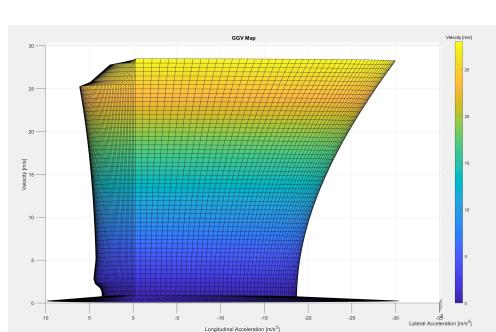


Figure 21: Darstellung einer GGV-Map für den autonomen Rennwagen CM-22x von Cure Mannheim e.V. Bildquelle: Cure Mannheim

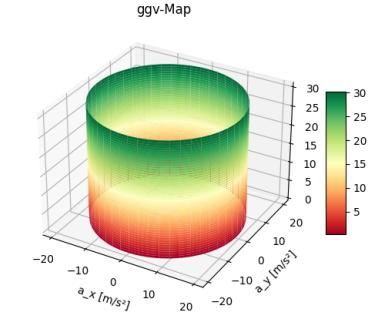


Figure 22: Abbildung der linearisierten GGV-Map auf der Basis der, in Abbildung 21 dargestellten, idealen GGV-Map für den Motor des Rennwagens CM-22x

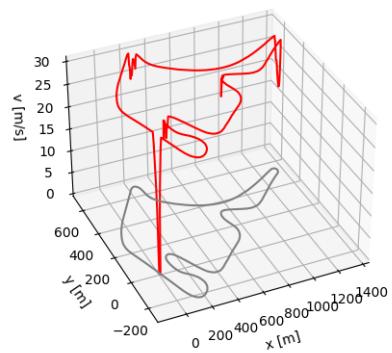


Figure 23: Darstellung einer Ideallinie mit Geschwindigkeitsprofil am Beispiel der ehemaligen Formel1 Stecke des Hockenheimrings mit Initial- und Endgeschwindigkeit von 0 Meter pro Sekunde

5. Validation und Ergebnisse

In diesem Kapitel wird die Herangehensweise für die Validation der Ergebnisse vorgestellt. Grundlage für die softwareseitige Validation ist dabei eine gemeinsame Simulation des Submoduls Planning und Control. Da diese sehr eng miteinander verbunden sind und eine so große Abhängigkeit im Sinne des Gesamtprojektes besitzen, werden diese zusammen simuliert. Die Simulation soll dabei ebenfalls eine Grundlage für das Auslegen einer modellprädiktiven Regelung im Anschluss an diese Studienarbeit darstellen. Da die Regelung jedoch kein Teil dieser Studienarbeit ist, werden hier nur die Reaktion auf zwei klassische Regelungsansätze dargestellt, die nicht weiter betrachtet werden können. Auf den folgenden Seiten wird die Simulation, die Vorstufe der Simulation, sowie ihre Ergebnisse vorgestellt.

5.1. Simulation

Zwischen der Bewegungsplanung und der Regelung des Fahrzeugs besteht eine große Abhängigkeit. Einzelne Ansätze der Regelung sind beispielsweise nur umsetzbar, wenn die Trajektorie fahrzeugunabhängig geplant wird. Modellbasierte Bewegungsplanung und Regelung haben dabei viele Gemeinsamkeiten, ebenso Abhängigkeiten. Für das Gesamtsystem des Rennwagens CM-22x ist daher eine gemeinsame Simulation der Submodule Planning und Control sehr sinnvoll. Die Simulation, die als Grundlage der Bewertung gilt, ist daher nicht allein eine Simulation um die Bewegungsplanung zu validieren, sondern auch um den Gesamtverbund von Bewegungsplanung und Regelung des Wagens zu visualisieren und das Verhalten zu analysieren. Ergebnis der Simulation ist demnach die Reaktion des Rennwagens auf einen bestimmten Sollwert. Der Sollwert stellt dabei den Pfadoptimierungsalgorithmus dar und die Reaktion das Regelverhalten der entworfenen, zum Zeitpunkt dieser Studienarbeit noch klassischen Regler.

Grundlage für die gemeinsame Simulation ist ein erstelltes Fahrzeugmodell. Dieses Fahrzeugmodell versucht möglichst realitätsnah das Fahrzeugverhalten vorzugeben. Für die Modellierung wurden verschiedene Kraftwirkungen errechnet und in Plots dargestellt. Das Fahrzeugmodell ist eine Zusammensetzung verschiedener Modelle. Grundlage für die Dynamik, bzw. die zweidimensionale Bewegung, ist dabei das Einspurmodell nach Riekert und Schunk [24]. Das Einspurmodell besitzt dabei verschiedene Eingangsgrößen. Eine dieser Eingangsgrößen sind die errechneten Reifenkräfte. Die Reifenkräfte werden nach dem Reifenmodell von Pacejka berechnet [25]. Neben den Reib- und Luftwiderständen werden ebenfalls der Antriebsstrang (Powertrain), die Bremse und die Lenkung modelliert. Eine vereinfachte Darstellung des Gesamtmodells wurde aus [26] abgeleitet. Die der Simulation zu Grunde liegenden Modelle und ihre Abhängigkeiten wurden in Abbildung 24 dargestellt. Die dort dargestellten Abhängigkeiten sind dabei in der Regel zu verrechnende Kräfte. Die aerodynamische Lift Force, die Kraft die das Auto nach oben drückt, verrechnet mit der Drag Force, die Kraft die das Auto auf den Boden zieht ist dabei mit der Achslast verrechnet. Durch stärkeres auf den Boden drücken durch Drag Force oder aber stärkeres Anheben des Fahrzeugs durch Lift Force entsteht eine Mehr- oder Wenigerbelastung der Achsen. Auch die anderen für das Fahrzeug

modellierten Teilbereiche verechnen sich und dienen schlussendlich als Grundlage für das Einspurmodell. Anhand dieser möglichst realistischen Fahrzeugmodellierung soll das Fahrverhalten innerhalb der Simulation nachgebildet werden. Die Grundlegenden Daten, beispielsweise der Reifen, liegen dabei bereits aus der letzten Saison vor. Andere Daten, wie beispielsweise des Antriebsstranges, konnten in der Vorbereitung auf die Saison durch andere Subteams bereits bestimmt werden. Die Simulation wurde in der Programmiersprache Python umgesetzt. Dies hat den Hintergrund, dass das autonome Gesamtsystem ebenfalls in Python umgesetzt wird und eine Migration und Einarbeitung folglich wenig Mehraufwand liefert. Die Simulation nutzt die Vorteile der Objektorientierung, demnach sind die einzelnen Modelle innerhalb von Klassen umgesetzt. In einer zentralen Datei, kann dann ein Fahrzeug bestehend aus den verschiedenen Modellen zusammengesetzt werden. Die Berechnung der verschiedenen Modellewerte erfolgen dabei immer zeitabhängig. Als Eingang der Simulation dienen Ausgangswerte des Track-filtering (die Mittellinie des Tracks und die zugehörigen Trackbegrenzungen). Anhand dieser Eingangsgrößen wird eine Trajektorie mit Geschwindigkeitsprofil erstellt. Diese Zwischenberechnung dient dann wiederum als Eingangsgröße für das Fahrzeugmodell und den Regelungscode.

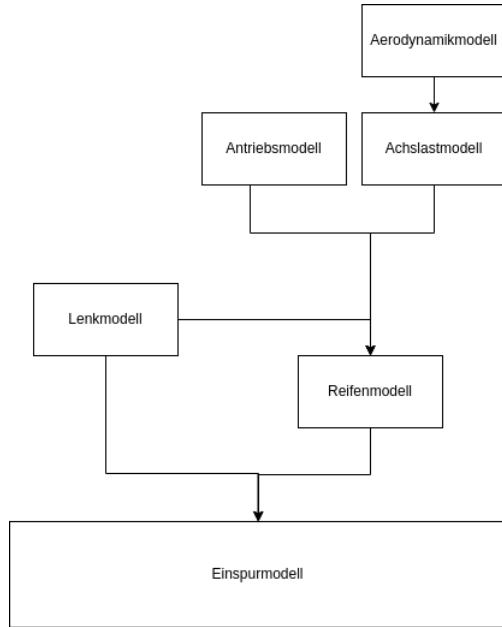


Figure 24: Darstellung der Simulation zu Grunde liegenden Modelle und ihre Abhängigkeiten

5.2. Simulationsdurchläufe

Die Simulationsergebnisse lassen sich in zwei Teile unterteilen. Der erste Teil der Simulation ist dabei eine visuelle Überprüfung der Trajektorie. Der zweite Teil die ausgiebig thematisierte Simulation mit Control zusammen. In dieser wird auf die Reaktion des

Fahrzeuges auf die gegebenen Input Daten geschlossen. Für den ersten Teil, die visuelle Überprüfung wurden Plots für verschiedene Rennstrecken angefertigt. Visualisiert werden neben der Strecke ebenfalls die Ideallinie und das Geschwindigkeitsprofil. Die Haupterkenntnisse der visuellen Simulation sind, dass die Trajektorie zusätzlich geglättet werden muss und die Parameter der Kostenfunktion variiert werden müssen. Auf das weitere Vorgehen wird im Folgenden Kapitel 6 eingegangen. In den untenstehenden Teilabschnitten werden die einzelnen Testläufe beginnend mit dem Hockenheimring vorgestellt und die eben genannten Probleme visualisiert.

5.2.1. Hockenheimring

Eingangsgröße dieses Testdurchlaufs sind die karthesischen Koordinaten der Mittellinie des Hockenheimrings. In den Folgenden Abbildungen ist zuerst die Ideallinie und dann das Geschwindigkeitsprofil gemeinsam mit der Ideallinie visualisiert:

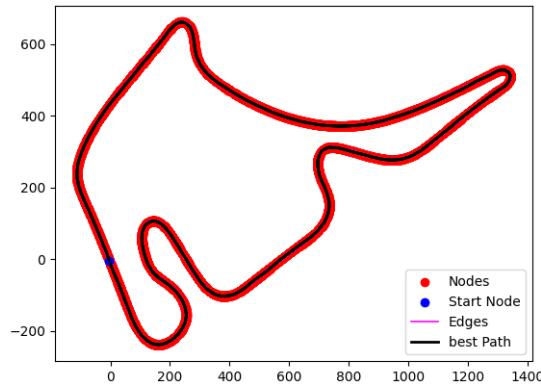


Figure 25: Darstellung eines Plots für die durch den in dieser Arbeit entwickelten Algorithmus zur Berechnung der Ideallinie anhand des Beispiels des Hockenheimrings

Die Abbildungen 25 & 26 stellen dabei vielversprechende Ergebnisse vor. Bei genauerer Betrachtung des idealen Pfades stellt sich heraus, dass dieser noch einmal geglättet werden muss (siehe Abbildung 27). Ohne die Glättung bleiben größere Geschwindigkeitssänderungen im Bereich der Kurven enthalten. Durch die sich dort schnell ändernde Krümmung, wird die Kurve ungeglättet nicht zeitoptimal gefahren. Außerdem sollte eine Anpassung der Kostenparameter vorgenommen werden, um unter anderem das angedeutete Kurvenschneiden (siehe Abbildung 28) zu verbessern. Die Abbildungen 27 & 28 befinden sich nocheinmal in größerer Ausführung im Anhang.

Im Folgenden wird auf die Reaktion des Fahrzeugs auf die vorgegebene Trajektorie eingegangen. Die für diese Simulation verwendeten Ansätze wurden ebenfalls als Teil einer Studienarbeit entwickelt und sollen als Rückfallebene für die Modellprädiktive Regelung genutzt werden. Aus den verschiedenen Ansätzen werden hier nur zwei beispielhaft als Referenz aufgeführt um die Validität der Trajektorie im Zusammenhang

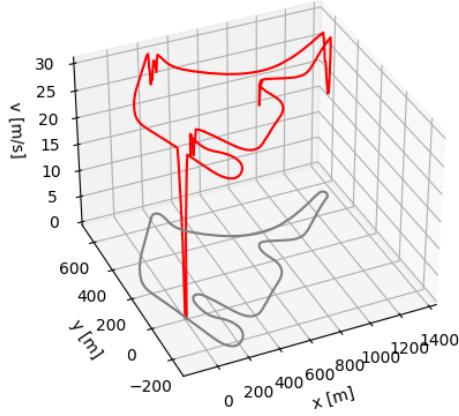


Figure 26: Visualisierung des Geschwindigkeitsprofils in Abhängigkeit der errechneten Ideallinie am Beispiel des Hockenheimrings mit Initial- und Endgeschwindigkeit von 0 Metern pro Sekunde

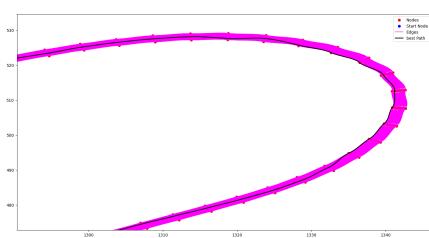


Figure 27: Visualisierung des ungeglätteten Pfades, der besonders in Kurven zu Welligkeit neigt

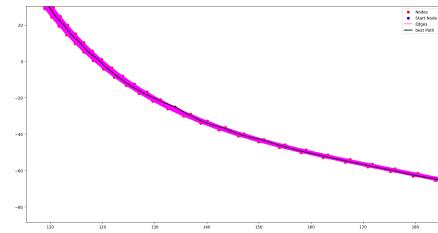


Figure 28: Darstellung des angedeuteten Kurvenschneidens durch den berechneten idealen Pfad für den Hockenheimring

mit dem Fahrzeugmodell darzustellen. In Abbildung 29 ist dabei eine allgemeine Übersicht über die zwei möglichen Regler bzw. das Fahrzeugverhalten zu sehen. Neben dieser allgemeinen Übersicht wurde für diese Studienarbeit noch die laterale Abweichung bestimmt und dargestellt (Abbildung 30). Aus dieser Darstellung lässt sich der Rückschluss ziehen, dass der kombinierte Pure Pursuit Regler für die Beispieltrajektorie des Hockenheim Rings gut geeignet wäre. Bis auf die in Abbildung 31 dargestellte größere Abweichung, bei der das Fahrzeug vermutlich kurz vom eigentlichen Track abkommen würde, wird das Fahrzeug sehr gut auf die Trajektorie geregt und entspricht somit den durch den Usecase gegebenen Ansprüchen.

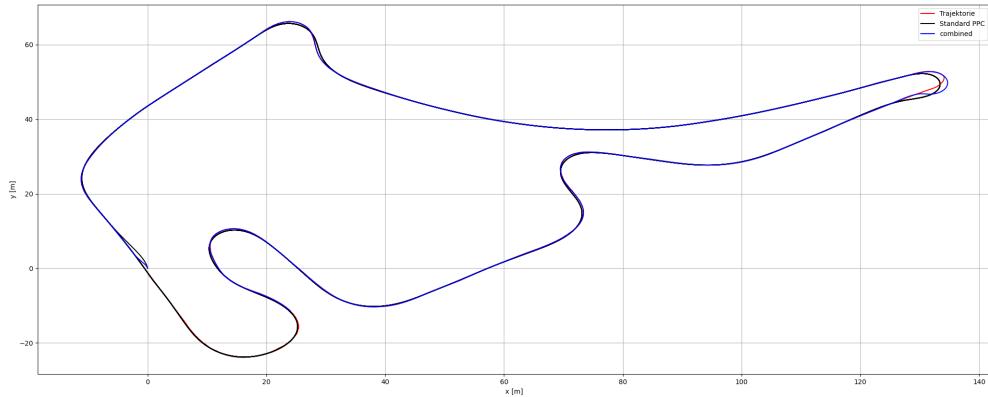


Figure 29: Darstellen des Fahrzeugverhalten durch Regelung auf die errechnete Ideallinie am Beispiel des Hockenheim Rings

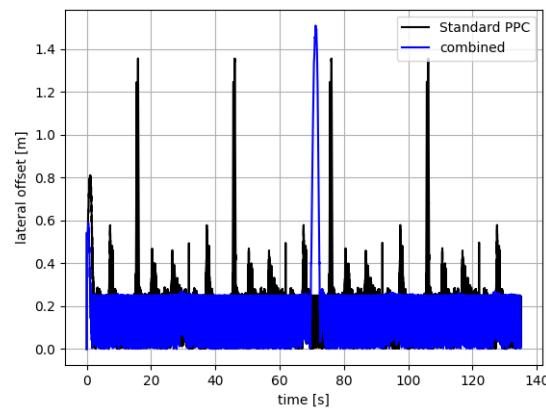


Figure 30: Darstellung des lateralen Offset für ausgewählte Regelansätze für die Validierung der errechneten Ideallinie des Hockenheim Rings. Bildquelle: Oliver Zbaranski

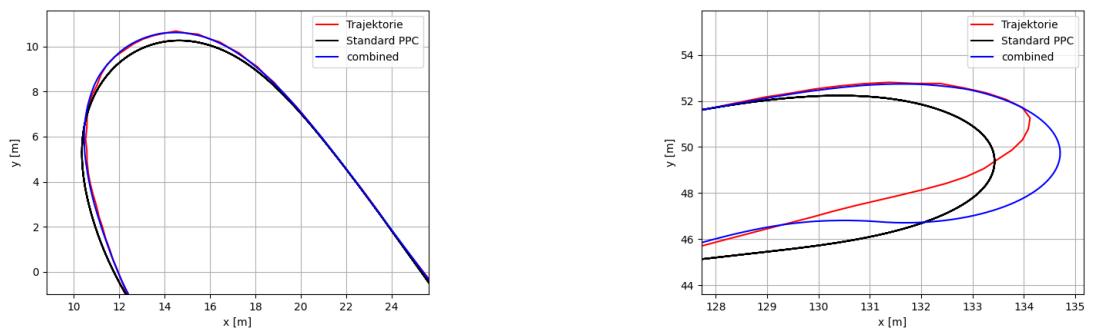


Figure 31: Gegenüberstellung einer geregelten Fahrzeugtrajektorie mit wenig lateraler Abweichung und einer Trajektorie mit großer lateraler Abweichung am Beispiel der für den Hockenheimring errechneten Ideallinien

5.2.2. Melbourne

Ein zweiter Testdurchlauf wurde für die Formel1 Strecke in Melbounre (vor ihrem Umbau zur Saison 2022) durchgeführt. Auch bei diesen Tests sind ähnliche Schwachpunkte des Algorithmus zu beobachten. Die errechnete Ideallinie stimmt dabei ebenfalls noch nicht mit der wirklichen Ideallinie überein, was an der Gewichtung der Kostenparameter liegt. In besonders kurvigen Streckenabschnitten neigt der Pfad dazu wellig zu werden und durch die dadurch entstehenden großen Krümmungsänderungen entsteht ein nicht optimaler Geschwindigkeitsverlauf.

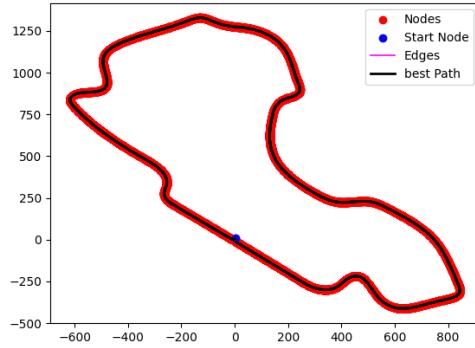


Figure 32: Darstellung der durch den Algorithmus berechneten idealen Trajektorie für die Formel1 Strecke von Melbourne, vor ihrem Umbau 2022

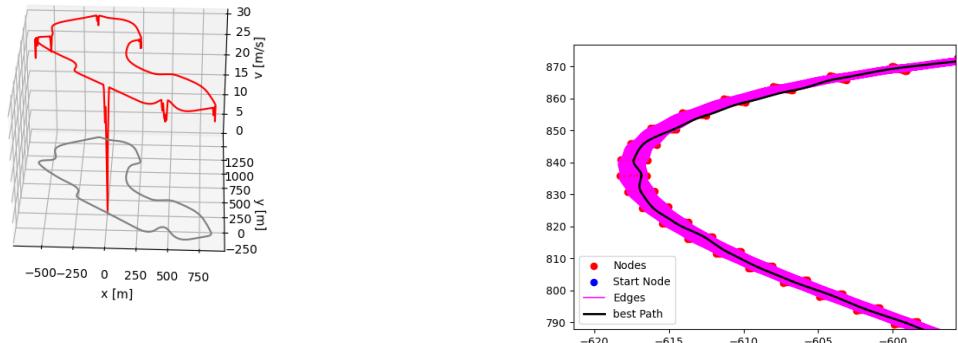


Figure 33: Darstellung des Geschwindigkeitsprofils für die errechnete Ideallinie der Formel1 Strecke Melbourne mit Initial- und Endgeschwindigkeit von 0 Metern pro Sekunde

Figure 34: Darstellung des zu glätten- den Pfades für die Kurve 1 der Formel 1 Strecke Mel- bourne

Auch für den Test in Melbourne bestätigen sich die Ergebnisse des Fahrzeugverhaltens mit Hinblick auf die bisherigen Erfahrungswerte. Für diesen Teil der Simulation sticht besonders die Regelung mittels des kombinierten Pure Pursuit Controllers hervor. Das Fahrzeug folgt nahezu ideal der gegebenen Trajektorie und kommt auch mit den gegebenen Welligkeiten des Pfades gut zurecht (zu sehen in Abbildungen 35 & 36). In der lateralalen Abweichung lässt sich bis auf eine anfängliche Fehlstellung, durch nicht identische Startpunkte keine größere Abweichung feststellen. Die Abweichung pendelt zwischen 20 cm und dem ideal vorgegebenen Wert (ebenfalls zu sehen in Abbildung 37).

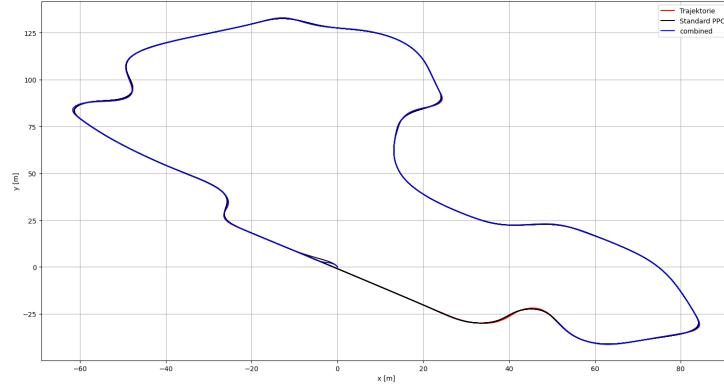


Figure 35: Darstellung des Fahrzeugverhaltens für zwei Regleransätze auf Basis der errechneten Trajektorie für die Formel1 Strecke Melbourne vor ihrem Umbau



Figure 36: Darstellung des Simulierten Fahrzeugverlaufs für zwei beispielhafte Kurven der Formel1 Strecke Melbourne

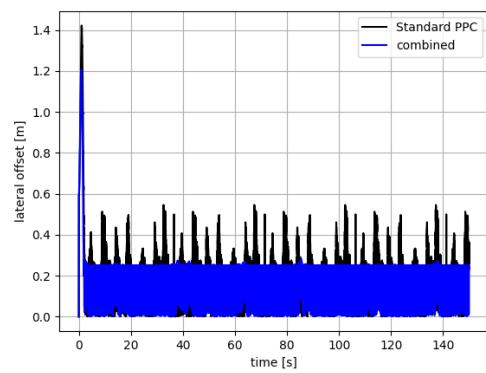


Figure 37: Darstellung der lateralen Abweichung im Vergleich von errechneter Trajektorie und geregeltem Fahrzeug für die Formel1 Strecke vor ihrem Umbau.
Bildquelle: Oliver Zbaranski

5.2.3. Berlin

In diesem Teil wird auf die Simulation einer Rennstrecke in Berlin eingegangen. Dieser Testlauf stellt den längsten dar. Bei der visuellen Überprüfung der Trajektorie fällt besonders noch das Problem der ungeglätteten Trajektorie auf. Durch die bereits in der Draufsicht in der Ideallinie zu erkennenden großen Krümmungsänderungen, kommt es zu vielen drastischen Geschwindigkeitsänderungen (siehe Abbildung 38). Die daraus entstehende Trajektorie stellt definitiv nicht den idealen Strecken- und Geschwindigkeitsverlauf dar. Die aus den drastischen Krümmungsänderungen folgenden Geschwindigkeitssänderungen sind in Abbildung 39 dargestellt. Im Laufe der Simulation müssen hierfür weiter die Kostenparameter verändert werden und die Trajektorie geglättet werden. Anders als bei den anderen Strecken tritt ein Pfad mit großer Krümmungsänderung bereits in nicht Kurven auf. Durch die Neubestimmung der Kostenparameter und das Glätten der Trajektorie soll dies jedoch komplett kompensiert werden.

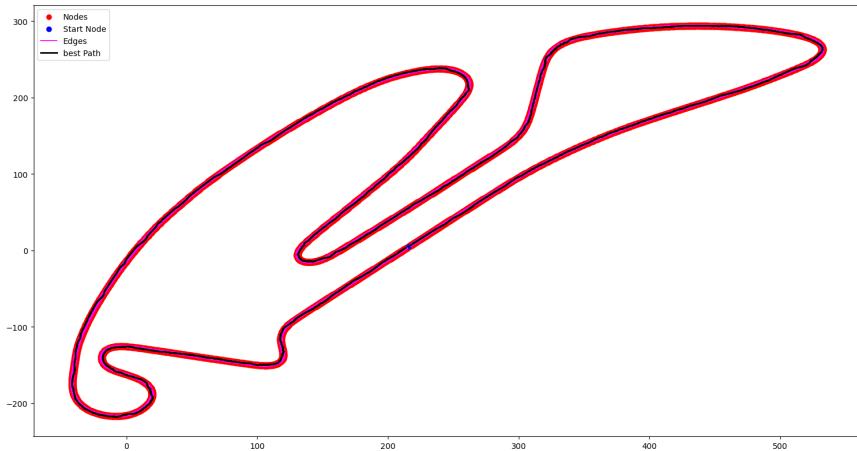


Figure 38: Darstellung der errechneten Ideallinie für die Rennstrecke von Berlin. Besonderes Augenmerk liegt auf der Welligkeit des Pfades

Trotz der extremen Welligkeit der gegebenen Referenztrajektorie und dem nahezu sprunghaften Verhalten des Geschwindigkeitsprofils, sieht das Simulationsergebnis für die Strecke in Berlin sehr gut aus (siehe Abbildung 40). Das Simulationsergebnis sieht dabei sogar besser aus als das des Hockenheimrings. Gerade der kombinierte Pure Persiute Controller regelt das Fahrzeug nahezu konstant auf die Referenztrajektorie (siehe Abbildung 41). Die laterale Abweichung bewegt sich dabei wieder im Bereich zwischen 0 und 20 cm (siehe Abbildung 42).

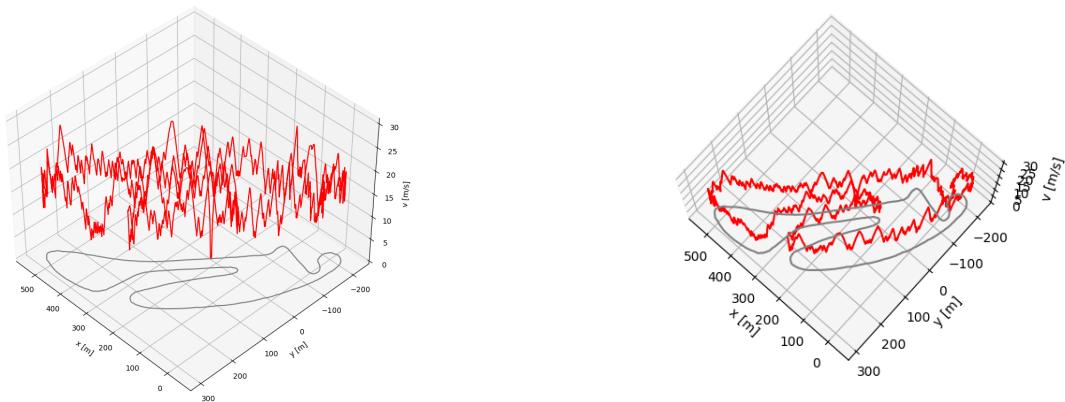


Figure 39: Darstellung des durch die Welligkeit des Pfades beeinflussten Geschwindigkeitsprofil mit nahezu sprunghaften Geschwindigkeitswerten für die Rennstrecke in Berlin

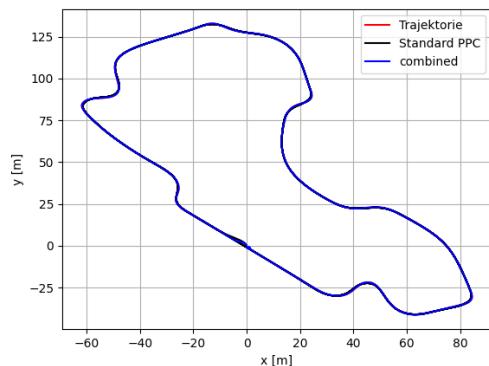


Figure 40: Visualisierung der simulierten Fahrdynamik für den Rennwagen CM-22x am Beispiel der errechneten Referenztrajektorie für die Rennstrecke in Berlin

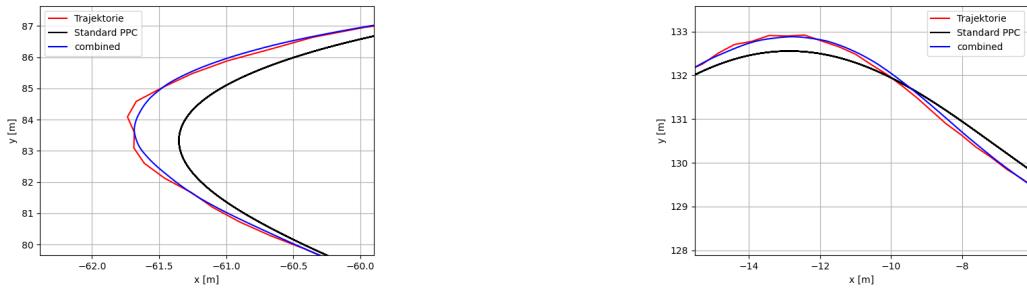


Figure 41: Darstellung des simulierten Kurvenverlaufs für den Rennwagen CM-22x für ausgewählte Streckenabschnitte auf der Rennstrecke Berlin. Zu sehen ist neben der Referenztrajektorie ebenfalls das Verhalten bei Regelung mit dem Standard Pure Pursuit Controller und einem von CURE weiterentwickelten Controller

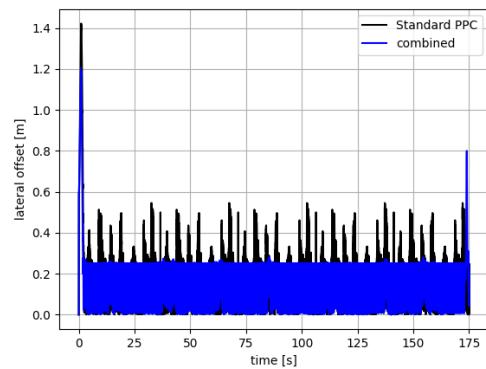


Figure 42: Darstellung der lateralen Regelabweichung bei errechneter Ideallinie für die Rennstrecke von Berlin. Bildquelle: Oliver Zbaranski

5.3. Simulationsergebnisse

Die Simulation zeigt, dass die simulierte Fahrdynamik gut mit den erwarteten Werten übereinstimmt. Gerade durch die Verbesserungen des Reglungsansatzes entsteht ein Fahrzeugverhalten, was fast identisch dem, der vorgegebenen Trajektorie entspricht. Wie im vorhinein bereits beschrieben, sollte die Simulation dazu genutzt werden, die Kostenparameter zu validieren. Anhand der visuellen Simulationsergebnissen lässt sich ableiten, dass diese noch nicht richtig gewählt werden. Es wird zwar eine Trajektorie ausgegeben, welche in Ansätzen die wichtigen Manöver zum Beispiel des Kurvenschneidens durchführt, der Pfad stellt dabei jedoch noch nicht die Ideallinie dar. Durch die Welligkeit des Pfades und die daraus entstehenden Krümmungsänderungen entsteht außerdem ein im Vergleich sprunghaftes Geschwindigkeitsprofil. Um die Rundenzeiten weiter verbessern zu können, muss der Pfad als Eingangsgröße des Geschwindigkeitsprofils noch einmal geglättet werden. Die Ergebnisse der Simulation zusammengefasst:

- funktionsfähiger Algorithmus
- Pfad stellt noch nicht den idealen Pfad dar
- Anpassung der Kostenparameter nötig
- Sprunghaftes Geschwindigkeitsprofil durch sprunghafte Krümmungsänderungen
- Weiteres Glätten des Pfades nötig
- Simulierte Fahrdynamik deckt sich gut mit vorgegebener Dynamik

6. Zusammenfassung und Ausblick

Der in dieser Arbeit entwickelte Algorithmus ist in der Lage verschiedene Pfade mit zugehörigem Geschwindigkeitsprofil zu erstellen und stellt bereits jetzt einen großen Fortschritt zum letztjährigen Ansatz dar. Dennoch gibt es noch Verbesserungsmöglichkeiten und Probleme, derer sich bis zum Start der Events angenommen werden soll. Auch wenn gerade im Bereich des Geschwindigkeitsprofils durch die Welligkeit der Trajektorie noch Verbesserungspotential besteht, ist hierfür bereits eine Lösung, die bis zu den Events implementiert werden kann, in Sichtweite. Trotz der sehr variablen Geschwindigkeiten stellte sich die Reaktion des Fahrzeug als relativ störsicher dar. Im Nachgang an die Simulation wurde beschlossen, den Abstand der Planning Normalen zu verringern, da innerhalb des AS-Teams davon ausgegangen wird, dass sich mit geringerem Abstand der Planning Normalen sogar die konstante Abweichung von 20 cm kompensieren lässt. Gerade die gemeinsame Simulation von Planning & Control stellte sich als wichtiger und wegweisender Meilenstein dar. Welche Änderungen in Zukunft, vor allem aber noch für die Events dieser Saison geplant sind, wird im Folgenden betrachtet.

Bis zum Start der Events muss vor allem die Kostenfunktion überarbeitet werden. Für die Bestimmung der Kostenfunktionsparameter ist ein Machine Learning Modell geplant. Um Algorithmik des Machine Learnings anwendbar zu machen, bedarf es weitaus mehr, vor allem aber realistischere Trackdaten. Dies soll mit einem von Cure entwickelten Trackgenerator möglich gemacht werden. Durch das Generieren verschiedener Tracks und dem Bestimmen der idealen Trajektorie soll auf die möglichst besten Parameter geschlossen werden. Neben dem Bestimmen der neuen Kostenfunktionsparameter muss die Trajektorie ebenfalls nochmal geglättet werden. Hierfür soll im Nachgang der Studienarbeit ein passendes Verfahren ausgewählt und angewendet werden. Es soll besonders darauf geachtet werden, dass die ausgewählte Methode sehr zeiteffizient ist. Neben den durch die Simulation aufgefallenen Verbesserungsmöglichkeiten soll außerdem noch die Möglichkeit der globalen Pfadoptimierung implementiert werden. Auch wenn es auf den Plots der Simulation bereits so aussah, ist die Trajektorie bisher noch nicht in sich geschlossen (was beim diesjährigen Gesamtsystem erst beim Vorliegen einer globalen Karte der Fall sein wird). Für diese Erweiterung wurden im Code bereits einige Extras implementiert, die bisher aber noch nicht funktionsfähig sind und durch die Priorität der lokalen Planung erst einmal zurückgestellt wurden. Bis zu den Events sollen demnach noch Folgende Punkte implementiert werden:

- Neubestimmung der Kostenparameter
- Glätten der Trajektorie
- Implementierung einer in sich geschlossenen Trajektorie

Durch die neu entwickelte Algorithmik ist zu jedem Zeitpunkt eine Verbesserung zur Vorsaison gegeben. Besonders auf geraden Strecken, wie zum Beispiel der Acceleration ist es sehr wahrscheinlich, dass der Rennwagen an der Belastungsgrenze gefahren werden kann. Mit den geplanten zusätzlichen Implementierungen im Bereich des Planning und der zusätzlichen Implementierung einer modellprädiktiven Regelung ist es sehr

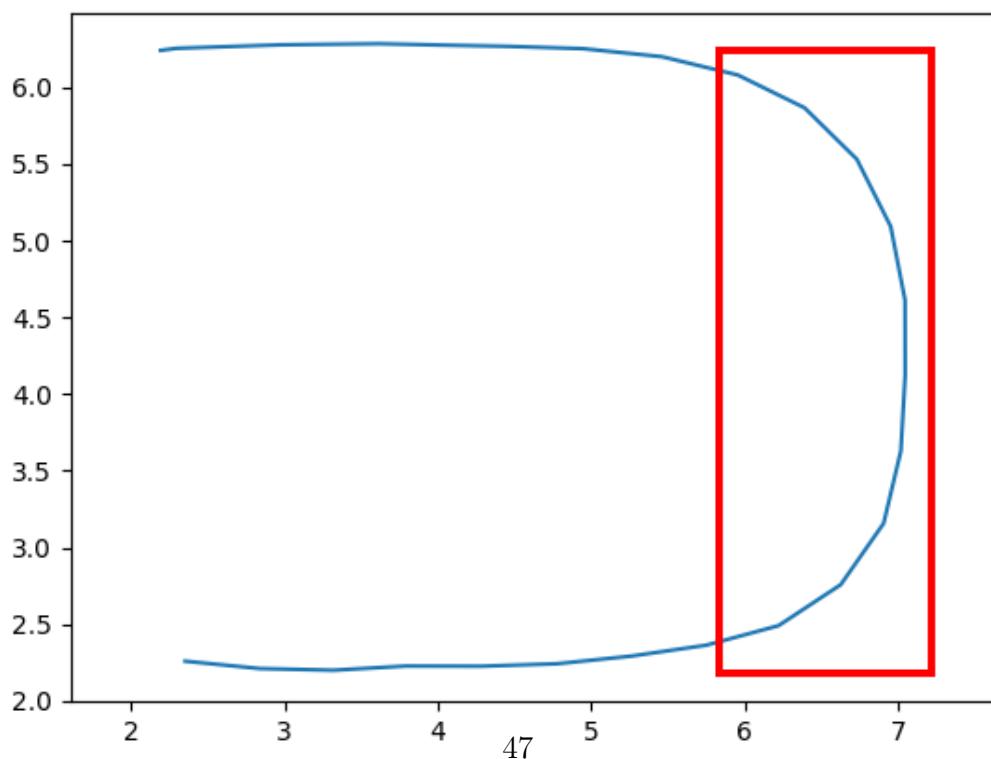
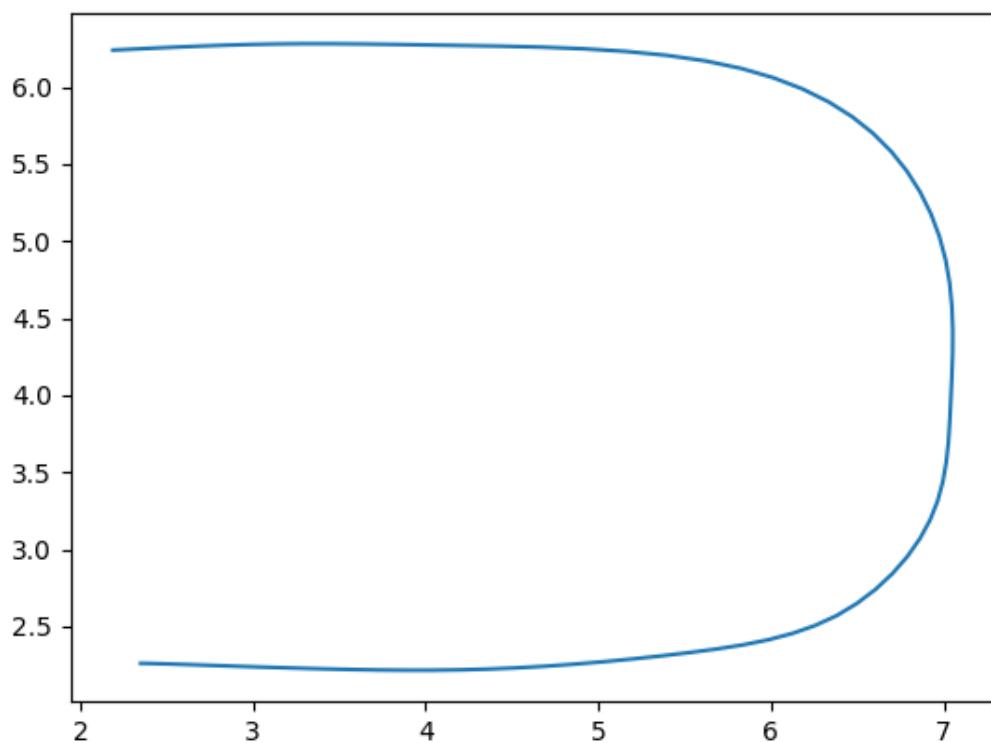
wahrscheinlich, dass das Auto auf dem ganzen Track die bestmögliche Performance abliefern kann. Es könnte lediglich zur Leistungslimitierung durch fehlende Computing Power kommen. Gerade der Umstieg von einer auf drei Kameras und gegebenenfalls die modellprädiktive Regelung könnten viel Rechenkapazität in Anspruch nehmen. Leider ist zum jetzigen Zeitpunkt keine Prognose für diese Thematik abzugeben, die Ergebnisse werden sich erst nachdem Fertigstellen aller Algorithmen durch Tests auf dem Auto Anfang Juni 2022 validieren lassen. Für die Zukunft sollte sich überlegt werden, den Code in *C₊₊* und der neuen ROS Version ROS2 umzusetzen. Diese Umsetzung könnte zu einer noch effizienteren Codenutzung führen, ist aber frühstens zur nächsten Saison geplant. Alles in allem wurden die geplanten Ziele der Studienarbeit erreicht und neue Verbesserungsmöglichkeiten erschlossen, auf dieser Grundlage sollte es dem Verein CURE Mannheim möglich sein, eine gute Driverless Rennsaison zu fahren.

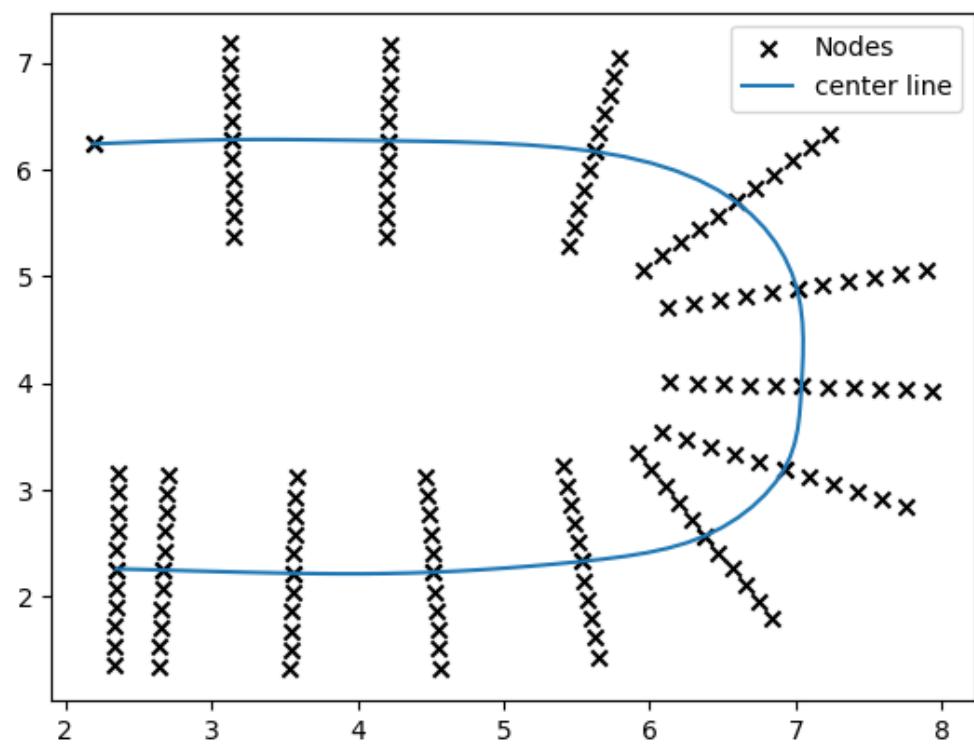
Literaturverzeichnis

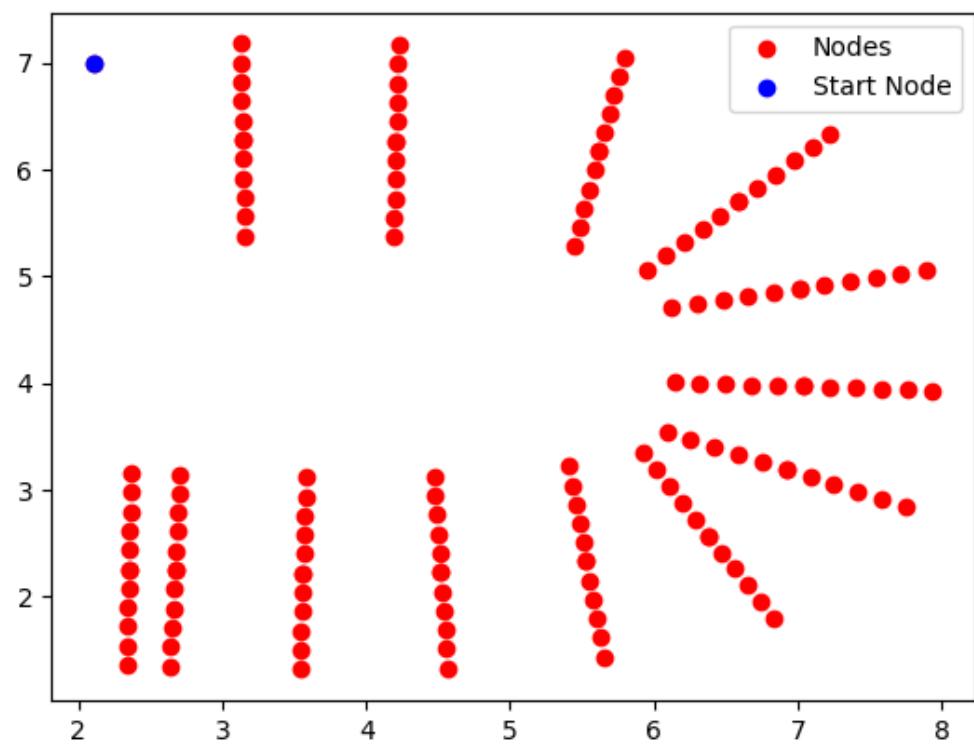
- [1] Formula Student Germany GmbH. „Formula student germany - competitions.“ (2022), [Online]. Available: <https://www.formulastudent.de/world/competitions/> (visited on 01/12/2022).
- [2] Formula Student Germany GmbH. „Formula student rules 2022, D5 acceleration event.“ version 1.0. (2021), [Online]. Available: https://www.formulastudent.de/fileadmin/user_upload/all/2022/rules/FS-Rules_2022_v1.0.pdf#section.7.5 (visited on 01/13/2022). Published 11/21/2021.
- [3] Formula Student Germany GmbH. „Formula student rules 2022, D4 skid pad event.“ version 1.0. (2021), [Online]. Available: https://www.formulastudent.de/fileadmin/user_upload/all/2022/rules/FS-Rules_2022_v1.0.pdf#section.7.4 (visited on 01/13/2022). Published 11/21/2021.
- [4] Association of Automotive Engineers. „Fs east, Final result.“ (2021), [Online]. Available: https://fseast.eu/wp-content/uploads/2021/08/FS-East-2021_DV_Final_v2.pdf (visited on 01/14/2022).
- [5] J. Ziegler, „Optimale Bahn- und Trajektorienplanung für Automobile,“ Dissertation, Karlsruher Institut für Technologie, 2015, pp. 31–33. DOI: <https://doi.org/10.5445/IR/1000057846>.
- [6] LaValle, Steven M., in *Planning Algorithms*. 2006, pp. 855–858. [Online]. Available: <http://planning.cs.uiuc.edu/>.
- [7] Universität Leipzig - Abteilung für Bild- und Signalverarbeitung, „§4 voronoi und delaunay,“ o.D.
- [8] Blumhoff, Birk, „Modelgestützte Pfadplanung eines autonomen Rennsport-Prototyps,“ Duale Hochschule Baden Württemberg, 2021.
- [9] B. Sahre, „Untersuchung der Effizienz von RRT* bei autonomen Autos,“ Freie Universität Berlin, 2018, pp. 5–9. [Online]. Available: <https://www.mi.fu-berlin.de/inf/groups/ag-ki/Theeses/Completed-theses/Bachelor-theses/2018/Sahre/index.html>.
- [10] Ranganathan, Ananth and Koenig, Sven, „Pdrrts: Integrating graph-based and cell-based planning,“ 2004. DOI: <https://doi.org/10.1109/IROS.2004.1389833>.
- [11] M. Yastremsky, „Rrt-based path planning and model predictive control for an autonomous race car,“ Technische Universität Hamburg, Master Thesis, 2019.
- [12] M. Yastremsky , „Rrt with multiple remote goals,“ Repository, 2020. [Online]. Available: https://github.com/MaxMagazin/ma_rrt_path_plan.
- [13] A. Raving, „Formula student driverless resources,“ 2020. [Online]. Available: <https://github.com/AMZ-Driverless/fsd-resources>.
- [14] Benz, Lukas, „Entwicklung eines Algorithmus zur Zeitoptimierung der Trajektorie bei Rundkursen,“ Duale Hochschule Baden Württemberg, 2020.

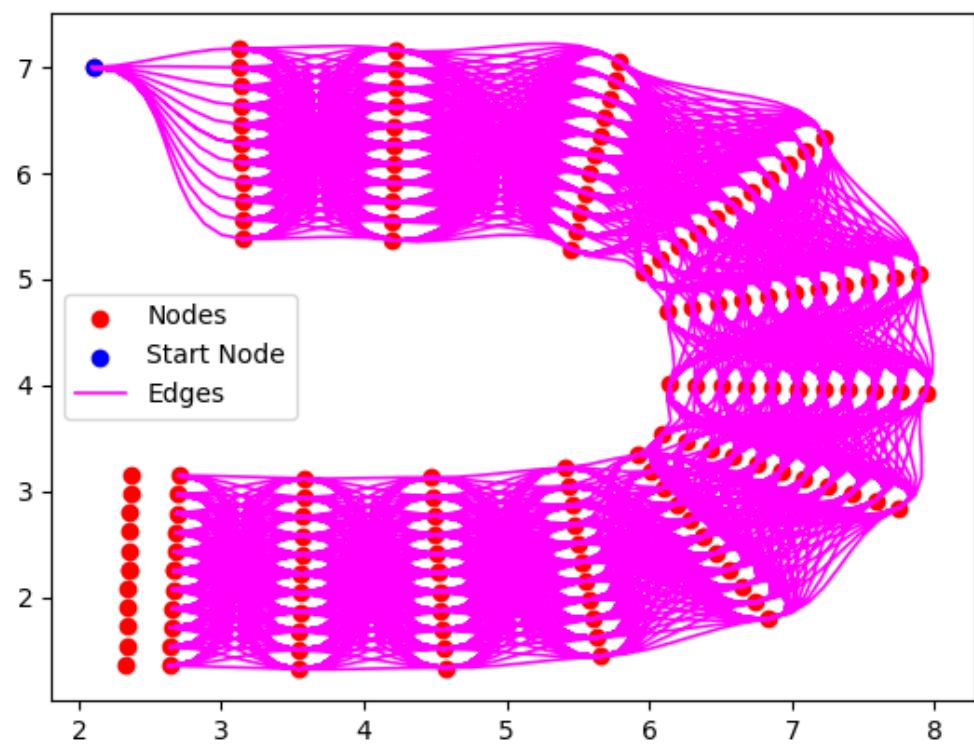
- [15] A. Ng, „Linear regression with one variable, Cost function,“ o.D.
- [16] A. Heilmeier, M. Geisslinger, and J. Betz, „A quasi-steady-state lap time simulation for electrified race cars,“ in *2019 Fourteenth International Conference on Ecological Vehicles and Renewable Energies (EVER)*, 2019, pp. 1–10. DOI: 10.1109/EVER.2019.8813646.
- [17] T. Novotny, „Lap time simulation (výpočet teoretického času vozidla na závodním okruhu),“ 2017. DOI: <https://doi.org/10.13140/RG.2.2.34448.46085>.
- [18] T. Stahl, A. Wischnewski, J. Betz, and M. Lienkamp, „Multilayer graph-based trajectory planning for race vehicles in dynamic scenarios,“ in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019, pp. 3149–3154.
- [19] Heilmeier, Wischnewski, Hermansdorfer, Betz, Lienkamp, Lohmann, „Minimum curvature trajectory planning and control for an autonomous racecar,“ 2019. DOI: <https://doi.org/10.1080/00423114.2019.1631455>.
- [20] Christ, Wischnewski, Heilmeier, Lohmann, „Time-optimal trajectory planning for a race car considering variable tire-road friction coefficients,“ 2019. DOI: <https://doi.org/10.1080/00423114.2019.1704804>.
- [21] Hermansdorfer, Betz, Lienkamp, „A concept for estimation and prediction of the tire-road friction potential for an autonomous racecar,“ 2019. DOI: <https://doi.org/10.1109/ITSC.2019.8917024>.
- [22] Herrmann, Passigato, Betz, Lienkamp, „Minimum race-time planning-strategy for an autonomous electric racecar,“ 2020. DOI: <https://doi.org/10.1109/ITSC45102.2020.9294681>.
- [23] L. Papula, „Mathematische Formelsammlung, Für Ingenieure und Naturwissenschaftler,“ in 12th ed. 2017, pp. 372–373. DOI: <https://doi.org/10.1007/978-3-658-16195-8>.
- [24] Riekert, P., Schunck, T.E., in *Fahrmechanik des gummibereiften Kraftfahreugs*. Ing. Arch, 1940. DOI: <https://doi.org/10.1007/BF02086921>.
- [25] Racejka, Hans B., in *Tire and Vehicle Dynamics*. Butterworth-Heinemann, 2012, ISBN: 978-0-08-097016-5. DOI: <https://doi.org/10.1016/C2010-0-68548-8>.
- [26] Halfmann, C., Holzmann, H., in *Adaptive Modelle für die Kraftfahrzeugsdynamik*, VDI-Buch, Ed., 1st ed., Berlin: Springer Verlag, 2003, ISBN: 978-3-642-18215-0. DOI: <https://doi.org/10.1007/978-3-642-18215-0>.

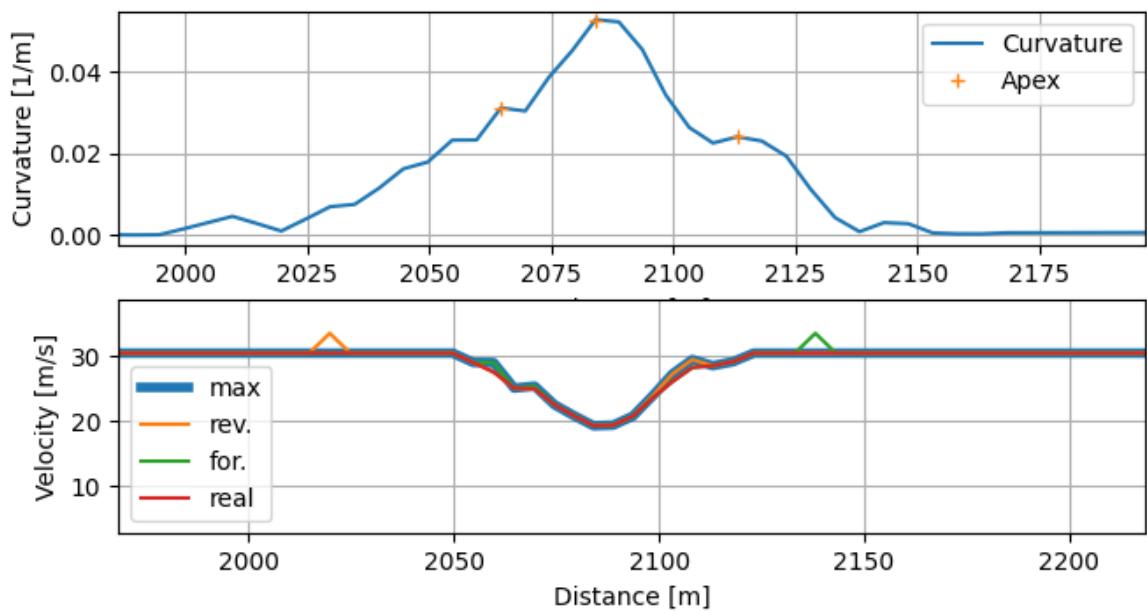
A. Bildanhang Kapitel 4

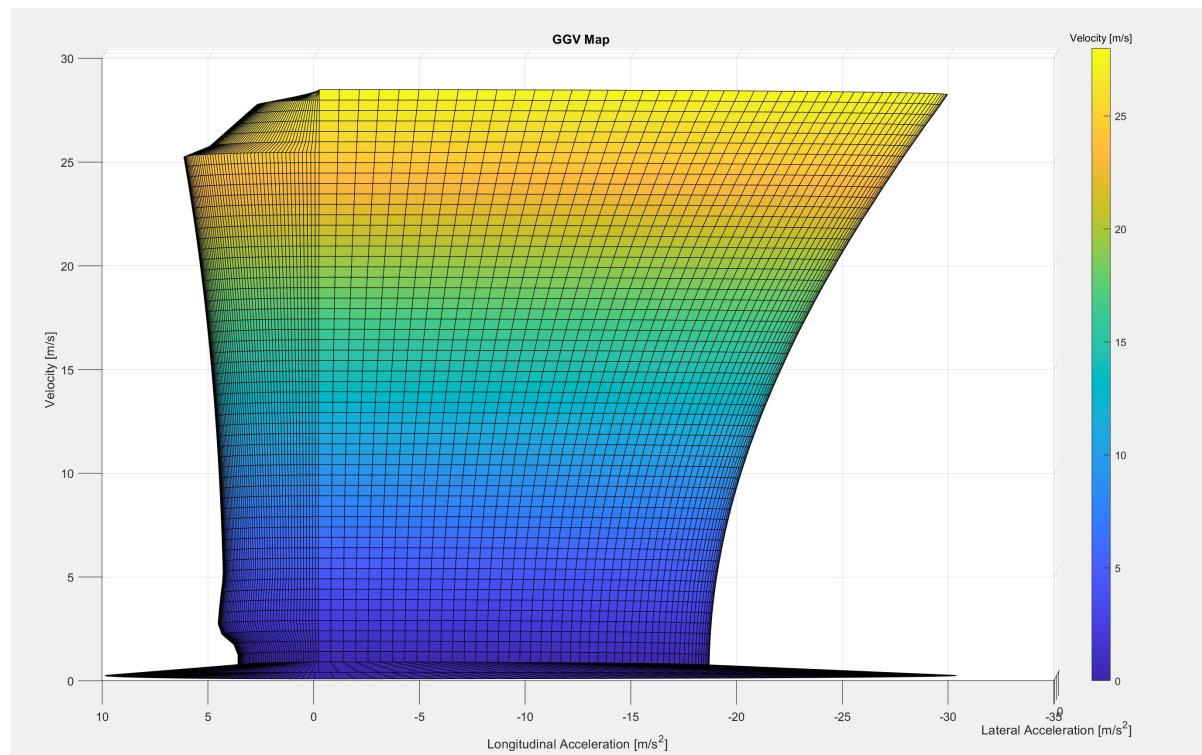




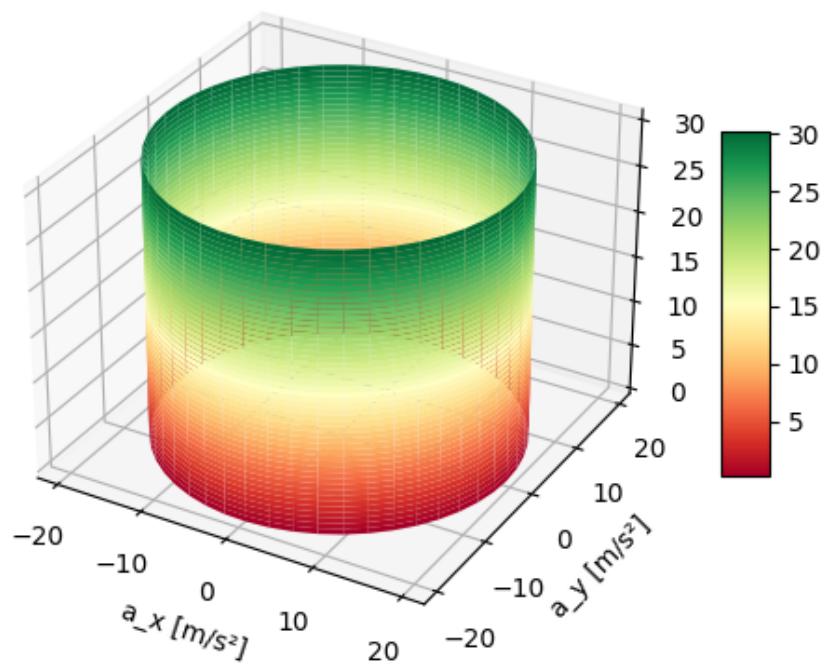


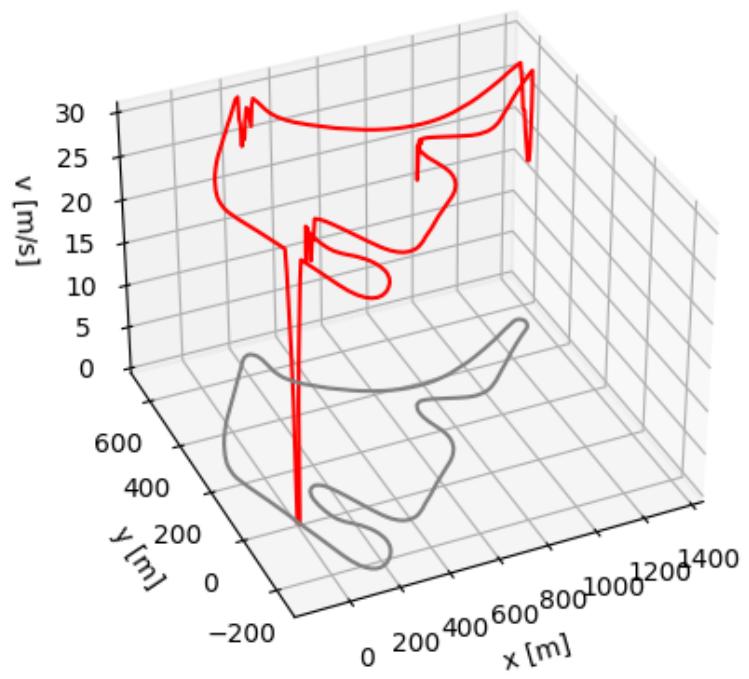






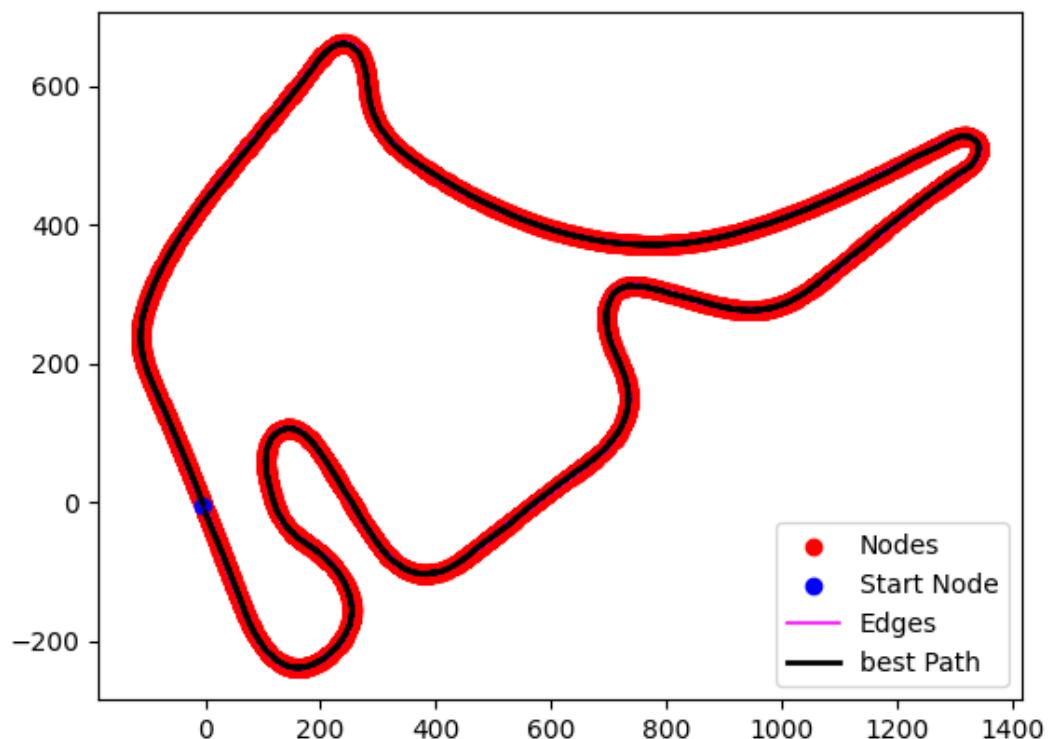
ggv-Map

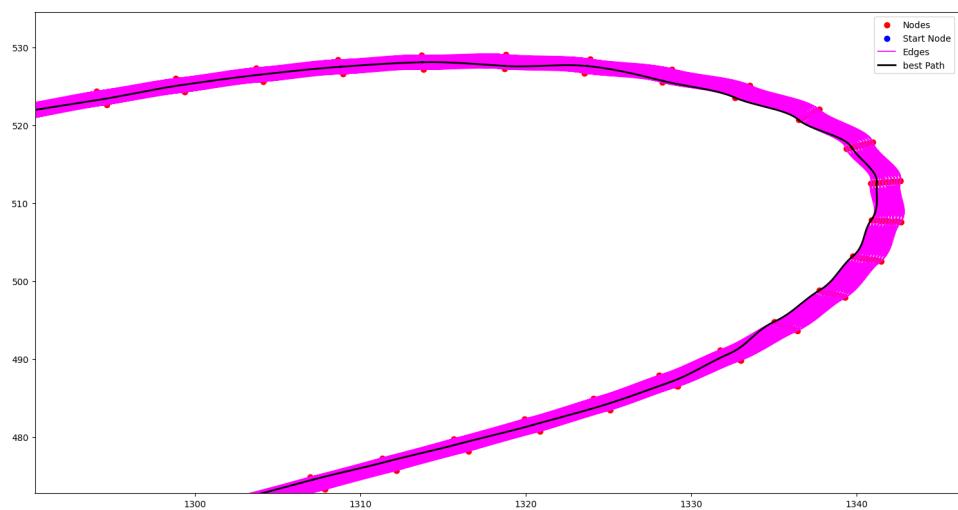
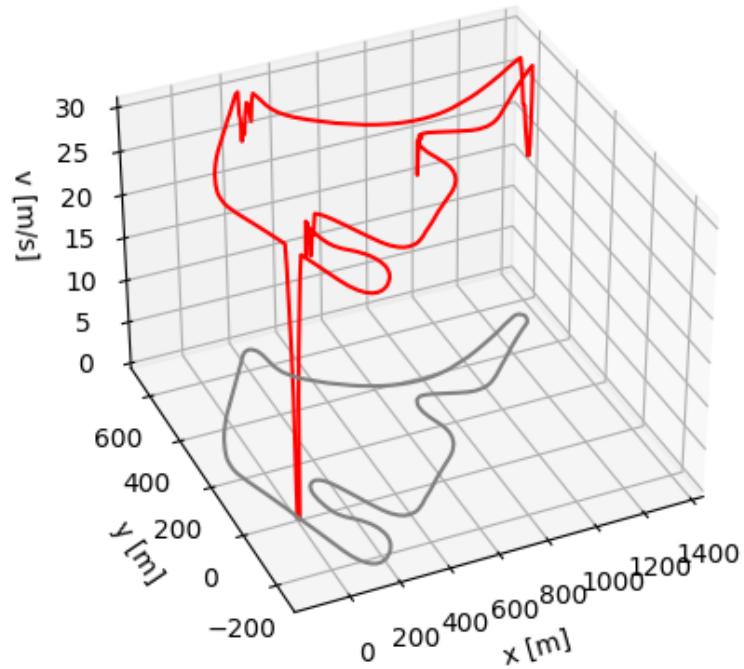


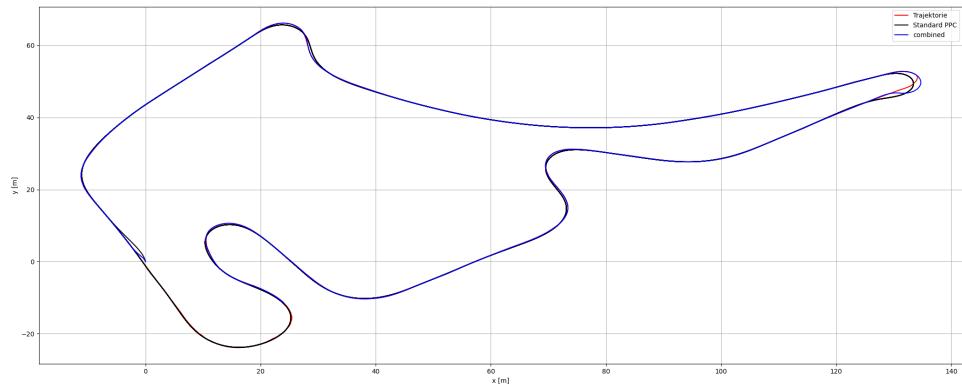
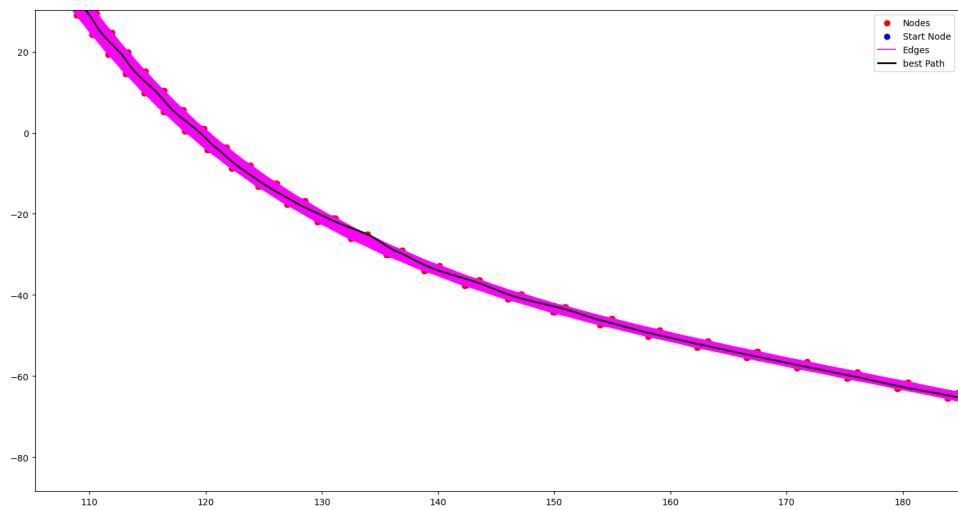


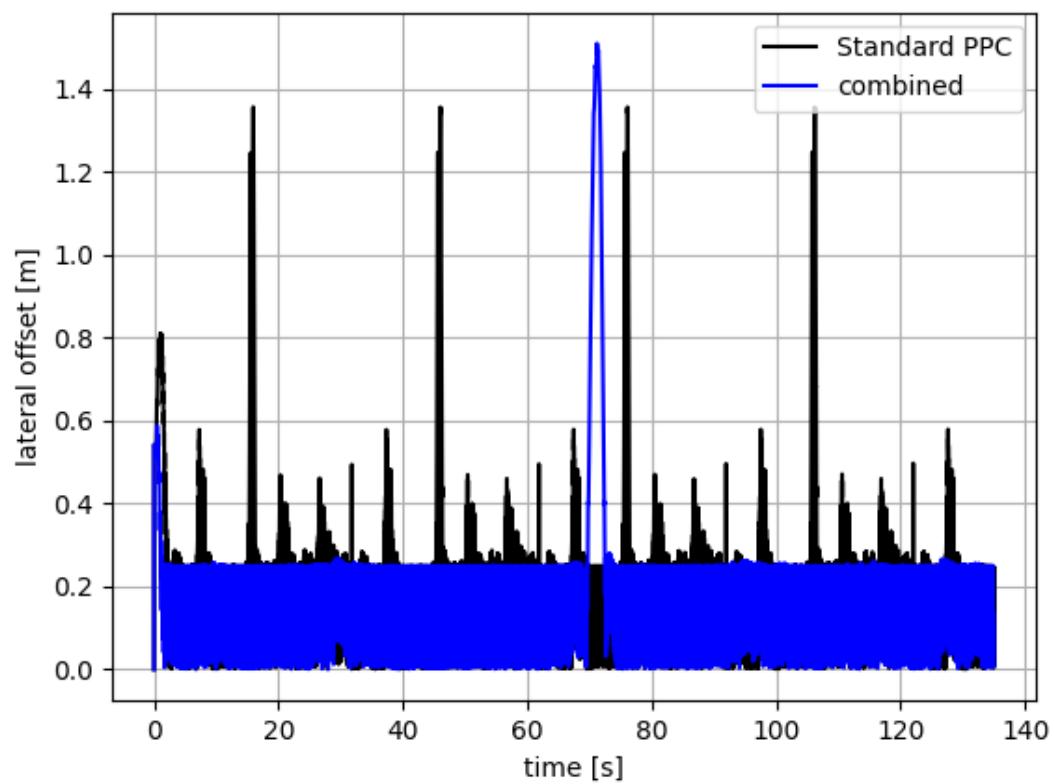
B. Bildanhang Kapitel 5

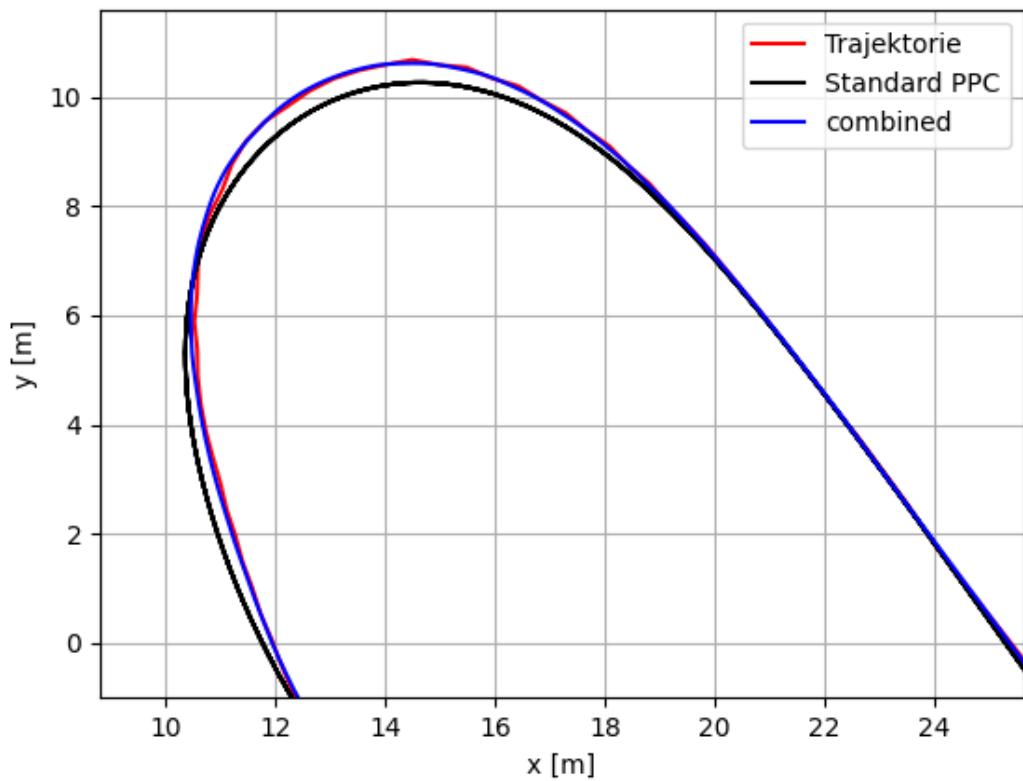
Hockenheim:

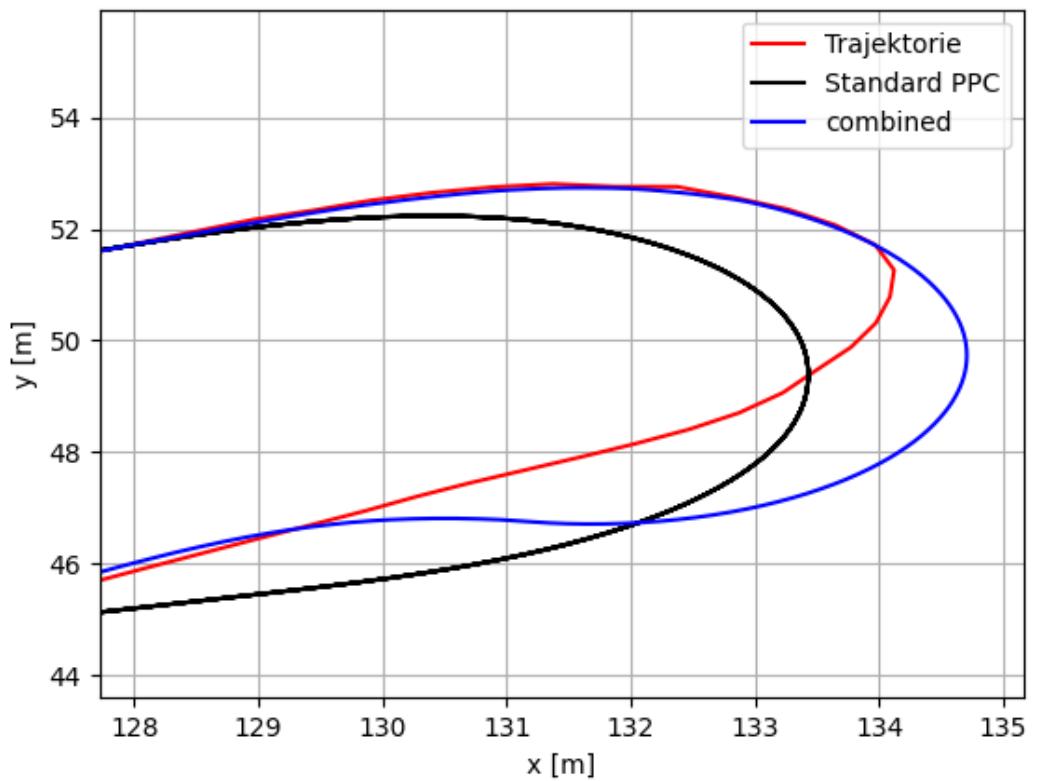




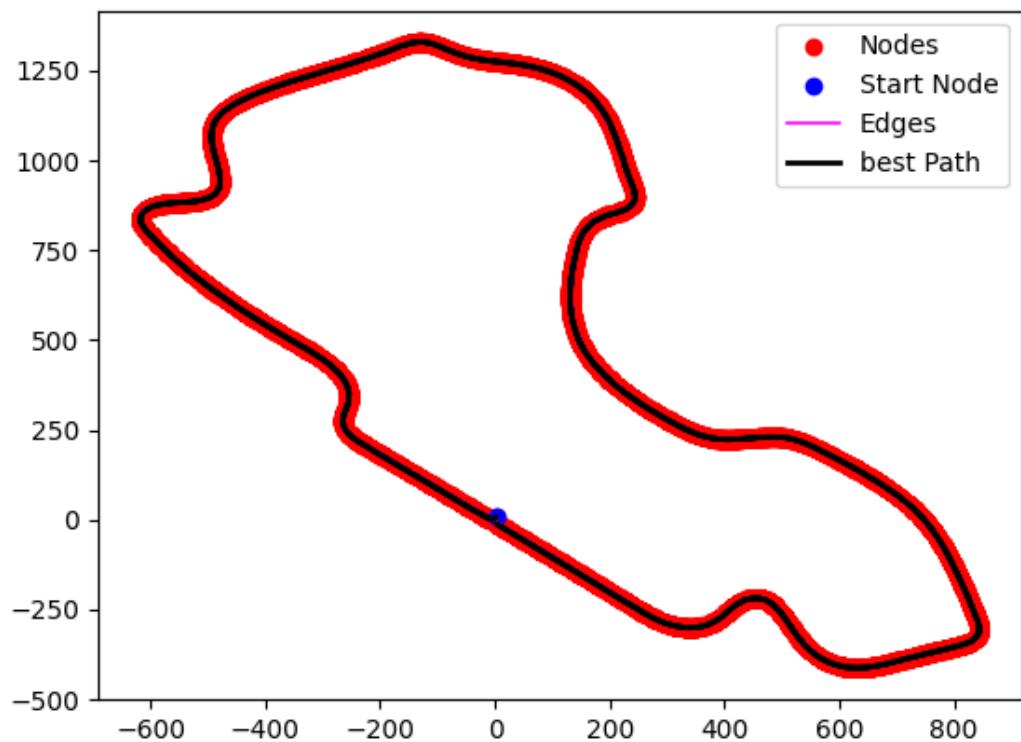


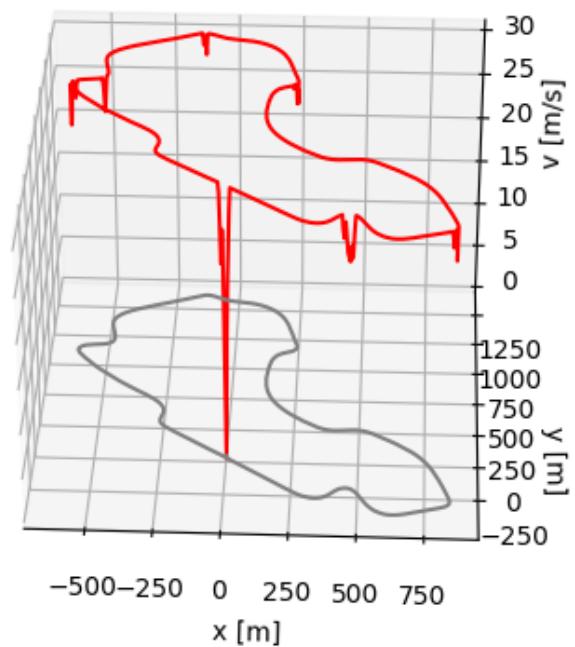


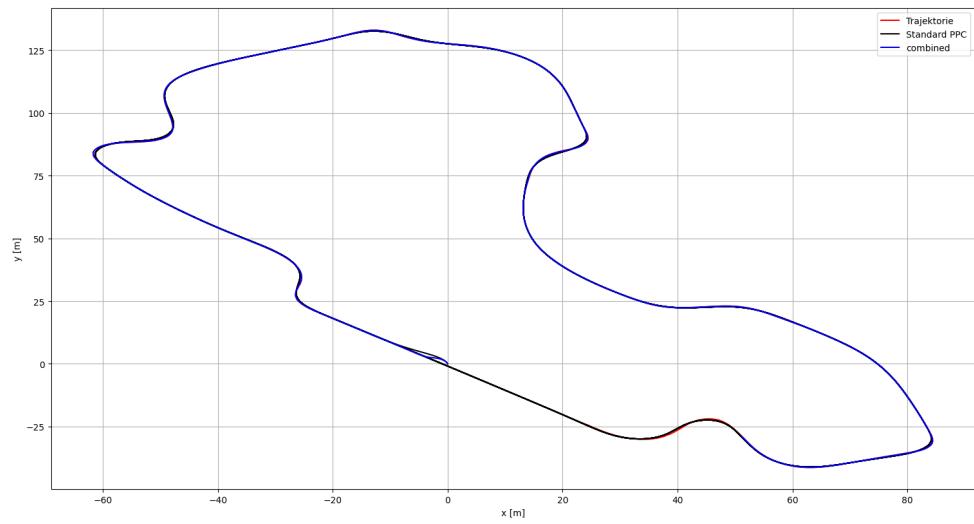
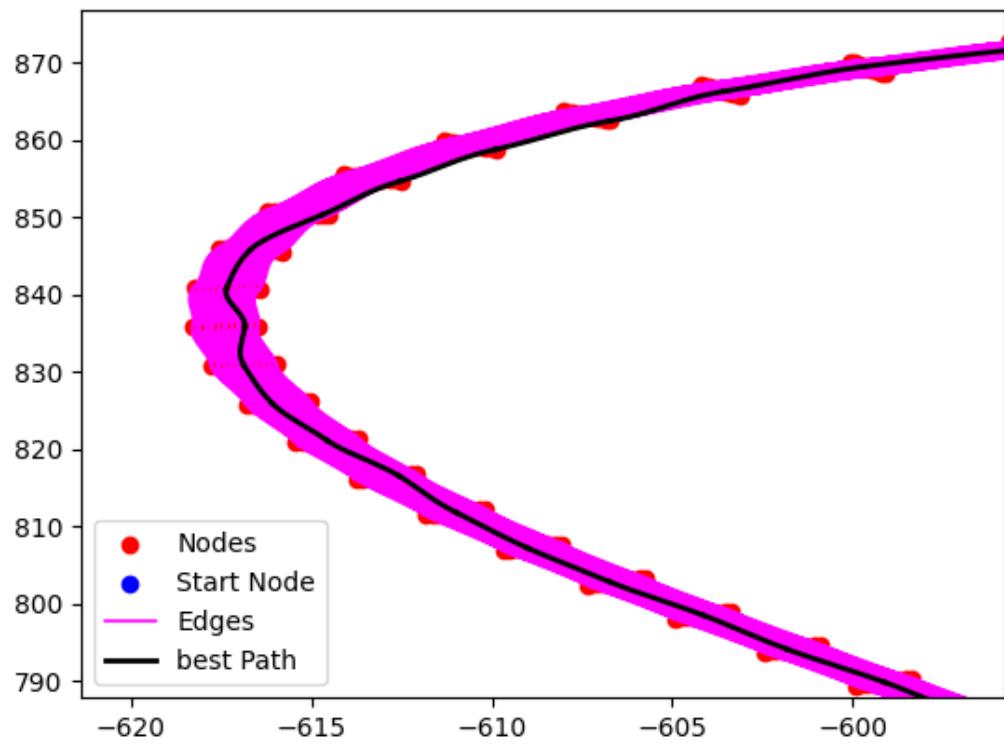


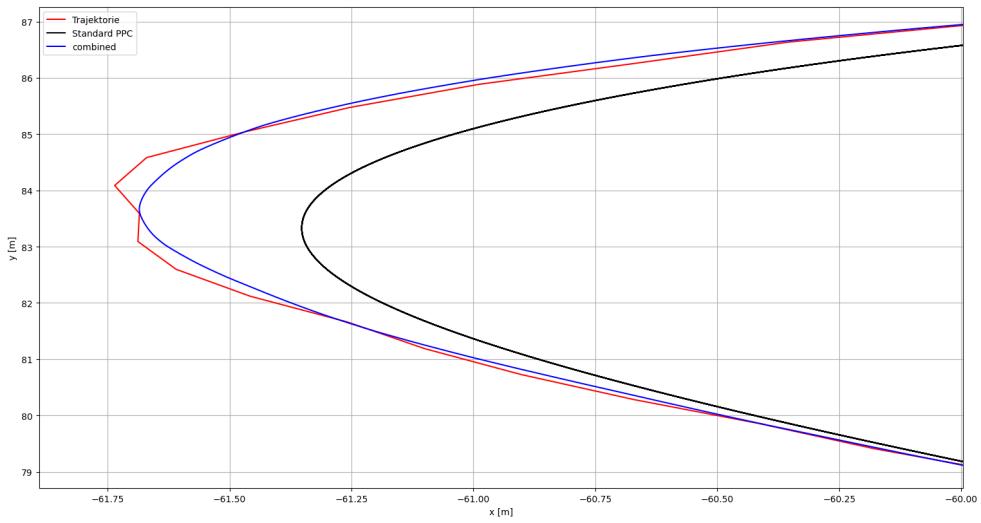
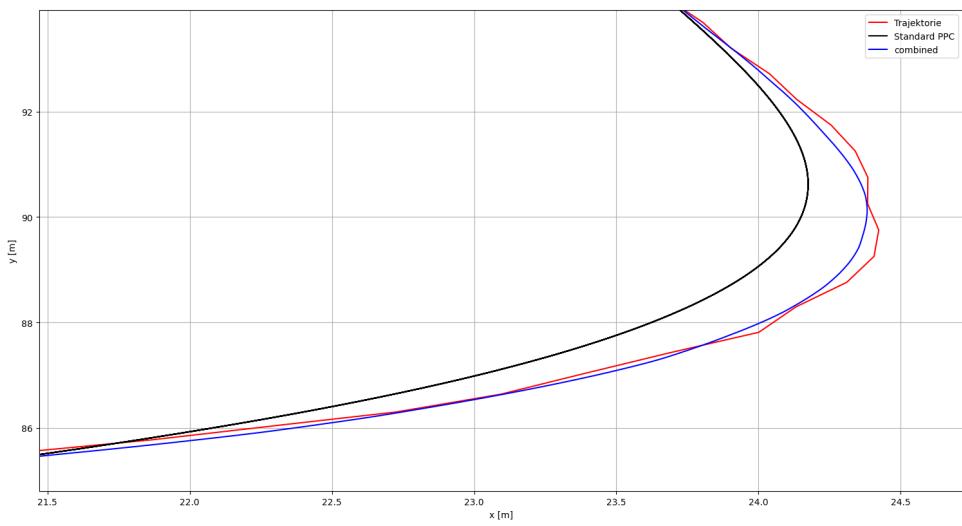


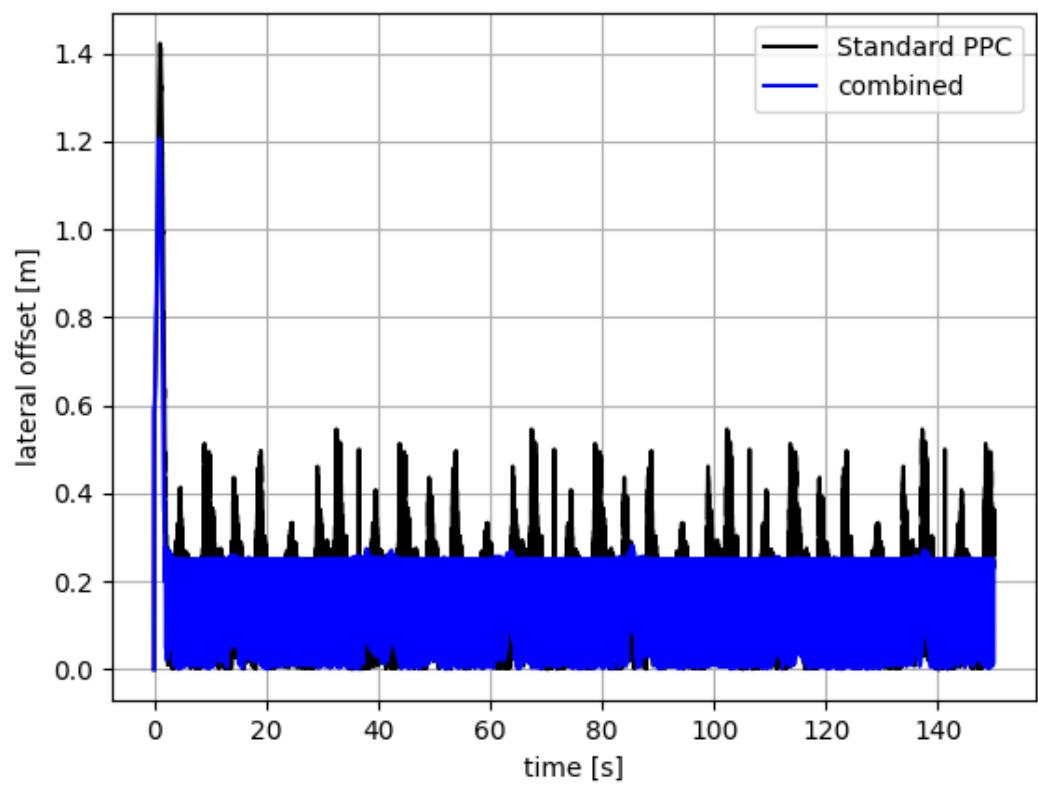
Melbourne:











Berlin:

