



UNIVERSITÀ DI PARMA

UNIVERSITÀ DEGLI STUDI DI PARMA

DOTTORATO DI RICERCA IN
MATEMATICA

CICLO xxxvi

**Model Predictive Planning and Control for Autonomous
Racing, from HPC to Embedded Platforms**

Coordinatore:

Chir.mo Prof. Leonardo Biliotti

Dottorando:

Ayoub Raji

Tutore:

Chir.mo Prof. Marko Bertogna

Anni accademici 2020/2021-2022/2023

Contents

1	Introduction	10
1.1	Background	10
1.1.1	Autonomous Racing on Scaled Platforms	11
1.1.2	Autonomous Racing on Full Scale Vehicles	12
1.1.3	Model-based Planning and Control	14
1.2	Contribution and outline	16
1.2.1	A Racing Game-based Platform to Test Planning and Control Algorithms	17
1.2.2	Frenet frame based Planner for Racing	17
1.2.3	Accurate Path Tracking at the Tire Limits of a Fullscale Race-car using MPC	18
1.2.4	MPC for Planning and Control	18
1.2.5	Execution Analysis on Nvidia Embedded Boards	19
2	Experimental Platforms	20
2.1	Indy Autonomous Challenge	20
2.1.1	Dallara AV-21	20
2.1.2	Race Events	22
2.1.3	The Full Autonomous Stack - er.autopilot	23
2.2	Simulation Environments	25
2.2.1	A Racing Game-based Platform	26
2.2.2	FMU Multi-body Model	27
2.3	Computation Platforms	28

3	Frenet frame based Planner	29
3.1	From Highway to the Racetrack Scenario	29
3.1.1	Trajectories Generation	29
3.1.2	Trajectory Selection	30
3.1.3	Following Mode	32
3.2	Enhancing the Performance	32
3.2.1	Calculation of track limits	33
3.2.2	Calculation of target speed	34
3.2.3	Multi-Step trajectory generation	36
3.2.4	Defensive evasive maneuver	38
3.2.5	Attacker "Right of Way"	39
3.2.6	Track re-entry trajectories	39
3.2.7	Emergency path	40
3.3	Results	41
3.3.1	Static Obstacle Avoidance	41
3.3.2	Head-to-Head Racing	42
3.4	Conclusions	44
4	Model Predictive Control on a Fullscale Racecar	45
4.1	Vehicle Models	45
4.1.1	Curvilinear Single Track Model	45
4.1.2	Locked Differential Tricycle Model	49
4.2	MPC Formulation	53
4.2.1	Single Track MPC formulation	53
4.2.2	Tricycle Model - MPC formulation	54
4.2.3	Locked Differential Model Progression	55
4.2.4	Micro-steps Model Discretization	55
4.3	Separating Low-level and High-level Control	56
4.4	Results	57
4.4.1	MPC Performance under Model Mismatch	57
4.4.2	Highly Accurate Path Tracking at the Tire Limits	62
4.4.3	Exceeding the Tire Limits	67
4.5	Conclusions	68

5	Model Predictive Planning and Control Scheme	70
5.1	MPC for Planning and Control	70
5.1.1	Hierarchical Planning	70
5.1.2	Problem Formulation	71
5.2	Results	72
5.2.1	Keeping Distance and Overtaking on Turn Exit	74
5.2.2	Turn Entry Overtake	77
5.2.3	Overtake on Turn	79
5.3	Conclusions	81
6	Execution Time Analysis	82
6.1	HPC CPU and Nvidia Embedded Platforms	82
6.2	Profiling	84
6.3	Conclusions	87
7	Conclusions and Future Work	88
7.1	Conclusions	88
7.2	Future Work	89
7.2.1	Online Tire Model Estimation	89
7.2.2	MPP-MPC interaction and Multivehicle Free Racing	90
7.2.3	Planning and Perception Uncertainty	90
7.2.4	MPC Acceleration	90

Abstract

Autonomous car racing has emerged as a challenging field, pushing the boundaries of autonomous driving capabilities. The unique demands of racing scenarios serve as a catalyst for developing superior autonomous driving skills, with the aim of surpassing human drivers in handling vehicles under extreme tire limits, and instilling public trust in this transformative technology. This thesis details a comprehensive exploration of motion planning and control methodologies tailored for autonomous car racing, extending the capabilities of Model Predictive Control (MPC)-based approaches and demonstrating their effectiveness in experimental testing on a Dallara AV-21 and in a high-fidelity multibody model simulation.

The research extends the current state-of-the-art in MPC and model-based control for autonomous racecars, showcasing breakthroughs in edge-case scenarios, algorithm testing tools, and performance evaluations on both High-Performance Computing CPUs and embedded platforms. The final proposed pipeline, rooted in hierarchical planning and model predictive control (MPP-MPC) serves as a valuable benchmark for realizing fully autonomous racing scenarios involving multiple agents.

The contributions cover several key facets. First, a simulation platform is described, offering an environment based on a racing game for effective testing, benchmarking, and refinement of planning, control, and forecasting algorithms. This addresses accessibility concerns regarding high-fidelity simulation tools, which is a crucial aspect for making autonomous car racing research even more accessible.

A Frenet frame-based planner extended to the racing domain is presented. This planner, which produces trajectories that minimize jerk, demonstrated effectiveness in oval racing by performing active overtakes at high speed and avoiding static obstacles.

To control the autonomous Dallara AV-21 racecar, an MPC is proposed. The solution achieves highly accurate path tracking, below 20 cm when driving at the tire limits, and introduces novel contributions to vehicle modeling, specifically in capturing combined forces and the yaw moment produced by the locked differential.

A hierarchical planning and control scheme is introduced, leveraging MPC in both modules. Demonstrated in a simulation of the Dallara AV-21 on the F1 Monza track, the approach showcases the capability of overtaking other agents while oper-

ating at the tire limits.

An analysis of MPC execution time on Nvidia embedded boards is conducted, shedding light on the computational demands and demonstrating the potential application of the proposed MPC on low-power and low-resource platforms, a critical aspect of production cars.

Sommario

Le corse autonome stanno emergendo come un ambito di ricerca stimolante, che amplia i confini delle capacità della guida autonoma. Le esigenze uniche degli scenari di gara fungono da catalizzatore per lo sviluppo di capacità di guida avanzate, con l'obiettivo di superare l'umano nella guida al limite delle prestazioni del veicolo e di instillare la fiducia del pubblico in questa tecnologia. Questa tesi riguarda le metodologie di pianificazione e controllo del movimento per le corse automobilistiche autonome, estendendo le capacità degli approcci basati sul Model Predictive Control (MPC) e dimostrando la sua efficacia nei test sperimentali su una Dallara AV-21 e in un ambiente simulativo basato su un modello multibody.

La ricerca della tesi estende l'attuale stato dell'arte nel controllo MPC mostrando alcuni importanti contributi in scenari limite, propone strumenti di test per gli algoritmi e valuta le prestazioni degli approcci proposti sia su CPU ad alte prestazioni che su piattaforme integrate Nvidia. La pipeline finale, basata su una pianificazione gerarchica e nel controllo predittivo (MPP-MPC), funge da punto di riferimento per la realizzazione di soluzioni in grado di guidare autonomamente in scenari di corsa coinvolgendo più veicoli.

I contributi coprono diversi aspetti chiave. Innanzitutto, viene proposta una piattaforma di simulazione, che offre un ambiente di simulazione basato su un gioco di corse per effettuare test, benchmarking e perfezionamento degli algoritmi di pianificazione, controllo e predizione. Ciò risolve i problemi relativi ai costi e alla disponibilità di strumenti di simulazione che includono un modello di fisica altamente accurato, il che è un aspetto cruciale per rendere la ricerca sulle corse automobilistiche autonome ancora più accessibile.

Un planner basato sulle coordinate Frenet è stato esteso allo scenario delle corse. Questo planner, producendo traiettorie che minimizzano il jerk, ha dimostrato la sua efficacia nelle corse autonome su piste ovali eseguendo sorpassi attivi ad alta velocità ed evitando ostacoli statici.

Per il comando dell'auto da corsa autonoma Dallara AV-21, è stato utilizzato un MPC. La soluzione ha dimostrato un tracking estremamente accurato della traiettoria prodotta dal planner, inferiore a 20 cm durante la guida al limite della gomma, e introduce nuovi contributi alla modellazione del veicolo, in particolare nel catturare le forze combinate e il momento di imbardata prodotto dal differenziale bloccato.

Inoltre, viene introdotto uno schema di planning e controllo gerarchico, sfruttando MPC in entrambi i moduli. La soluzione proposta ha dimostrato la sua efficacia in test simultativi della Dallara AV-21 sulla pista di Formula 1 di Monza. In particolare, l'approccio è stato capace di sorpassare altri veicoli controllando il veicolo al limiti degli pneumatici.

Infine, è stata effettuata un'analisi dei tempi di esecuzione del planner e controllore basati su MPC sulle schede embedded Nvidia. Questo studio contribuisce a dimostrare la fattibilità dell'esecuzione di questa metodologia di controllo su piattaforme a basso consumo, il che è una considerazione importante al fine di poter portare potenzialmente queste soluzioni su veicoli di produzione.

Acknowledgement

I express my heartfelt gratitude to my advisor, Prof. Marko Bertogna, for his unwavering belief in me during the initial years of my PhD journey. His guidance and encouragement have been invaluable in shaping the course of my research.

I am deeply indebted to Alex Liniger for allowing me to build on his extensive work on Model Predictive Control (MPC). His time in sharing knowledge and insights has significantly enriched the depth and scope of this thesis and my overall vision of research in the autonomous car racing domain.

A special acknowledgment goes to Nicola Musiu, whose substantial contributions to our collaborative research, particularly in the domain of vehicle dynamics, have been pivotal to the overall success of this work.

I extend my thanks to all the members of the Unimore Racing team, also previously named TII Euroracing, the HiPeRT lab, and Hipert srl.

I thank Alessandro Toschi, Daniele Morra, and Andre Jesus for their significant contributions to tuning and refining the control approach used in this thesis during the race events of the Indy Autonomous Challenge. Additionally, my appreciation goes to Daniele Lombardo for his dedicated efforts in implementing and testing the enhancements for the planning modules designed in this work.

Finally, I want to express my deepest gratitude to my family, especially my parents, for their immense sacrifices and the values instilled in me. I also thank my sister for her support and encouragement.

The collective efforts of these individuals have played a pivotal role in the successful completion of this research, and for that, I am truly thankful.

Chapter 1

Introduction

1.1 Background

In the domain of autonomous car racing, a prevailing assertion assumes that the successful achievement of safe high-speed navigation on a racetrack against other vehicles serves as a foundational step toward the realization of autonomous capabilities in real-world settings. However, this assumption oversimplifies the intricate challenges posed by various scenarios and goals encountered in urban environments. Drawing a distinction between the two contexts is crucial, as the dynamics of race-track scenarios differ significantly from the complexities of navigating through unpredictable real-world conditions.

Therefore, contrary to the notion that autonomous racing directly translates into advancements in real-world autonomous driving, it is essential to recognize the nuanced nature of each domain. Rather than a direct parallel, the merits of autonomous racing lie in its potential to contribute valuable insights and innovations to urban autonomous driving efforts [9].

Much akin to the historical influence of Motorsport on the automotive industry, where advancements in safety and performance technologies originated, autonomous racing emerges as an open research laboratory for the enhancement of autonomous driving capabilities. The underlying premise is that the unique challenges posed by racing scenarios can serve as a catalyst for developing superior autonomous driving skills that are capable of outperforming human drivers in handling the vehicle under tire limits, improving perception capabilities, and fostering public trust in this

transformative technology.

The accessibility of low-cost research racecar prototypes and the increased visibility of prominent autonomous racing challenges, such as F1Tenth, Formula Student Driverless (FSD), Roborace, and the Indy Autonomous Challenge (IAC), has significantly propelled research in this field. In a comprehensive survey presented by Betz et al. [10], an exhaustive examination of autonomous racing cars is carried out. The survey reviews seminal publications that include autonomous driving modules, vehicle modeling, simulation techniques, and holistic software architectures, reflecting the increasing interest and progress in scholarly studies in this evolving field. Most research, particularly in planning and control tasks, has been applied in simulation, scaled vehicles, or FSD.

1.1.1 Autonomous Racing on Scaled Platforms

Miniature race cars, 1:43 size, have been successfully used to demonstrate the capabilities of optimization-based planning and control [51]. The authors presented a two-level design for the path planner and the non-linear model predictive control (NMPC) for path tracking, as well as a controller combining both the modules exploiting contouring control and approximating the nonlinear optimization problem (NLP) using convex quadratic programming (QP). To improve performance, robustness over model uncertainties, and constraint satisfaction, a Cautious NMPC based on a sparse Gaussian Process (GP) approximation using inducing inputs is presented [38], as well as an heuristic based on Randomized MPC exploiting disturbance feedback policies [54]. In [16], a similar approach has been used to demonstrate the real-time application of randomized algorithms. Viability-based approaches have been used to guarantee recursive feasibility through a conservative approximation of the viability kernel, which represents the safe states and controls, to produce a safer driving style [53] or reduce computational time for path planning [52]. One of the latest works on the miniature race car platform shows that a reinforcement learning-based method trained in simulation can be transferred to the real application [18]. After a dedicated tuning of the policy, the solution has been able to achieve lap times similar to the more classical MPC while reducing constraint violations.

The f1tenth platform [62][63], consisting of a 1:10 scale race car equipped mainly with a low power consumption Nivida board and a 2D LiDAR, serves as a first access

point to the field of autonomous driving for students and research laboratories. The main outcome of the platform is the organization of competitions during the most important conferences related to Automation, Control, and Intelligent Vehicles, in which students from universities of all over the world participate. In [34], a Reinforcement Learning (RL) approach is presented, where the authors exploit the trajectory generated by a reliable and classical motion planner to make the agent more robust and facilitate the deployment in the real 11tenth racecar. A recent and relevant work is presented in [6], where a Pure Pursuit path tracking algorithm[19] has been enhanced including the vehicle model knowledge thanks to a Lookup Table that maps a lateral acceleration to the steering angle.

The literature presents several works related to FSD competitions in which a brief description of each module is given [17][60][83][20][3]. A key aspect of these works is the perception module, in particular the detection of cones using Convolutional Neural Networks (CNN) methods in the camera frame [22][87][66]. [44] can be considered the most complete system paper presenting implementation details for the common modules needed to succeed in the FSD events, describing the adopted testing framework and the weak points for future improvements. Extending the work presented in [51], the AMZ Driverless team demonstrated the capabilities of hierarchical optimization-based control [86] and online learning data-driven MPC [43] on an FSD electric vehicle. On the same platform, the latest work demonstrated that a three-layer control architecture [77], with a particular focus on properly tuning the low-level controller to follow the prediction of the mid-level MPC controller, has been able to outperform a professional racecar driver.

1.1.2 Autonomous Racing on Full Scale Vehicles

While small scaled racecars are an ideal platform to reduce costs, make easy the deployment and maintenance, and the FSD can be considered a challenging problem for students and researchers on the design of both the hardware and software, they do not address important aspects of full scale vehicles, such as large racetracks, high-speed and the concerns about safety and risks related to it.

For full-scale racecars, [15] and [8] present their architecture for the Robo-race competition, paying particular attention to the motion planning and control modules, with very limited details on the object detection problem and a limited

top speed. Competing teams presented different solutions to solve the problem of state estimation and localization, exploiting the availability of multiple sensors and LiDAR-based mapping and localization [79][91][57][74]. Work related to multi-vehicle racing on the Roborace platform are limited to a simplified adversarial context at low speed [78] based on multi-layer graph-based planning tested with two vehicles at Monteblando track, and a planning and control approach based on NMPC on simulation and a speed of 40 m/s [14].

Thanks to the IAC, research in this domain has advanced to high-speed head-to-head racing. In [42], the authors presented the software stack of the KAIST team capable of achieving speeds above 230 km / h while aiming for a resilient control thanks to redundancy and a perception-based local emergency controller [50]. MIT-Pitt-RW team presented their software in [73] and [72], where the main contributions are the overall design, strategy, and the focus on processing time of the perception module. In [76], the control scheme is presented discussing the limitations of using a purely linear representation of the tires. The TUM Autonomous Motorsport, one of the best teams amongst the competition achieving a speed of around 74 m/s during an active overtake, describes its software stack in [7]. In particular, an important contribution is to how simulation environments and their setup have been developed and exploited for testing and parameter optimization. Further details of the software stack have been presented in separate works. In [11], an extensive case study is presented, in which the latency of the whole pipeline from perception to control is analyzed in different scenarios. Accurate and reliable sensor fusion for object tracking is presented in [46], where thanks to a delay compensation approach and the usage of an Extended Kalman Filter (EKF) the TUM Autonomous Motorsport team has been able to reduce the measurement residuals and improve the precision. In [40], the problem of assessing the domain shift from simulation to reality for neural network-based LiDAR object detection is presented. The authors found that the precision for models trained in simulation can be improved, including an appropriate noise model. Overall, the mentioned papers on the software stacks give valuable hints and lessons learned, which can potentially benefit also autonomous systems in urban scenarios and field robotics in general. For what concerns control, the approaches vary from LQR [75] [76] to MPC-PI [90] with different methodologies in terms of robustness, safety and model complexity, while for online local planning

the most common solutions are based on the generation and selection of graph-based trajectories through minimization of a cost function [71][42].

Similar works have been published using sportcars and not being associated with any competition. One of the earliest works in this domain is [32], where the authors present a system architecture that focuses on the location, planning of the path and control of an Audi TTS at its limits, including the design of a safety module. The architecture and hardware are based on the one developed for the DARPA Urban Challenge [80], while for the software the focus has been on ensuring a 200Hz hard real-time control which is the same frequency of the localization module. Executing the steering command control at high frequency is particularly important when running at very high speed for the convergence and stability of the tracking path [56].

1.1.3 Model-based Planning and Control

A popular methodology in the design of the control scheme for autonomous racing is to consider the behaviour of the racecar, exploiting models that represent the main vehicle dynamics. Typically, the choice of vehicle model depends on the trade-off between simplicity and accuracy, which is closely tied to the specific application. The most common models in the literature are the point-mass model, kinematic single-track model, dynamic single-track model, double-track model, and multi-body model [2][93]. Whereas in urban scenarios a kinematic single-track model could be sufficient [47][65], in more complex maneuvers that produce not negligible lateral accelerations, which is the common case for the racing scenario, the dynamic single-track model is commonly used where dynamic effects, such as tire forces and additional states such as lateral velocity, are taken into account.

In [45], the model is used in the feedforward part of a feedback-feedforward steering control scheme. The solution demonstrates stability in the path tracking thanks to the introduction of a reference for the vehicle sideslip angle into the feedforward loop, while the feedback steering controller is based on a simple lookahead-based proportional control common in lane-keeping systems[70]. On the same platforms, a slip angle-based controller is presented in [49]. The approach has been applied to track a circular path obtaining a lateral error of 1m without a priori known friction estimate. In [48], the authors present a NMPC for planning and control

cascading models with different complexity in a planning horizon. The solution demonstrated similar performance of using only more complex models and lower median solve time. An interesting approach for active driver assistance is presented in [82], where a NMPC with the single-track model with a Fiala brush tire model, calculates the driving commands and actuates the vehicle when it predicts that the driver will exceed the vehicle limits.

The previously mentioned works demonstrated satisfactory results regarding vehicle stability but, not fully capturing transient dynamics and combined effects, they lack accuracy in path tracking which is important when applying control on a race track and aiming to achieve the optimal laptime [81] and to perform active overtakes while driving at the limit. Modeling the longitudinal weight transfer and lateral brake distribution in a single-track model proved to be effective in controlling a Lexus LC 500 at 95% of the maximum tire force usage using a NMPC[21]. The model is exploited also to estimate the friction in an Unscented Kalman Filter(UKF) which improves the model accuracy. The solution, using the UKF for online estimation of the tire-road friction, achieved an average lateral error of 0.43m and a peak value of 0.77m, whereas in the case of the non-adaptive version are respectively 1.4m and 3.26m. The importance of model fidelity in MPC-based controllers for autonomous racecars at the limit is further demonstrated in [88]. In order to achieve a root mean squared path tracking error of 0.13m and a peak value of 0.47m performing a "Figure 8" maneuver on a Toyota Supra under drifting conditions, dynamic weight transfer and the rear wheelspeed dynamics are modeled. A final simulation study showed that the effect of the tire-road parameters uncertainty is the main remaining aspect that limits the controller tracking performance, demonstrating the sufficient fidelity of the vehicle model.

Accurate vehicle modeling requires a proper familiarity of vehicle dynamics, modelling tools, and parameter estimation. To cope with that, the principal alternatives are simplified model-based approaches relying on robustness over performance and learning-based control. In [90], a Tube-MPC approach applied to the IAC on TUM Autonomous Motorsport racecar is presented. The proposed solution lets the controller freely optimize the trajectory that should be followed within an uncertainty tube. Despite the usage of a simplified point-mass model, thanks to the split of the dynamics into certain ones managed by the high-level control and un-

certain handled by the introduction of low-level PID loops, the controller achieved speeds of over 70 m/s at LVMS maintaining a path tracking error slightly below 1m. Learning MPC is another option to cope with model uncertainties. In [92], the authors exploited the direct regression of local affine dynamics into a Learning MPC on the Dallara AV-21 on the Putnam Park Road Course. Through iterative learning, the controller safely improved the mean lap time at each lap and reached a speed of 20 m/s.

1.2 Contribution and outline

The works presented in this thesis extend the actual state-of-the-art on MPC and model-based control of autonomous racecars, producing novel results in edge-case scenarios, proposing tools to test the algorithms, and evaluating the performances on both High-Performance Computing CPUs and embedded platforms.

A great part of the formulations, figures, and results presented are produced from the following published papers and co-advised theses.

- **A. Raji**, A. Liniger, A. Giove, A. Toschi, N. Musiu, D. Morra, M. Verucchi, D. Caporale, and M. Bertogna. "Motion planning and control for multi vehicle autonomous racing at high speeds." In 2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC), pages 2775–2782, 2022, [68].
- **A. Raji**, N. Musiu, A. Toschi, F. Prignoli, E. Mascaro, P. Musso, F. Amerotti, A. Liniger, S. Sorrentino, M. Bertogna. "A Tricycle Model to Accurately Control an Autonomous Racecar with Locked Differential." In 2023 IEEE 11th International Conference on Systems and Control (ICSC), 2023, [69].
- **A. Raji**, D. Caporale, F. Gatti, A. Giove, M. Verucchi, D. Malatesta, N. Musiu, A. Toschi, S. R. Popitanu, F. Bagni, M. Bosi, A. Liniger, M. Bertogna, D. Morra, F. Amerotti, L. Bartoli, F. Martello, and R. Porta. "er.autopilot 1.0: The full autonomous stack for oval racing at high speeds." Field Robotics, 4:99–137, 2024, [67].
- N. Musiu, "Identificazione e validazione di modello multibody di Dallara F308 e sintesi di single-track per applicazioni Autonomous Racing", UNIMORE,

Master Thesis, 2022, [59].

- A. Toschi, "Integration of Model Predictive Control for autonomous racing", UNIBO, Master Thesis, 2022, [84].
- D. Lombardo, "Motion planning algorithms for autonomous racing in multi-vehicle scenarios", UNIMORE, Master Thesis, 2023, [55].

In the following, the contributions of the thesis are presented, including a brief outline.

1.2.1 A Racing Game-based Platform to Test Planning and Control Algorithms

In Chapter 2, the experimental platforms used are presented, including an introduction to the race competitions, the racecar, the full software stack, and the simulation environments. In particular, a simulation platform based on a racing game is proposed in subsection 2.2.1. The simulation environment offers an accurate and user-friendly solution for testing, benchmarking, and refining planning, control, and forecasting algorithms for autonomous car racing. In fact, the usage of high-fidelity multibody model-based simulators, such as rFPro, VI-Grade, or the custom one based on Dymola[24] described in subsection 3.1.1, is important in order to reduce as much as possible the mismatch between simulation and reality, which is crucial if the motion planning and control modules are intended to be used in a fully autonomous racecar like the Dallara AV-21. However, this comes with substantial costs and time effort, making it inaccessible to smaller research groups. In addition, the use of such simulators often requires a deep understanding of vehicle dynamics to ensure realistic modeling. Open-source or low-cost alternatives are not appropriate for the racing scenario [26], limited for scaled racing [4][5] or for only a few racecars and tracks[27][37]. These limit the possibility of testing and benchmarking algorithms under different conditions.

1.2.2 Frenet frame based Planner for Racing

In Chapter 3, we present a Frenet-Frame-based method for local planning that extends the original work of [89] adapting it to the racing environment and demonstrating active overtakes at 230 km/h and avoidance of static obstacles on oval

tracks. The strength of the solution relies on creating trajectories that minimize the lateral jerk and the ease of adjusting the weights of the cost function. Other more complex graph-based solutions are presented in [78] and [71] showing, respectively, the efficacy in a simplified scenario in a Roborace car and on oval tracks in a Dallara AV-21, without any evident result of the benefits of the complex design in more difficult scenarios, such as overtakes on curves or with more than two moving vehicles.

1.2.3 Accurate Path Tracking at the Tire Limits of a Fullscale Racecar using MPC

In Chapter 4, we present the MPC developed to control the autonomous Dallara AV-21 of the TII UNIMORE Racing team (also known previously as TII Euroracing) participating in the IAC events on different oval tracks and on the F1 Monza road course track. The solution extended the work presented in [86] with more complex models and the integration with a local planner demonstrating its ability to reach high speeds and lateral accelerations even under model mismatch. After further improvements in fitting the model parameters, the introduction of novel contributions to the models, and their integration into the MPC, the solution demonstrated highly accurate path tracking, lower than 20cm, at the tire limits. Furthermore, we present some results on how the controller behaves when exceeding tire limits and how direct human interference over autonomous commands can end in a major crash.

1.2.4 MPC for Planning and Control

In Chapter 5, we present a hierarchical approach to planning and control where, with appropriate adjustments, the MPC is exploited in both tasks. In fact, graph-based and sampling-based methods for local planning have some limitations related mainly to neglecting vehicle dynamics and inefficiency in managing complex track layouts. Using an optimization-based approach for planning helps to solve these issues and makes the coupling with the controller more efficient. MPC-based planning solutions, available in the literature for autonomous racing, are limited to static obstacle avoidance [1] or simulation at low speed [14]. In the solution proposed in this thesis, it has been decided to make the planning module more hierarchical,

including a high-level overtake logic that can find the feasible running tube to be used as a track constraint in the MPC Planner (MPP). The MPP uses an offline generated global trajectory as a reference, and limits its velocity profile over a long horizon following the longitudinal and lateral acceleration bounds set as physical limits or requested online by the user. The final produced trajectory is followed by an MPC dedicated solely to track the path and speed profile. The proposed solution is tested in a simulation of the Dallara AV-21 on the F1 Monza track demonstrating the capability of overtaking other agents while driving at the limit of the tires.

1.2.5 Execution Analysis on Nvidia Embedded Boards

The computational efficiency of MPC plays a critical role in determining its suitability for real-time applications. Traditionally, MPC research and applications have predominantly focused on platforms equipped with high-end CPUs, typically found in HPC environments, such as those powered by Intel Xeon processors.

In Chapter 6, we explore the viability of implementing the proposed MPC on less powerful platforms by analyzing its computational demands on Nvidia embedded boards. This shift in focus is particularly noteworthy as it delves into the adaptability of MPCs to low-power and low-resource platforms, characteristics common in road cars. Furthermore, it addresses the growing trend where commercial vehicles like the Lotus Eletre and the Xiaomi SU7 are equipped with Nvidia AGX Orin boards to deploy autonomous driving features. This transition underscores the evolving landscape of MPC applications, extending beyond traditional high-performance computing environments to embrace the constraints and opportunities presented by emerging technologies in the automotive sector.

Chapter 2

Experimental Platforms

2.1 Indy Autonomous Challenge

The IAC¹ is an international competition that brings together public-private partnerships and academic institutions to challenge university students around the world to imagine, invent and prove a new generation of automated vehicle software to run fully autonomous race cars.

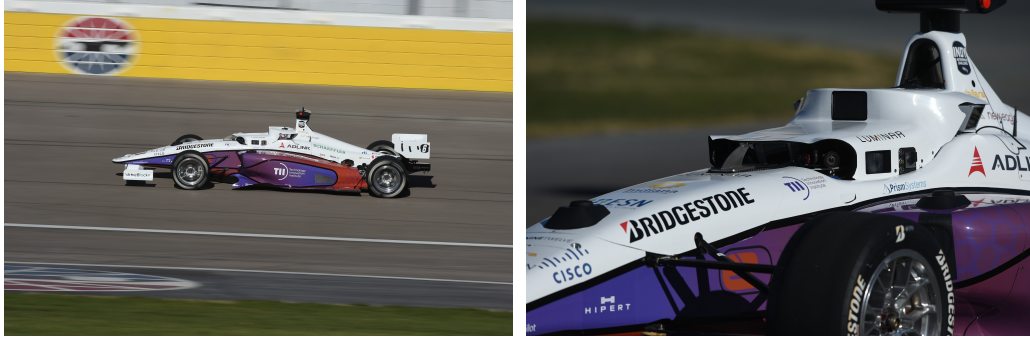
The challenge was carried out in two steps: a simulation race, and a real race. Of the 30 teams from universities all around the world that participated in this competition, only 9 passed the simulation step. The first race, the Indy Autonomous Challenge powered by Cisco, was held on October 21st 2021 at the Indianapolis Motor Speedway (IMS), and the second one, the Autonomous Challenge @ CES, on January 7th 2022 at the Las Vegas Motor Speedway (LVMS) during the Consumer Electronics Show (CES). The car shakedown and a considerable part of the development before the IMS race was conducted at Lucas Oil Raceway (LOR). Other races and events have been organized in the following years at the Texas Motor Speedway (TMS), LVMS, and Monza.

2.1.1 Dallara AV-21

Each team of the IAC participates with a Dallara AV-21, shown in Figure 2.1, a fully autonomous open-wheel race car based on the official Indy Lights IL-15. Unlike the original race car, the engine mounted is a turbo-charged Honda K20 with 390

¹<https://www.indyautonomousschallenge.com/>

horsepower. The mechanics, suspensions, and aerodynamics are adjusted for oval racing and high banked tracks with an asymmetrical setup.



(a) AV-21 with the TII EuroRacing livery. (b) Close view on the onboard sensors.

Figure 2.1: Dallara AV-21

On the perception side, the car is equipped with two GNSS modules, three LiDARs, six cameras (two cameras in a stereo setup and four to cover the 360 range), and three RADARs. The Novatel GNSS Pwrpak 7d receivers provide a centimetre precision localization of the car thanks to four antennas and RTK correction. The three solid-state LUMINAR H3 LiDARs have a range of 200m and they operate at 20 Hz. The Aptiv ESR 2.5 frontal radar and the MRR side radars have a range of around 160m and they provide the detected obstacles at 10 Hz. The six high-resolution RGB Mako G319C cameras from Allied Vision are mounted to have a view of almost 360 degrees around the car. The computing platform used on the vehicle is an ADLINK AVA-3501 consisting of an 8 core Intel Xeon E 2278 GE CPU, an NVIDIA RTX Quadro 8000 GPU, and 64 GB DDR4 RAM. For external communication, Cisco FM-4500 radio transceivers are used in the car and around the tracks in order to make telemetry data available to the crew team in the pitlane. Race Control signals can be exchanged by means of a MYLAPS RaceLink system mounted on the racetracks and a transponder mounted on the vehicle.

On the actuation side, the car has a Drive-by-Wire (DBW) system realised by Schaeffler to actuate the steering, the throttle pedal, the brake pedal, and the gearbox. The New Eagle GCM 196 Raptor control module is used as an interface between the DBW system, the computing platform in which the algorithms are executed, and the other units related to the engine and minor subsystems.

The IAC updated the vehicle in 2022 with a refreshed version of the AV-21,

increasing the engine power to around 500hp to achieve speeds higher than 300 Km/h, updating the computing platform with a dSpace AUTERA equipped with an Intel Xeon Processor D-2166NT (2.3GHz) CPU, NVIDIA RTX A5000 GPU, 128GB RAM, and an additional IMU/GNSS source consisting of a Vectornav VN-310.

A further improvement, the Dallara AV-24, has been announced at the CES 2024, including a new customized actuation package, four LUMINAR IRIS LiDARs including a unit on the back side, and Continental ARS540 Radars.

2.1.2 Race Events

The race at IMS was a solo time trial competition, that consisted of a semi-final and final event. In order to get access to the race, the teams had to demonstrate a set of requirements during testing. This event, besides the time trial, included an obstacle avoidance challenge: two static obstacles were placed in the front stretch, to prove that the car was capable of actively avoiding static obstacles. The final leg was limited to the three teams that achieved the best score in the semi-finals. The entire run was formed of four warm-up laps and two performance laps. The winner was determined based on the highest average speed achieved during the two consecutive performance laps. TUM Autonomous Motorsport, with a final average speed of 60.77 m/s, held the first position despite a better lap by our team, TII Euroracing, due to a bug in counting the number of laps which ended us to slowing down and performing the cooldown lap earlier during the last performance lap, achieving therefore only the second place.

The race at LVMS consisted in a Passing Competition, where multiple rounds of head-to-head matches were conducted by two cars that had to take turns playing the role of defender and attacker, attempting to overtake at increasing speeds, until one or both cars were unable to complete a pass. In each round, the attacker had to follow the following four steps:

1. Reduce the gap with the defender.
2. Keep a longitudinal safety distance.
3. Overtake the defender once reached a passing zone.
4. Switch the role to defender and reduce the velocity to a pre-determined constant value.

A time trial event was created to determine the teams' seeding in the brackets of the Passing Competition, ending up with the PoliMove Autonomous Racing team achieving the fastest lap (32.54s), TII EuroRacing second (33.99s) and TUM Autonomous Motorsport third (35.3s). In the Passing Competition, we faced TUM for the Semifinal of the event. We've been able to pass the rounds up to the defending speed level of 58 m/s overtaking at 63 m/s (226.8 km/h). An edge case for the perception stack and the motion planning occurred during the round at the defense speed of 60 m/s due to a false detection of a standing obstacle by the Radar as it is described in [67]. This resulted in a third place for us, second place for TUM Autonomous Motorsport, and first place for Polimove Autonomous Racing who has been able to pass the round of 67 m/s defence speed.

While for the LVMS race the defender had to keep an inner line, additional complexity has been added for the third race at TMS, where the cars could keep a free raceline that had to be accepted by race control prior to the event. In this way the detection and forecasting play a big role in the success of an overtake. The overall performance during this event has been limited by the weather conditions, achieving just a defensive speed of 44.4 m/s during the semifinal round between us and Polimove, being the latter the winner of the race, with AI Racing Tech second.

In June 2023, the IAC had its first event in a non-oval racetrack. During the MIMO 2024 event at the Monza F1 circuit, five of the autonomous racing teams competed in six sessions over three days for a time trial competition. Polimove won the race with a lap time of 2:05.87, TUM achieved the second place in 2:08.66 and TII Unimore Racing third in 2:11.24.

2.1.3 The Full Autonomous Stack - `er.autopilot`

The `er.autopilot` 1.0 software stack used by team, consists of multiple modules following the Perceive-Plan-Act paradigm. In Figure 2.2, a block diagram with a high-level overview of the modules is shown.

A Localization module produces the state estimate of the ego vehicle used by all the other modules. It has been decided to implement a robust localization filtering scheme based on an Extended Kalman Filter (EKF) and to develop a LiDAR-based localization system as an additional source to the EKF to further enhance the system's robustness in case the GNSS signal is lost. This has been

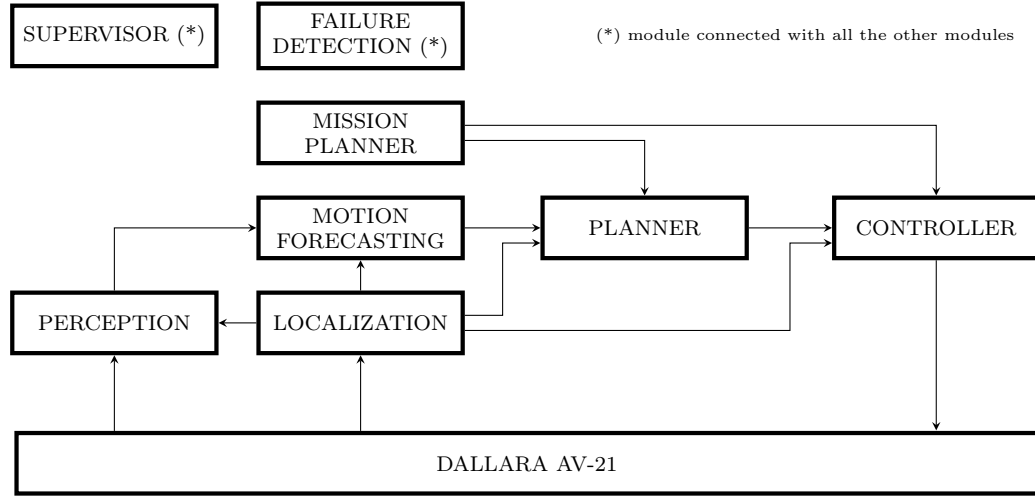


Figure 2.2: Diagram block with the software modules of `er.autopilot`. The modules marked with an asterisk are connected with all the other modules.

done considering that the RTK correction and the GNSS signal are not always reliable. The localization module is executed at a frequency of 125Hz, the same of the IMUs which are the fastest source, to guarantee a tight correspondence between the car's position on track and its latest available estimate, considering that the racecar can reach a maximum velocity of around 300 km/h.

By getting the data from the sensors of the AV-21, the Perception stack is able to detect the other vehicles and objects in the environment. This information is used mainly by the Motion Forecasting and Planner modules to predict the opponent's movement and generate a local path to avoid the collision or perform an overtake. The Motion Forecasting consists of a Kalman filter with a model defined in a Frenet frame where the longitudinal speed of the obstacle and its lateral displacement from a certain reference are constant. The local planner is an extension of [89], as it computes the trajectory generation in a Frenet coordinate frame, where some adjustments have been implemented to satisfy the needs of our racing scenario. A Mission Planner is integrated into the software stack as a behavioral planner able to get the signals from Race Control and from an internal state machine in order to give high-level decisions to the Planner and Controller, such as entering or exiting the pitlane and starting to overtake the opponent. Lastly, the Controller module is the one responsible to generate the correct actuation commands to be sent to the

vehicle. During the race events different controllers have been used. For the events on ovals, a modified version of the Pure Pursuit path tracking algorithm [19] with a PID longitudinal controller has been used at low speed while for high speeds a Model Predictive Controller (MPC) with a dynamic model in curvilinear coordinates is executed. For the most recent race in Monza, it has been used constantly the MPC blending a kinematic model with the dynamic one.

The modules, represented as nodes, communicate via the ROS2 framework and Eclipse Cyclone DDS. Nodes are compiled into a shared library for runtime loading, enabling execution in separate processes or as a single process. A foundational class for the control encompasses essential callbacks for vehicle state reception from the localization node and actuation commands feedback, along with common methods. Each implementation extends this base class with additional callbacks, methods, and interfaces to required libraries. Notably, the team has chosen to use ROS2 nodes as communication wrappers, separating algorithmic components into standalone software interfacing with these nodes.

All vehicle-executed code is written in C++ or generated from Matlab Simulink using the C-code generation tool. Python and Julia are used for offline scripts, covering data analysis, trajectory optimization, refinement, and visualization. To streamline deployment on the vehicle, developers' machines, and the online GitLab pipeline for basic testing, a Docker container has been developed.

2.2 Simulation Environments

In order to assess the integrity of the entire software stack, our team employed a lightweight Unity-based simulator, which mirrors the interface of the actual vehicle, ensuring simplicity in installation and usage. To evaluate the Motion Forecasting and Planning algorithms comprehensively, a specialized module capable of simulating moving vehicles was incorporated.

In the context of autonomous racing, the efficacy of a simulator is contingent upon several critical criteria. These encompass the fidelity of Vehicle Dynamics, the Simulation to Reality gap in terms of interfaces and sensors, and user-friendliness. While the Unity-based simulator adeptly addresses the latter two considerations, it falls short in guaranteeing precise fidelity with respect to the underlying physics.

Given the paramount significance of accurate vehicle dynamics for the Planning

and Control modules, our research and development endeavors necessitated the utilization of two additional simulation environments over the course of several years. These environments were specifically chosen to enhance the accuracy of vehicle dynamics representation and to complement the Unity-based simulator's strengths in interface and sensor simulation.

2.2.1 A Racing Game-based Platform

Assetto Corsa, developed by Kunos Simulazioni², emerges as an optimal choice in the absence of an open source simulator that seamlessly combines superior physics, flexible adaptability in changing car and track conditions, and user-friendly accessibility. This scarcity of alternatives accentuates the significance of exploiting Assetto Corsa not only for our specific application, but also as a versatile platform for motion forecasting and benchmarking diverse approaches. The simulator provides a Python interface, facilitating the real-time retrieval of intricate data pertaining to active vehicles, encompassing details such as position, velocities, accelerations, tire dynamics, and aerodynamics.

In our pursuit of refining system integration, we fashioned additional interfaces to transmit actuation commands and devised ROS2 wrappers, ensuring the synchronization of messaging protocols with those of the authentic system. Commencing with bespoke modifications of the Dallara IL-15 and available online oval tracks, our efforts centered on aligning the car model intricacies to mirror the engine map, configuration, and tire model of the Dallara AV-21.

Within this simulation environment, we subjected our Motion Planning and Control algorithms to rigorous testing. This setting affords the facile creation of diverse and challenging scenarios, allowing the manipulation of parameters such as road friction coefficients, car configurations and stability, wind dynamics, slipstream effects, and the introduction of multiple AI or human-driven agents.

To orchestrate this simulation, a dedicated Windows machine manages the racing game and publishes ROS2 messages. Concurrently, a distinct Linux-based machine executes a version of the `er.autopilot 1.0` software. This configuration entails the deactivation of specific nodes associated with Perception, along with the adjustment of parameters pertaining to Race Control and other communication

²<https://www.assettocorsa.it/>

facets not replicated within the simulation environment.

Further details, public release of the UDP/TCP/ROS interfaces, and the final contribution of this simulation platform will be presented in a separate work.

2.2.2 FMU Multi-body Model

Due to the lack of a steering wheel in the research vehicle, the traditional maneuvers used to collect data for the identification of the vehicle model were not practical [28, 25]. We relied on information provided by the IAC organizers and tire and vehicle manufacturers, initially limiting our model to a static identification.

Dymola, a specialized tool for physical modeling and simulation [24], has been instrumental in the meticulous modeling of the AV-21 vehicle dynamics. This modeling process was facilitated by the integration of the VeSyMA - Motorsports Library [23], which offers tailored solutions for modeling various components of open-wheel race cars, including suspensions, aerodynamics, tires, and powertrains.

A comprehensive multi-body simulation of the AV-21 vehicle was crafted, leveraging available information on mechanical components, such as static parameters of the Indy Lights chassis and the engine map obtained from a dedicated test bench. To account for unknown components and potential suspension setup variations, estimates were derived from Dallara IL-15 and IndyCar oval configurations, with specific attention given to parameters like camber, caster, and toe.

The simulation also encompassed the modeling of IMS and LVMS tracks in Dymola, with the banking angles estimated from data sourced from sim-racing games. Subsequent validation of these modeled tracks was conducted using data acquired from on-board LiDAR sensors, ensuring a high level of fidelity in the representation of the real-world tracks.

In the software developed for the MPC, it is included an internal standalone simulation which employs the same model used by the controller. This environment serves as an ideal testing ground to verify the fundamental correctness of the algorithm. For enhanced physical accuracy, the Dymola multibody model has been integrated into the standalone MPC simulation, leveraging functional mock-up interface (FMI) interfaces [13][12] to achieve this integration.

2.3 Computation Platforms

The computing platforms used in the Dallara AV-21, namely ADLINK AVA-3501 and dSpace AUTERA, incorporate powerful CPUs, GPUs, ample RAM, and disk space. These attributes render them particularly well-suited for the preliminary testing phases of autonomous driving algorithms. However, it is noteworthy that these platforms, characterized by high power consumption and spatial demands, may not align with the constraints inherent in real-world applications. The latter often necessitate the integration of compact embedded boards within the confines of vehicular environments.

In light of this consideration, it has been decided to evaluate the efficacy of Planning and Control algorithms, specifically those rooted in Model Predictive Control (MPC), on Nvidia Embedded boards. For this evaluation, it has been used the Nvidia Xavier NX and the Nvidia Xavier AGX.

For a detailed overview of the specifications of all the computation platforms used, refer to Table 2.1.

	AVA-3501	AUTERA	Xavier NX	Xaver AGX
CPU	Intel Xeon E 2278 GE CPU (up to 4.7GHz)	Intel Xeon D-2166NT (up to 2.3GHz)	4-core ARM A57 (up to 1.9GHz)	6-core ARM V8.2 (up to 2.2GHz)
GPU	NVIDIA Quadro RTX 8000	Nvidia Quadro RTXA5000	384 Cores Volta	512 Cores Volta
Memory	64GB	128GB	8GB	32GB
Storage	4TB	16TB	16GB	64GB
Power	Unknown	130-520W	Up to 15W	10-30W
Size	210 x 210 x 350 mm	330 x 376 x 156 mm	69.6 x 45 mm	87 x 100 mm

Table 2.1: Specifications of ADLINK AVA-3501, dSpace AUTERA, Xavier NX, and Xavier AGX.

Chapter 3

Frenet frame based Planner

Graph-based planners demonstrated their effectiveness for autonomous driving on the road and highway [89][58][35][39]. The main advantages of this approach are the guarantees on the determinism of the final solution due to the spatial or spatiotemporal limits imposed in the creation of the graph, the ease in managing the cost function related to the smoothness of the trajectories, and the potential acceleration of the algorithms on GPUs. Ensuring smooth trajectories is crucial, especially for autonomous racecars operating at high speeds. This becomes even more significant on oval tracks due to the elevated lateral accelerations and banking involved.

3.1 From Highway to the Racetrack Scenario

The planner module presented in this chapter is an extended version of [89] that adds further considerations to the collision check of moving obstacles and adapts it from the highway to the racing scenario.

3.1.1 Trajectories Generation

Given a race line computed offline, a Frenét frame is defined and used to generate multiple trajectories, which for example merge to the reference line, follow a vehicle or perform an overtake. Each single trajectory is defined as a combination of a lateral movement $n(t)$ and a longitudinal movement $s(t)$ at time t with respect to the reference path.

Starting from the lateral movements, let $N_0 = [n_0, \dot{n}_0, \ddot{n}_0]$ be the start state and $N_1 = [n_1, \dot{n}_1, \ddot{n}_1]$ be the end state. As we want to move parallel to the reference line, we generate the set of lateral movements by changing n_1 in an interval $[n_{\min}, n_{\max}]$ and set $\dot{n}_1 = \ddot{n}_1 = 0$. Given N_0 , N_1 and the time interval T between them, a quadratic polynomial is fully defined, and its coefficients can be calculated. For each lateral movement, we then assign a cost based on the following cost function:

$$C_{\text{lat}} = k_j J_t(n(t)) + k_t T + k_d n_1^2.$$

This cost function penalizes the solutions with slow convergence to the reference, i.e. the ones that at the end of the trajectory are off from the reference path $n = 0$. Unlike what is proposed in [89], we decided to keep T constant for all the trajectories in the set in order to provide to the controller a trajectory with a fixed time horizon.

A similar approach has been used for the longitudinal movement generating trajectories that bring the car to a desired velocity $\dot{s}_n$, while minimizing the jerk. As shown in [89], quartic polynomials can be found to minimize the cost function

$$C_{\text{long}} = k_j J_t(s(t)) + k_t T + k_s (\dot{s}_1 - \dot{s}_n)^2 \quad (3.1)$$

for a given start state $S_0 = [s_0, \dot{s}_0, \ddot{s}_0]$ at t_0 and $[\dot{s}_1, \ddot{s}_1]$ of the end state S_1 at some $t_1 = t_0 + T$. This means that we can generate a set of optimal longitudinal trajectories by varying the end constraints $\dot{s}_1 = \dot{s}_n + \Delta \dot{s}$ and T .

The set of lateral movements $\mathcal{T}_{\text{lat}}$ and longitudinal movements $\mathcal{T}_{\text{lon}}$ are then combined, resulting in a set $\mathcal{T} = \mathcal{T}_{\text{lat}} \times \mathcal{T}_{\text{lon}}$ of complete trajectories.

3.1.2 Trajectory Selection

All the trajectories $\tau_i \in \mathcal{T}$ are checked to evaluate whether they exceed the track boundaries or collide with an obstacle. We decided to perform these checks in the Frenét frame, to avoid converting the trajectories to a Cartesian frame. Furthermore, rather than doing the checks on the polynomials, we sampled each trajectory in a finite number of points. The sampling is done by fixing a time interval Δt and evaluating the trajectory in $\Delta t, 2\Delta t, \dots, M\Delta t$. Thus, the trajectory is converted to

a set of points, where each point is associated with a time instant:

$$\tau_i \rightarrow \bar{\tau}_i = \left\{ \{t_k, (s_{\tau_i,k}, n_{\tau_i,k})\}, \quad k = 1, \dots, M \right\}.$$

Given the track width in every point of the reference path, it is trivial to check if a trajectory $\bar{\tau}_i$ goes out of the track boundaries.

From the Motion Forecasting module, an obstacle $\mathcal{O}_j$ is defined as a set of points. Each point is associated with a time instant:

$$\mathcal{O}_j = \left\{ \{t_k, (s_{o_j,k}, n_{o_j,k})\}, \quad k = 1, \dots, M \right\}.$$

To account for the safety margins, a rectangle is built around every point of the predicted trajectory of the obstacle. A trajectory $\bar{\tau}_i$ collides with an obstacle $\mathcal{O}_j$ if $\exists k \in \{1, \dots, M\}$ such that $(s_{\tau_i,k}, n_{\tau_i,k})$ is inside the rectangle built around $(s_{o_j,k}, n_{o_j,k})$. The described collision check treats the obstacle as a hard constraint for the planning algorithm. This approach could lead to undesired behavior in the scenarios in which an obstacle is blocking the race line, because the best trajectory will be the nearest one to the obstacle that does not collide with the obstacle itself. Following such a trajectory could bring the car to the edge of the obstacle safety margins. This can be critical if we consider the possible noise in the detection module. To overcome this issue, the collision check method is improved by adding a soft constraint. For each trajectory τ_i , a collision coefficient $\gamma_i \in [0, 1]$ is computed, where $\gamma_i = 0$ indicates that the trajectory is not colliding with any obstacle, whereas $\gamma_i = 1$ indicates that the trajectory is violating the safety margins (hard constraint). Given this change, the cost becomes

$$C_{\text{tot},i} = k_{\text{lat}} C_{\text{lat},i} + k_{\text{lon}} C_{\text{lon},i} + k_{\text{soft}} \gamma_i^2,$$

with $k_{\text{lat}}, k_{\text{lon}}, k_{\text{soft}} > 0$.

To compute γ_i we decided to exploit the Euclidean distance from the safety margin. For every trajectory τ_i the minimum distance n_i from the safety margin is computed. Then, γ_i is defined as

$$\gamma_i = \max \left\{ 1 - \frac{n_i}{\Delta_{\text{soft}}}, 0 \right\}, \quad (3.2)$$

where $\Delta_{soft} > 0$ is a parameter to enlarge or reduce the effect of the soft constraint. In Figure 3.1 a graphical representation of the soft constraint is given.

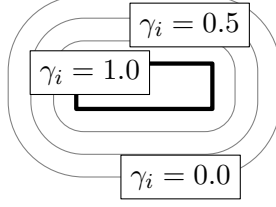


Figure 3.1: Graphical representation of the collision coefficient as defined in (3.2). The black rectangle indicates the safety margin from the obstacle (hard constraint).

The final step of our planner is to select the trajectory with the minimal cost which does stay inside the track margins.

3.1.3 Following Mode

A following mode is implemented in the planner to keep a desired distance to the opponent when it is not allowed to perform an overtake by the rules of the competition. Differently from [89], given the position $\mathbf{P}_{opp} = [s_{opp}, n_{opp}]$ and speed $\dot{s}_{opp}$ of the car to keep the distance from and the desired distance Δ_{des} , the desired speed of the ego car $\dot{s}_{des}$ is regulated by a simple proportional controller:

$$\dot{s}_{des} = \dot{s}_{opp} - k (\Delta_{des} - \Delta_{gap})$$

where $\Delta_{gap} > 0$ is the current distance between the ego and the opponent car, and $k > 0$ is the gain of the controller. The computed desired speed is then used in (3.1) for the longitudinal movement generation.

3.2 Enhancing the Performance

This chapter will provide a description of the extensions made in the motion planner module with respect to the previous version. In particular, the improvements made are listed here:

- *Calculation of track limits:* A feature implemented to compute the track limits before generating the set of trajectories to limit the range in which the

trajectories will be generated.

- *Calculation of target speed:* The approach employed for calculating the desired speed to achieve in order to generate appropriate longitudinal motion, especially when the autonomous vehicle needs to follow the leading vehicle without passing it.
- *Greater path horizon:* A feature that allows generating longer path horizons without over-shrinking the generated graph of trajectories, resulting in better management of overtakes when the opponent moves from side to side in front of ego.
- *Management of emergency cases:* A set of methods implemented to handle unusual scenarios and emergency cases, trying to maintain a safe and competitive trajectory, while continuously feeding the controller with a valid path to follow.

In the following, each of these improvements is detailed.

3.2.1 Calculation of track limits

The set of trajectories is generated within a specific lateral range, referred to as $[d_{min}, d_{max}]$. This range is determined by the layout of the track in the longitudinal direction and the desired length of the planning horizon. In addition, to account for the size of the vehicle and ensure a safe distance from the track boundaries, two margins, m_i and m_o , are defined. The values of d_{min} and d_{max} represent the minimum distance from the outer and inner borders, respectively, related to the final position of the planning horizon s_f .

To compute the final position of the planning horizon s_f , the desired constant velocity, $\dot{s}$, must be taken into account when computing the longitudinal movement, $s(t)$. This is accomplished by using a quartic polynomial that is based on the initial state, $S_0 = [s_0, \dot{s}_0, \ddot{s}_0]$, at time t_0 , and the final state, $S_1 = [\dot{s}_1, \ddot{s}_1]$, at time $t_1 = t_0 + T$. The polynomial $s(t)$ is then sampled at the final time $t_1 = t_0 + T$ to obtain the final position s_f .

For the final position s_f , the lateral distances d_i and d_o , related to the inner border and the outer border respectively, are calculated by projecting the Frenét coordinates $(s_f, 0)$ onto the global Cartesian coordinates (x_f, y_f) in relation to the race

line. The global coordinates are then projected back onto the Frenét coordinates in relation to the inner and outer borders to get d_i and d_o .

Finally, the track limits $[d_{min}, d_{max}]$ are computed as follows (Figure 3.2):

$$d_{min} = -d_o + m_o \quad (3.3)$$

$$d_{max} = -d_i - m_i \quad (3.4)$$

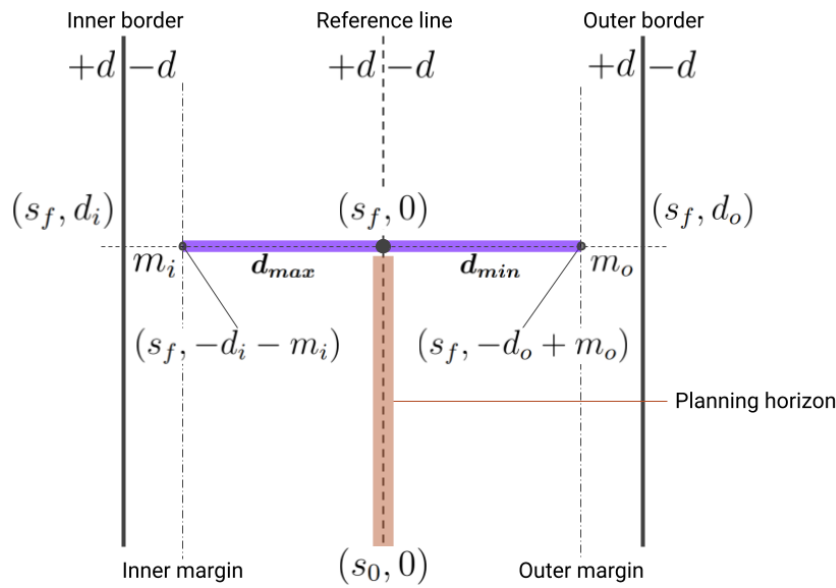


Figure 3.2: Representation of the points related to s_f in Frenét coordinates

3.2.2 Calculation of target speed

The planner must determine the desired speed to achieve and maintain at the end of the planned trajectories. To do this, several factors are taken into account. First, the target speed is set as $\dot{s}_{target} = \max(\dot{s}_{match}, \dot{s}_{min})$, where $\dot{s}_{match}$ is the speed required by the IAC race control and $\dot{s}_{min}$ is the minimum speed needed to keep the engine running. Second, the speed is limited by the zone speed as $\dot{s}_{target} = \min(\dot{s}_{target}, \dot{s}_{zone, \zeta})$ for $\zeta = 1, \dots, Z$, where Z is the number of sectors on the track and $\dot{s}_{zone, \zeta}$ is the maximum achievable speed for zone ζ . These $\dot{s}_{zone, \zeta}$ were established to provide more strategic margin for deceleration on curves and acceleration on straightaways, in order to avoid the need to maintain a steady speed

throughout the track and to offer greater flexibility during testing.

Additional factors that must be taken into account when computing the target speed include the current mission scenario and the ego vehicle's role in the race. If the ego vehicle is the Defender, it will simply maintain the target speed set by the race control. On the other hand, if the ego is the Attacker and overtaking is not yet permitted, the target speed must be configured to maintain a distance of at least 30 meters from the leading opponent. This is accomplished with a custom implementation of an *Adaptive Cruise Control*, which is described in Appendix ??.

The target speed may exceed the vehicle's safe acceleration limits. To address this, a speed ramp is employed to limit the target speed to the vehicle's maximum acceleration and deceleration capabilities. First, a starting ramp speed $\dot{s}_{ramp}$ is set to

$$\min(\dot{s}_{ramp}, \dot{s}_{ego}) \quad (3.5)$$

with $\dot{s}_{ramp}$ being set to 0 for the first iteration. The target acceleration, $\ddot{s}_{target}$, is then calculated based on the maximum and minimum accelerations and the desired change in speed over time, as shown in the following:

$$\ddot{s}_{target} = \begin{cases} \ddot{s}_{min}, & \text{if } \frac{\dot{s}_{target} - \dot{s}_{ramp}}{T} < \ddot{s}_{min} \\ \ddot{s}_{max}, & \text{if } \frac{\dot{s}_{target} - \dot{s}_{ramp}}{T} > \ddot{s}_{max} \\ \frac{\dot{s}_{target} - \dot{s}_{ramp}}{T}, & \text{otherwise} \end{cases} \quad (3.6)$$

The target speed $\dot{s}_{target}$ is computed using the target acceleration:

$$\dot{s}_{target} = \dot{s}_{target} + T\ddot{s}_{target} \quad (3.7)$$

Finally, the starting ramp speed is updated for the next iteration using Equations (3.6) and (3.7), but with $T = \Delta t$ to find the starting speed the vehicle should reach at the next time step.

If the ego is the Attacker and overtaking is allowed, the mission is set to a special internal state that allows the ego vehicle to close the gap with the leading opponent. Once the ego vehicle has closed in on the opponent sufficiently, the internal state changes to permit the overtake, and the vehicle accelerates to a target speed of $\dot{s}_{target} = \dot{s}_{match} + \Delta\dot{s}$. The rules do not impose any limit on the target speed for

overtaking, and $\Delta \dot{s}$ is typically set to 40km/h.

The computed target speed will be then used to generate the longitudinal movement.

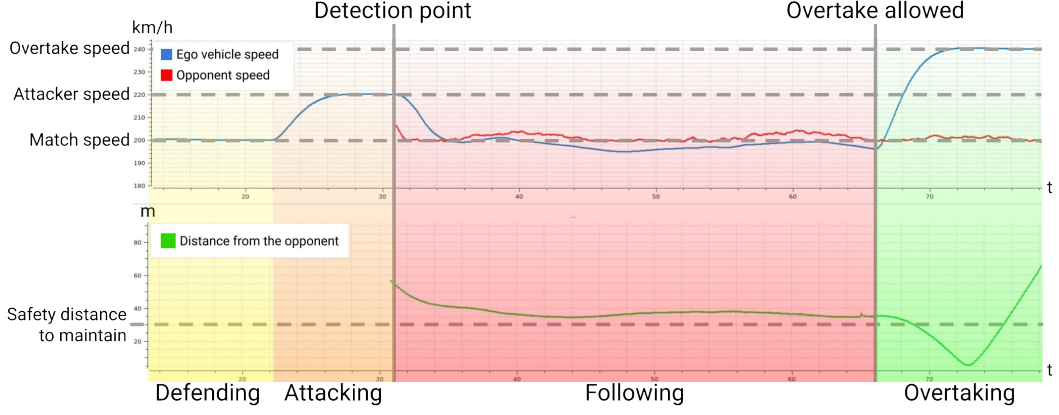


Figure 3.3: The calculated $\dot{s}_{target}$ for an overtake maneuver is illustrated with different phases highlighted. The upper graph displays the velocity of both the ego vehicle and the opponent, while the lower graph depicts the gap in distance between the two vehicles.

3.2.3 Multi-Step trajectory generation

By doing various simulation tests, it was realized that the time horizon of the generated trajectories was not sufficient to guarantee a smooth overtaking maneuver in every situation. In order to increase the time interval over which to generate the trajectory graph, it was necessary to make some changes. In fact, by simply increasing the parameter that defines the time interval, a very narrow trajectory graph is generated in the first few meters, increasing the risk of not finding a feasible trajectory. To solve this problem, the trajectory generation has been divided into two steps (Figure 3.4).

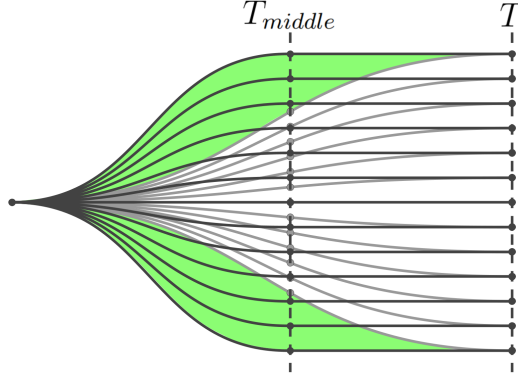


Figure 3.4: Representation of the two steps used to generate the trajectories with a greater horizon. In gray is shown the set of trajectories with the same final T but without the two step method. In green is highlighted the gained space with this feature activated.

The first step generates the trajectories normally, so that they span the interval $[d_{min}, d_{max}]$ until reaching an intermediate time T_{middle} . The second step, on the other hand, continues from the final state of the first step and for each trajectory generates its continuation, keeping $d(t)$ constant until reaching the desired final time T .

In cases where our and the opponent's trajectories are on a collision course, a steady trajectory must be created to avoid the crash and should not result in a zigzag maneuver from constantly changing sides during the overtake. To achieve a smooth overtake even in edge cases where the time to collision is close, the trajectory generation follows the multi-step process described. In particular, a set of trajectories is generated, and safety checks are performed on it. If the best trajectory is expected to collide with a time to collision $T_{min} < T_c + \Delta t \leq T_{max}$, the multi-step process is activated.

The first step generates trajectories in the time interval $[0, T_c + \Delta t]$, performs safety checks, and selects the best trajectory as τ_1 . The second step generates trajectories in the time interval $[T_c + \Delta t, T_{max}]$ using the end state of τ_1 as its starting state, and the best trajectory is selected as τ_2 . Lastly, τ_1 and τ_2 are combined to create the final trajectory, and the worst-case path properties from the two are applied. Figure 3.5 shows how two steps are used to create a final trajectory avoiding a failure in case the first set of generated trajectories collide.

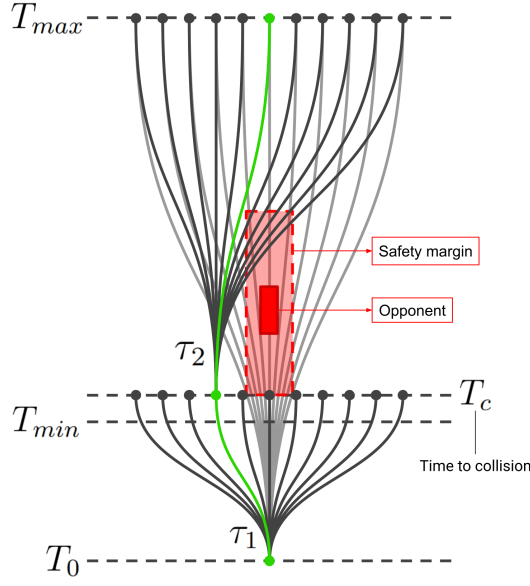


Figure 3.5: The first set of generated trajectories, represented in gray, collide all with the opponent safe bound. The Multi-Step is activated generating two sets of trajectories in two steps and the final trajectory is highlighted in green.

3.2.4 Defensive evasive maneuver

When the ego vehicle is in a defensive position and an opponent is trying to overtake, the situation could become hazardous. If the opponent is approaching and is too close, evasive action is necessary to prevent a collision. Additionally, the ego vehicle should not obstruct the opponent's overtaking maneuver unless there is a real safety concern. Thus, when the opponent is too close, the collision detection is postponed until the opponent is alongside the ego vehicle, and an action is taken only if the opponent is moving towards the ego vehicle.

To prevent unintended actions when the opponents are far away and there is no imminent collision threat, the cost of colliding with obstacles is temporarily ignored and then reconsidered when the opponents are close or when a collision is imminent.

These precautions are particularly useful in situations where the opponent is trying to overtake the ego vehicle with a low $\Delta\dot{s}$, remaining alongside for an extended period while traveling a considerable distance, which is the most dangerous scenario where things could go wrong.

3.2.5 Attacker "Right of Way"

If a collision between the opponent's and ego's trajectories is predicted, the planner should prevent the collision. However, suppose the ego vehicle is in an attacking position and ahead of the opponent. In that case, it's the opponent's responsibility to allow enough room for the ego to pass, a concept known as "right of way" in racing.

Disabling this feature would cause the ego vehicle to brake during an overtake if a collision is predicted, making overtaking even more challenging and slowing down the acceleration phase. Instead, with this feature enabled, the vehicle determines its position relative to the opponent. If it's in the lead, it maintains its path by adjusting the interval $[d_{min}, d_{max}]$ to the last computed lateral displacement $[d_{last}, d_{last}]$, creating a trajectory equal to the previous iteration's trajectory. This continues until no more collisions are expected, and the set of trajectories is generated as usual.

To check if the vehicle has passed the opponent the projection of the opponent's position from the Cartesian coordinates (x_{opp}, y_{opp}) to the Frenét coordinates (s_{opp}, d_{opp}) related to the ego race line is made. Using the opponent's longitudinal position s_{opp} the longitudinal difference Δs is computed as:

$$\Delta = \begin{cases} 0, & \text{if } \Delta s_{t-1} > 0 \\ -2.5, & \text{otherwise} \end{cases}$$

$$\Delta s = s_{ego} - s_{opp} - \Delta - \lceil (\frac{s_{ego} - s_{opp} - \Delta}{L} - \frac{1}{2})L \rceil \quad (3.8)$$

where L is the track length and Δ is an offset designed to obtain a hysteresis logic. If the computed Δs is positive means that the vehicle has passed the opponent.

3.2.6 Track re-entry trajectories

In the event that the vehicle goes beyond the track limits, it should attempt to return to the track instead of failing. This is especially important when soft boundaries are considered as actual track limits, such as areas with limited traction. Without this feature, the planner would activate a hard brake which could cause problems. With this feature, the vehicle generates a set of trajectories with the goal of getting

as close as possible to the race line within the boundaries, without braking (Figure 3.6).

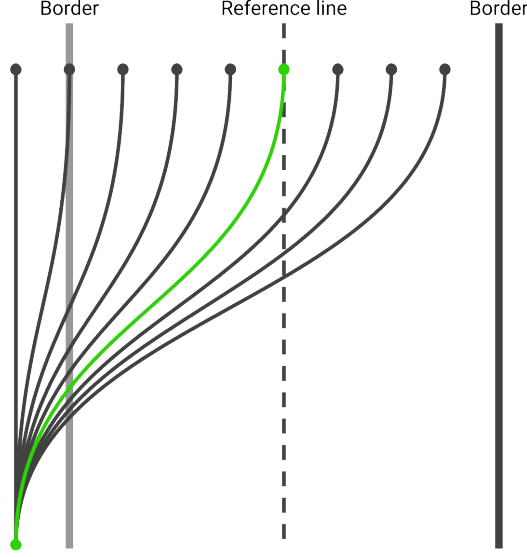


Figure 3.6: Set of trajectories generated to re-entry on track

To accomplish this, the border check is temporarily disabled to prevent a failure. Then, the set of trajectories is restricted to the interval $[d_{min}, d_{max}]$, computed as follows:

$$d_{min} = \min(d(t) - m_o, d_{right}) \quad (3.9)$$

$$d_{max} = \max(d(t) + m_i, d_{left}) \quad (3.10)$$

where d_{right} and d_{left} are defined in the Equations (3.3) and (3.4), respectively. This interval takes into account the vehicle's current position as being on one of the extremes when it's out of bounds.

3.2.7 Emergency path

In the event that the generated paths all go beyond the track limits, such as in a tight space, the planner must still create a backup plan to prevent failure. This emergency path will keep the current lateral position $d(t)$ in place until a feasible path that doesn't collide with boundaries can be found. If the vehicle is not making an attack and doesn't have the right of way, then a braking profile is applied to gradually slow it down. Whenever the current path is an emergency path, the range of lateral

positions $[d_{min}, d_{max}]$ is reset to its default values to allow the correct calculation of track limits for the next iterations.

3.3 Results

The proposed extended Frenet-frame-based planner has been tested in experimental tests during the preparation of the IAC events at LOR, the static avoidance test at IMS and the head-to-head competition at LVMS.

3.3.1 Static Obstacle Avoidance

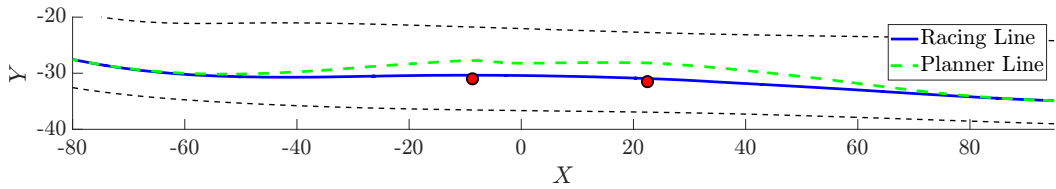


Figure 3.7: Static obstacles test at LOR. See https://youtu.be/LIzb-_8vrI8 for a video of the test.

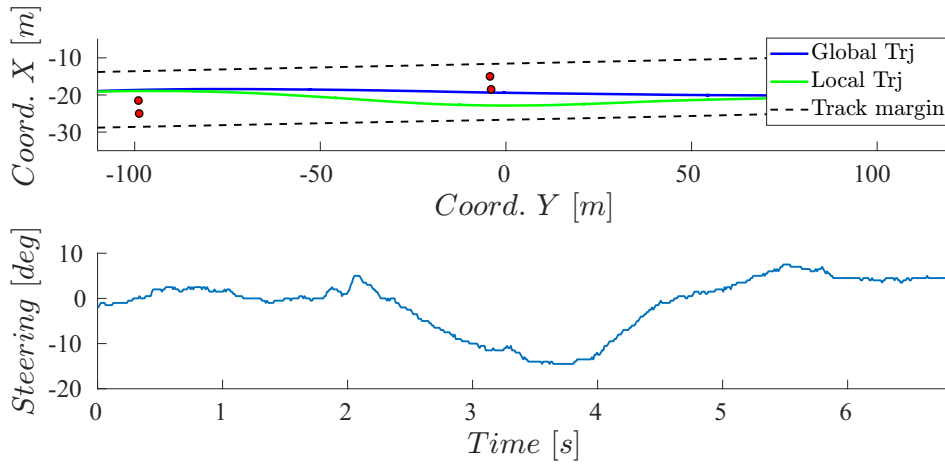


Figure 3.8: Avoiding pylons during the Semifinal at IMS. The location of four water-filled pylons is marked with red dots. The local planner is informed by the detection stack and reacts with a smooth trajectory to avoid them.

Figure 3.7 shows a scenario in which static obstacles have been added to the LOR track. The AV-21 racecar was able to safely avoid the obstacles at a velocity of 34

m/s. The safe sensor range for the LiDAR-based detection was set at 60m. Thus, the planner received the obstacle position 1.7s before the potential collision.

A further demonstration of the ability to avoid static obstacles is shown in Figure 3.8 depicting the maneuver followed at the Semifinal event of the first race of the IAC at IMS.

3.3.2 Head-to-Head Racing

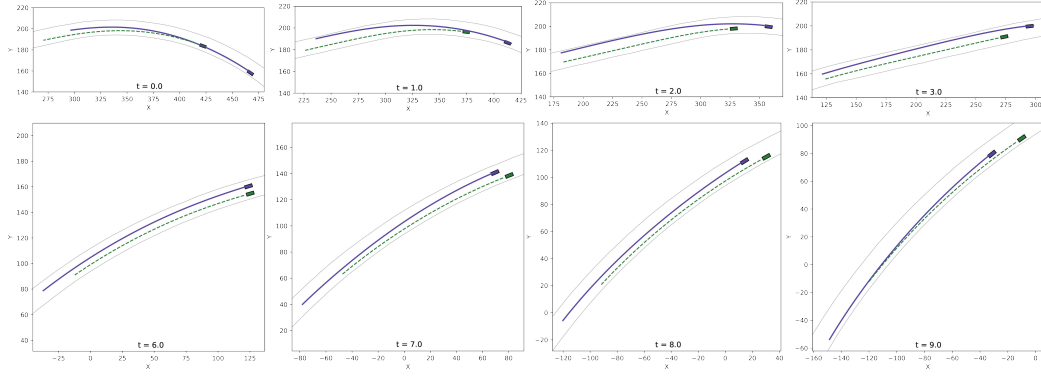


Figure 3.9: Frame sequence during the overtake at 63 m/s. The solid blue line is the planner trajectory. The dashed green line is the motion prediction of the opponent.

Figure 3.9 shows a frame sequence of the overtake at the highest speed achieved. It should be mentioned that the experimental data in the head-to-head scenario ends at a top speed of 63 m/s due to a wrong hard brake command triggered by a module separated from the motion planner and controller, causing the TII-ER vehicle to collide with the track borders.

A comparison between the original version of the planner and the improved one, with all the new features enabled, has been performed replicating the passing competition format on the Unity-based simulator briefly described in Chapter 2. All tests have been performed with different match speeds and overtake speeds to check the behavior in both the usual and unusual cases. As metrics have been used the space traveled during the overtake, the percentage of failed iterations which results in braking, and average execution time. Moreover, the average of the collision coefficient value is used, which indicates if the ego vehicle violated the safe bound around the opponent, and its value is defined in the interval $[0, 1]$, where 0 means

no violation, 1 means violation of the hardbound, and the values in between mean a violation of the softbound. In particular, we computed the average of all collision coefficients during the overtaking maneuver, giving an indication of how safe was the overtake. Tables 3.1 and 3.2 compare the two versions using the described metrics. The same data are summarized using charts for each metric in Figure 3.10.

Table 3.1: Original planner

Match speed	Overtake speed	Space traveled	Avg collision coefficient	Braking iterations	Avg exe. time	Avg gen. paths
180 km/h	220 km/h	1517.78 m	0.214	0.158 %	9.120 ms	41
180 km/h	200 km/h	1065.03 m	0.007	0.000 %	9.202 ms	41
180 km/h	190 km/h	2920.34 m	0.116	0.072 %	9.235 ms	41
220 km/h	260 km/h	1829.26 m	0.249	0.148 %	9.221 ms	41
220 km/h	240 km/h	2967.13 m	0.155	0.150 %	9.291 ms	41
220 km/h	230 km/h	2416.39 m	0.038	0.000 %	9.315 ms	41

Table 3.2: Improved planner

Match speed	Overtake speed	Space travelled	Avg collision coefficient	Braking iterations	Avg exe. time	Avg gen. paths
180 km/h	220 km/h	947.79 m	0.191	0.092 %	6.394 ms	25.161
180 km/h	200 km/h	1060.15 m	0.006	0.000 %	5.902 ms	22.639
180 km/h	190 km/h	2232.87 m	0.043	0.022 %	6.234 ms	23.133
220 km/h	260 km/h	797.13 m	0.066	0.006 %	5.999 ms	23.050
220 km/h	240 km/h	1736.58 m	0.074	0.048 %	6.206 ms	23.730
220 km/h	230 km/h	2477.14 m	0.011	0.004 %	6.016 ms	22.171

The average improvements in percentage are:

- Space traveled: **-27.24 %**
- Collision coefficient: **-49.92 %**
- Braking percentage: **-67.42 %**
- Execution time: **-33.64 %**
- Number of generated trajectories: **-43.14 %**

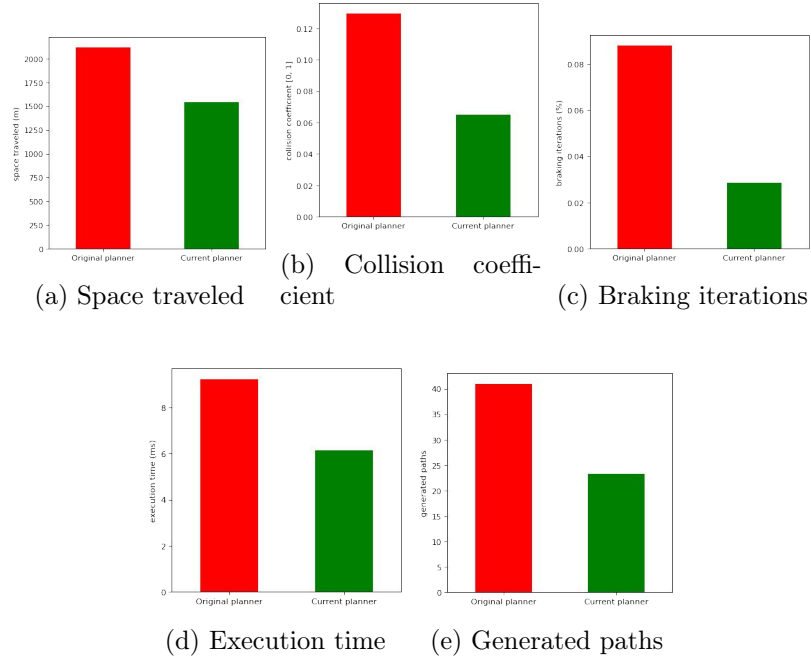


Figure 3.10: Original planner and improved planner average data comparison for each metric

3.4 Conclusions

In conclusion, the Frenet frame-based planner presented in this work demonstrated effectiveness in generating trajectories for autonomous racecars in oval racing in different scenarios. The planner successfully navigated static obstacles and facilitated overtaking maneuvers. Key features, such as track limit calculations, target speed management, and the ability to handle emergency situations, contributed to the performance of the planner and to the extension of the original work.

Chapter 4

Model Predictive Control on a Fullscale Racecar

The controller proposed in this chapter is highly based on the work presented in [86]. Its effectiveness is rooted in the utilization of a precise yet straightforward vehicle model, coupled with the efficiency afforded by CppADCodeGen¹ for automatic differentiation and HPIPM[30] for optimization solving. Furthermore, this approach has demonstrated its efficacy with different versions in both scaled racecars[51] and electric FSD[86][44][43].

4.1 Vehicle Models

In this chapter, the vehicle models used in the MPC are presented. It should be mentioned that the model equations are not linear; therefore, the model is discretized using the fourth order Runge-Kutta (RK4), and a linear approximation is applied to the constraints. This allows us to set the problem as a quadratic program (QP), more precisely in two steps of a sequential quadratic program (SQP), and solve it with an interface to HPIPM.

4.1.1 Curvilinear Single Track Model

A single track dynamic model, shown in Figure 4.1, is used for the offline trajectory optimization problem, which is not in the scope of this thesis, and the MPC. As in

¹<https://github.com/joaoleal/CppADCodeGen>

[86], we use curvilinear/Frenét coordinates to describe the state. Therefore, global position and heading are not directly considered, but transformed to a state relative to the reference path.

Equations of Motion

The vehicle state is given by $x = [s; n; \mu; v_x; v_y; r; \delta; T; B]$ and the input as $u = [\Delta\delta; \Delta T; \Delta B]$, where s , n and μ are the progress along the path, the orthogonal deviation from the path and the local heading. Longitudinal v_x and lateral v_y velocities are considered as well as the yaw rate r . δ , T and B are the steering angle, throttle command and brake command, which are included in the state. The control commands $\Delta\delta$, ΔT and ΔB are the derivatives of the inputs. Thus, the equations of motion are

$$\begin{aligned}\dot{s} &= \frac{v_x \cos(\mu) - v_y \sin(\mu)}{1 - n \kappa(s)}, \\ \dot{n} &= v_x \sin(\mu) + v_y \cos(\mu), \\ \dot{\mu} &= r - \kappa(s)\dot{s}, \\ \dot{v}_x &= \frac{1}{m} (F_{x_r} - F_d - F_{y_f} \sin(\delta) + F_{x_f} \cos(\delta) - F_{b_x} + m v_y r), \\ \dot{v}_y &= \frac{1}{m} (F_{y_r} + F_{y_f} \cos(\delta) + F_{x_f} \sin(\delta) - F_{b_y} - m v_x r), \\ \dot{r} &= \frac{1}{I_z} (l_f (F_{y_f} \cos(\delta) + F_{x_f} \sin(\delta)) - l_r F_{y_r}), \\ \dot{\delta} &= \Delta\delta, \\ \dot{T} &= \Delta T, \\ \dot{B} &= \Delta B,\end{aligned}$$

where $\kappa(s)$ is the curvature at the progress s , l_f and l_r are the distances from the center of gravity to the front and rear wheels, m is the mass and I_z the moment of inertia. F_{y_f} , F_{y_r} are the lateral tire forces at the front and rear wheels. F_{x_f} , F_{x_r} are the longitudinal forces at front and rear axles. F_{b_x} and F_{b_y} model the forces on the x and y-axis due to the road bank angle θ , and are given by $F_{b_x} = mg \sin(\theta) \sin(\mu)$ and $F_{b_y} = mg \sin(\theta) \cos(\mu)$. F_d represents the aerodynamic effects considering the air density ρ , the frontal area S and the drag coefficient C_d ,

$$F_d = 0.5 \rho S C_d v_x^2.$$

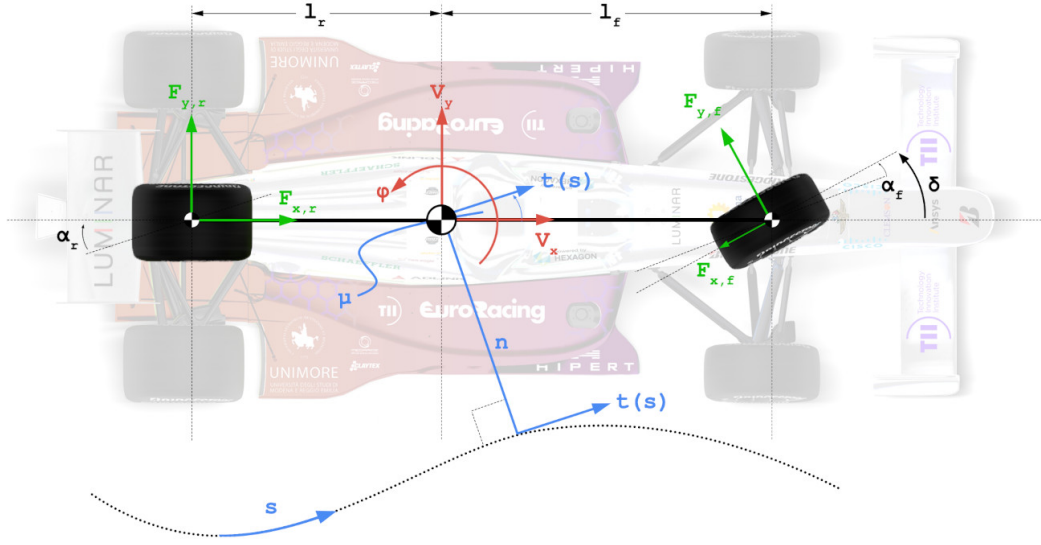


Figure 4.1: Dynamic single track vehicle model on curvilinear coordinates.

Tire Model

The tires effects are modeled using a simplified Pacejka Magic Formula [64] with a combined slip correction. Beyond the usual macro-parameters B , C , D and E , the lateral force offsets Sh_{y_i} , $i \in [f, r]$, have been included resulting in

$$\begin{aligned} F_{y_f, \text{lat}} &= Sh_{y_f} + D_f \sin \left(C_f \tan^{-1}(B_f \alpha_{y_f}) + \right. \\ &\quad \left. - E_f (B_f \alpha_{y_f} - \tan^{-1}(B_f \alpha_{y_f})) \right), \\ F_{y_r, \text{lat}} &= Sh_{y_r} + D_r \sin \left(C_r \tan^{-1}(B_r \alpha_{y_r}) + \right. \\ &\quad \left. - E_r (B_r \alpha_{y_r} - \tan^{-1}(B_r \alpha_{y_r})) \right), \end{aligned} \quad (4.1)$$

where $\alpha_{y_i} = \alpha_i + Sh_{y_i}$ is the resulting slip angle obtained by applying a shift Sh_{y_i} to the front slip angle α_f and the rear slip angle α_r

$$\begin{aligned} \alpha_f &= \tan^{-1} \left(\frac{v_y + l_f r}{v_x} \right) - \delta, \\ \alpha_r &= \tan^{-1} \left(\frac{v_y - l_r r}{v_x} \right). \end{aligned}$$

Sh_{y_i} and Sh_{v_i} are calculated using Pacejka micro-parameters [64] related to horizontal shifts and variation of the lateral force shift considering the change in the tire load with respect to the reference vertical load and the camber angle.

To consider the combined slip, we propose a combined slip weighting factor. Thus, the pure lateral forces $F_{y_f,lat}$ and $F_{y_r,lat}$ are weighted with G_{y_i} , such that we get the final forces, $F_{y_f} = G_{y_f} F_{y_f,lat}$ and $F_{y_r} = G_{y_r} F_{y_r,lat}$, with G_{y_i} given by

$$G_{y_i} = \cos \left(\arcsin(F_{x_i}/F_{\max_i}) \right),$$

where $F_{\max_i} = D_i \epsilon_i$ and ϵ_i is an ellipse shape parameter. Note that we clip the force fraction $F_{x_i}/F_{\max_i}$ at 0.98 to avoid singularity issues.

Longitudinal Forces

The front axle longitudinal forces are modeled as

$$F_{x_f} = -C_{b_f} B - C_{ro},$$

where C_{ro} is the rolling resistance. The braking force is represented as $C_{b_f} B$, with C_{b_f} the maximum brake pedal pressure and $B \in [0, 1]$.

Considering a rear wheels drive powertrain, the rear longitudinal force is modeled as

$$F_{x_r} = C_m T - C_{br} B - C_{ro}, \quad (4.2)$$

where C_m is a linear engine coefficient and $T \in [0, 1]$. The turbo-charged combustion engine used on the research vehicle produces a force which is not linear in the whole usable regions since it depends on the engine rpm and gear. In order to represent this behavior on (4.2), a scale factor k_s varying on the speed has been applied to the upper bound constraint of the throttle command, thus $T \in [0, k_s]$. To improve the accuracy of the engine modeling, during the experiments, another formulation formed by two polynomials has also been used which has been fitted on the engine map and experimental data.

Gear shifting effects are neglected, as well as the gear command. The latter is controlled separately and sent to the low-level controller when the desired engine RPM is reached.

4.1.2 Locked Differential Tricycle Model

To enhance the model and capture the effects of the locked differential, which are particularly important in a dynamic racetrack as in the F1 Monza layout, a novel model called tricycle model is presented. For what concerns the front axle, the tricycle formulation is derived from the classic single-track model presented in Section 4.1.1, while the rear axle is derived from the double-track model. The final structure is shown in Figure 4.2. The new state-space model is described by the state vector $\tilde{x} = [s; n; \mu; v_x; v_y; r; \delta; D]$ and the input vector $\tilde{u} = [\Delta\delta; \Delta D]$, where s , n , and μ represent the progress along the path, the orthogonal deviation from the path, and the local heading. The motion equations are expressed as the derivative of the longitudinal v_x and the lateral v_y velocities and the yaw rate r . δ represents the steering angle and D is the desired longitudinal acceleration. $\Delta\delta$ and ΔD represent the derivatives of the inputs.

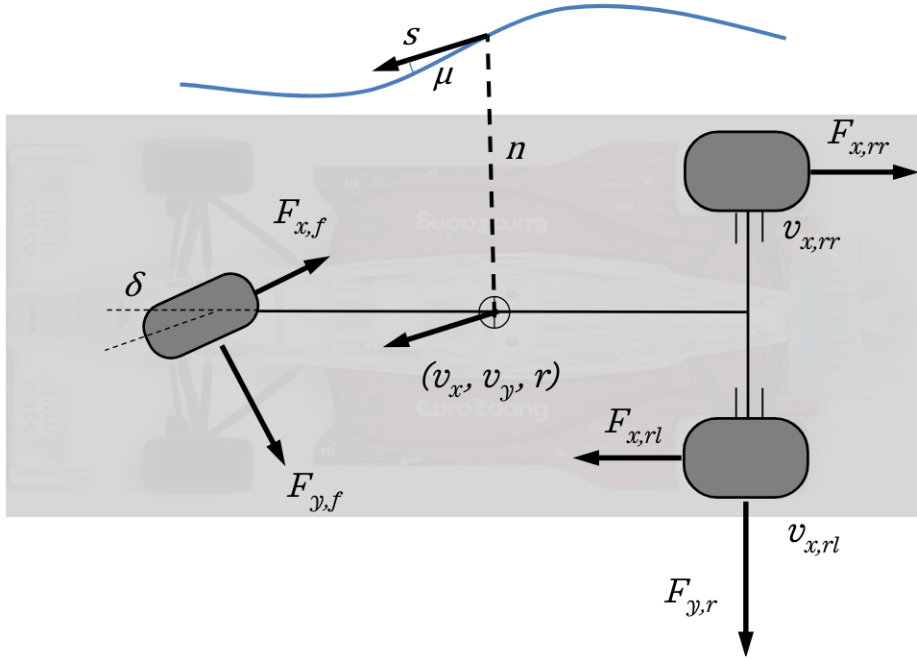


Figure 4.2: Scheme of the three-wheels model, in curvilinear coordinates, included in the MPC.

In the motion equations, we included the longitudinal forces required for traction and braking F_{xf}^{cmd} and F_{xr}^{cmd} , and the contribution of the longitudinal forces to the

total yaw moment, M_{diff} . The differences with respect to the single-track equations of motion are presented here:

$$\dot{v}_x = \frac{1}{m} (F_{xr}^{cmd} - F_d - F_{yf} \sin(\delta) + F_{xf}^{cmd} \cos(\delta) + mv_y r), \quad (4.3a)$$

$$\dot{v}_y = \frac{1}{m} (F_{yr} + F_{yf} \cos(\delta) + F_{xf}^{cmd} \sin(\delta) - mv_x r), \quad (4.3b)$$

$$\dot{r} = \frac{1}{I_z} (M_{diff} + l_f (F_{yf} \cos(\delta) + F_{xf}^{cmd} \sin(\delta)) - l_r F_{yr}), \quad (4.3c)$$

where m and I_z are the mass and the inertia of the vehicle, and l_f and l_r are the front and rear wheelbase. F_{yf} and F_{yr} are the lateral forces generated in the interface between the road and the tires, and F_d is the aerodynamic drag. The rolling resistance is included in the longitudinal equilibrium as well. From now on, the index i^{th} is used to identify the axle (front, rear), while the index j^{th} identifies the vehicle side (left, right).

The yaw moment is derived from the rotational equilibrium of the rear axle in the Z-axis direction:

$$M_{diff} = \frac{1}{2} (F_{x,rr} - F_{x,rl}) \cdot t_r. \quad (4.4)$$

where t_r represents the rear axle track. The longitudinal forces include both the contribution of the locked differential and the traction dynamics: $F_{x,rj} = F'_{x,rj} + F_{x,rj}^{cmd}$. The longitudinal forces experienced during vehicle coasting are estimated using the non-linear Pacejka tire model:

$$F'_{x,rj} = D_{x,rj} \sin \left(C_{x,r} \tan^{-1} (B_{x,r} k_{x,rj}) \right) \quad (4.5)$$

$D_{x,rj} = \mu_r F_{z,rj}$, B_r and C_r are the macro-parameters of the magic formula [64] and $F_{z,rj}$ represents the vertical load on the j^{th} wheel, as will be shown in 4.8. The slip ratio is expressed as

$$k_{x,rj} = \frac{v_x - v_{x,rj}}{v_x} \quad (4.6)$$

and depends on the ideal speed of the rear wheels, taking into account the vehicle's

geometry and the track's layout:

$$v_{x,rj} = r \cdot \left(R \pm \frac{t_r}{2} \right). \quad (4.7)$$

In the equation 4.7, R is the turning radius. To account for the traction forces, we have included a straightforward formulation based on the commanded acceleration. We defined $F_{x,r}^{cmd} = f(D)$, which is distributed between the rear wheels through the lateral load transfer. In this way, we can check for a possible nose-in during the phase of turn-exit.

Considering the tricycle structure, it proved necessary to include a model for estimating the lateral load transfer and, consequently, to distribute the vertical load on each wheel. Hence, the standard model based on the roll axis [36] has been included. This has been employed exclusively for the rear axle, as the axle effective characteristic is used for the front one. The vertical load on each wheel is computed:

$$F_{z,rj} = \frac{1}{2} F_{z,r} \pm \Delta F_{z,r}^{lat} \quad (4.8)$$

where the global load on the rear axle $F_{z,r}$ is the sum of the static load $F_{z,r}^0$, the aerodynamic effect $F_{z,r}^{aero}$ [68] and the longitudinal load transfer $\Delta F_{z,r}^{long}$ contributions [36]. The lateral load transfer is estimated with the relation:

$$\Delta F_{z,r}^{lat} = F_{y,r} \frac{h_r}{t_r} + \frac{M_y}{t_r} \frac{K_r}{K_{tot}}, \quad (4.9)$$

where h_r is the height of the roll axis at the rear axle, K_r and K_{tot} are the rear axle roll stiffness and the total roll stiffness. $M_y = m \cdot a_y \cdot q$ and $F_{y,r}$ represent the moment and force generated by the inertia on the roll axis. To estimate the latter, a rotational equilibrium of the whole vehicle on the Z-axis is performed, from which it follows that:

$$F_{y,r} = (m \cdot a_y \cdot l_f + M_{diff}^0) \cdot \frac{1}{l} \quad (4.10)$$

Here, l and q represent, respectively, the wheelbase and the Z-distance between the center of gravity and the roll axis. M_{diff}^0 represents the yawing moment calculated by the model in the previous integration step and a_y is calculated starting from the motion field.

The employment of this simplified roll-axis model for the load transfer estimation is an acceptable approximation since the roll motion of an open-wheel vehicle is negligible. This module allows the model to improve the accuracy in the calculation of the longitudinal forces, and consequently the prediction of the vehicle's pose.

The longitudinal forces are also used to account for the combined slip on the tires. The formulation is based on a friction ellipse, through which a weight factor $G_{y,ri}$ is computed as in [68]. Likewise, the weight factor is defined for the front axle.

Due to the characteristics of the system at low speeds, a kinematic model has been used for low-speed maneuvers. To keep this speed threshold low, a stability analysis of the dynamic model has been performed, finding the suitable discretization step time needed to keep the model stable.

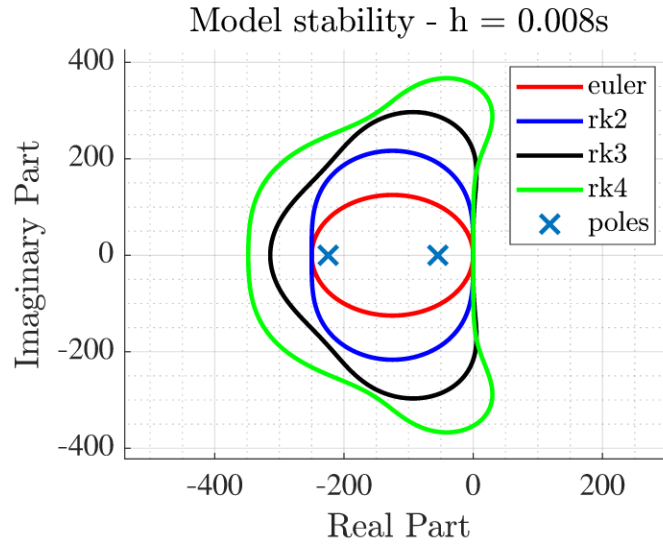


Figure 4.3: Poles for the three-wheels model at $v_x = 8m/s$. The proposed step size for the integration allows the system poles to lay within the stability region of the explicit Euler method.

For the single-track model, we used the numerical method Runge-Kutta 4 (RK4) and a time step $h = 0.04s$ [68]. This setup guarantees stability only for a longitudinal speed greater than $28 m/s$ on the single-track, and $32 m/s$ on the three-wheels model. This is not suitable for a road course track like Monza, where the speed can reach a minimum of $12 m/s$ (i.e., Variante 1). To guarantee the numerical stability of the dynamic model at low speeds, a small integration step size has been chosen

that includes the poles of the linearized system within the stability region defined by the numerical method [29].

The overall results of the stability analysis for the tricycle model are shown in Figure 4.3. An optimal value of the time step $h = 0.008s$ was chosen to cover all significant speed ranges expected on the track, i.e. $v_x^{min} = 8m/s$. Below this threshold, a kinematic model written in the dynamic model states is used to avoid numeric oscillations in the solution. The two models are blended similarly to what has been presented in [44].

4.2 MPC Formulation

4.2.1 Single Track MPC formulation

In the control problem, the model is discretized in time $f_t^d(x_t, u_t)$ using a fourth-order Runge Kutta method. The MPC cost function combines the progress optimization with the regularization terms in order to penalize the rate of change of the physical inputs and rear slip angle:

$$J_{MPC}(x_t, u_t) = -\dot{s}_t + q_n n_t^2 + q_\mu \mu_t^2 + q_v |s_{v,t}| + u^T R u + B(x_t).$$

The cost function includes path following weights q_n and q_μ , as well as a velocity tracking weight q_v on the slack variable $s_{v,t}$ of the upper velocity constraint. Note that the reference path is given by the Frenét-planner.

The MPC problem is formulated as

$$\begin{aligned} \min_{X, U} \quad & \sum_{t=0}^T J_{MPC}(x_t, u_t) \\ \text{s.t.} \quad & x_0 = \hat{x}, \\ & x_{t+1} = f_t^d(x_t, u_t), \\ & x_t \in X_{track} \quad x_t \in X_{ellipse}, \\ & v_{x,t} \leq \bar{v} - s_{v,t} \quad s_{v,t} \geq 0, \\ & a_t \in \mathbf{A}, u_t \in \mathbf{U}, t = 0, \dots, T. \end{aligned}$$

where $\hat{x}$ is the current curvilinear state, $\bar{v}$ is the upper velocity bound and T is the

prediction horizon.

The optimization problem is solved using a custom sequential quadratic programming framework, which uses HPIPM [30], a high-performance quadratic programming framework for MPC, and CppADCodeGen, a code generation automatic differentiation library.

4.2.2 Tricycle Model - MPC formulation

The main differences in the problem formulation for the tricycle model are:

- A velocity tracking weight q_v instead of a slack variable to follow more strictly the velocity reference.
- A weight on the yaw rate q_r which aims to mitigate potential oscillations caused by bumpy portions of the track or disturbances in the steering actuation.
- Additional independent variables, and an improvement of the model discretization.
- The inputs are the derivative of the steering angle and the requested longitudinal acceleration instead of using the throttle and brake commands.

The cost function is formulated as:

$$J_{MPC}(x_t, u_t) = q_n n_t^2 + q_\mu \mu_t^2 + q_v v_t^2 + q_r + u^T R u + B(x_t) \quad (4.11)$$

where q_n and q_μ are path following weights. The regularization term $B(x_k) = q_B \alpha_r^2$ penalizes the rear slip angle, while $u^T R u$ is a regularizer on the input rates where R is a diagonal weight matrix.

The MPC problem is formulated as

$$\min_{X, U} \sum_{t=0}^T J_{MPC}(x_t, u_t) \quad (4.12a)$$

$$s.t. \quad x_0 = \hat{x}, \quad (4.12b)$$

$$x_{t+1} = f_t^d(x_t, u_t), \quad (4.12c)$$

$$x_t \in X_{track} \quad x_t \in X_{ellipse}, \quad (4.12d)$$

$$a_t \in \mathbf{A}, u_t \in \mathbf{U}, t = 0, \dots, T. \quad (4.12e)$$

where $X = [x_0, \dots, x_T]$, and $U = [u_0, \dots, u_T]$. $X_{ellipse}$ represents a friction ellipse constraints, and X_{track} constrains the lateral deviation n ensuring that the vehicle stays on the track. $\hat{x}$ is the current curvilinear state and T is the prediction horizon. $\mathbf{A}$ and $\mathbf{U}$ are box constraints for the physical inputs $a = [\delta; D]$ and their rate of change u .

4.2.3 Locked Differential Model Progression

In the definition of the optimization problem, the critical issue is advancing the prediction without the ability to track the evolution of M_{diff} with respect to the previous iteration. which is essential for estimating the correct value of the load transfer and the yaw moment itself. One solution could be the introduction of an additional variable to the state vector but this would increase the dimension of the optimization problem raising the computational time. Therefore, the M_{diff}^0 , which is the initial yaw moment for each step time t in the control horizon, is calculated before the execution of the control optimization, using the states from the latest MPC prediction horizon, and included as an independent variable. This is an acceptable simplification considering that having a stable model, a well-posed optimization problem, and a high-frequency running control scheme, the prediction of the MPC should not produce any abrupt change compared to the previous one.

4.2.4 Micro-steps Model Discretization

The MPC is executed in real-time at a frequency of 100Hz and a time horizon of 2.6s with a sampling time of $\Delta t = 40\text{ms}$. In the single-track model version, the model was discretized in time $f_t^d(x_t, u_t)$ with the same sampling time of the optimization:

$$x_{k+1} = \Phi_k(x_k, u_k, \Delta t), \quad (4.13)$$

where Φ_k is the chosen integrator (Euler or RK4). However, this does not adhere to the outcome of the stability analysis presented in Figure 4.3. Therefore, a micro-step discretization approach has been applied

$$\Phi_k(x_k, u_k, \Delta t) = \underbrace{\Phi_k(\dots \Phi_k(x_k, u_k, h), u_k, h)}_{\Delta t/h \text{ times}} \quad (4.14)$$

where h indicates a chosen sampling time divisor of Δt , in our case $h = 8\text{ms}$. This can also be seen as performing Φ_k with $\Delta t/h$ step sizes of h and picking the resulting x_{k+i} at each Δt .

4.3 Separating Low-level and High-level Control

With the introduction of the tricycle model, we separated the high-level and low-level control to reduce the number of states in the MPC and to easily manage the longitudinal dynamics modeling.

The MPC produces longitudinal velocity v_x^{mpc} and acceleration a_x^{mpc} , which are then converted into throttle and brake signals using a combination of two PI controllers and feed-forward controllers.

To calculate the throttle feed-forward action, the current engine speed, rpm , and the torque target, T_{ref} , are used to query a heuristically constructed look-up table. The torque is determined by the target longitudinal force F_x^{ref} , the radius of the rear wheel r_w , the gear ratio τ_i (with i being the current gear), and the final drive τ_d . The transmission efficiency η_t is included as well:

$$T_{ref} = \frac{F_x^{ref} \cdot r_w \cdot \tau_i \cdot \tau_d}{\eta_t} \quad (4.15)$$

The force F_x^{ref} indirectly incorporates the target velocity v_x^{mpc} through its dependence on F_d and F_{roll} . It also directly considers the acceleration target a_x^{mpc} according to the relationship:

$$F_x^{ref} = m \cdot a_x^{mpc} - F_d - F_{roll} + \frac{J_0 \cdot a_x^{mpc}}{r_w^2}, \quad (4.16)$$

where J_0 represents the rotational inertia of the mechanics and F_{roll} is defined as in [68]. To ensure that the target does not exceed the tire's longitudinal limit, the actual target used for throttle calculation is the minimum value between F_x^{ref} and $F_{x,r}^{limit}$. The latter is determined using Pacejka's peak factor $D_{x,r}$, already mentioned in Section 4.1.2. However, when the force in 4.16 is negative, it is converted into a brake signal using the equation 4.17. Since braking dynamics are significantly simpler than powertrain dynamics, a linear equation can be employed with negligible

error:

$$B = \frac{F_x^{tar}}{C_{b,f} + C_{b,r}} \cdot B^{max} \quad (4.17)$$

$C_{b,f}$ and $C_{b,r}$ are the brake coefficients determined by the geometry and the characteristic of the braking system, e.g. the disk dimension, and parameterized on the maximum pressure that can be applied in the system, B^{max} . Likewise in the throttle calculation, the actual longitudinal force used in 4.17 is the minimum between F_x^{ref} and a new threshold $F_{x,b}^{limit}$. Unlike the traction force, braking dynamics involve both the front and rear axles, thus requiring the assessment of the maximum force on both axles. Indeed, the braking limit is proportional to the sum of both $D_{x,f}$ and $D_{x,r}$.

Any errors in the feed-forward actions due to discrepancies in force estimation and other modeling errors were compensated for by the PI controllers, whose actions were added to the feed-forward ones.

4.4 Results

4.4.1 MPC Performance under Model Mismatch

In the following, the results of the MPC under model mismatch are presented. Indeed, in the first two events of the IAC, due to the limited testing time, it has been decided to keep using a first set of parameters extracted from the Dymola multibody simulator. Despite the inaccuracy of the tire parameters in the model, the controller has been able to achieve high speeds and maintain the vehicle stable.

The capability of the designed MPC to control the vehicle at high speeds has been tested during the time trial part of the IAC events. At the IMS track, TII-ER achieved the fastest time reaching an average speed of 62.5 m/s over a lap. The tracking performance at the LVMS is shown in Figure 4.4, where a top speed of 75.5 m/s has been reached. The maximum lateral error is 1m, whereas the RMS value is 0.5m. The heading error is maintained between 0.7deg and -1.0deg. A limited heading error despite a not negligible lateral error is the expected effect of the strategy applied to the MPC regularization terms. In particular, due to the uncertainty of the actuation performance and the model mismatch at high speeds, it has been decided to set the regularization term related to the steering wheel rate of change one order of magnitude higher than the value used in simulation

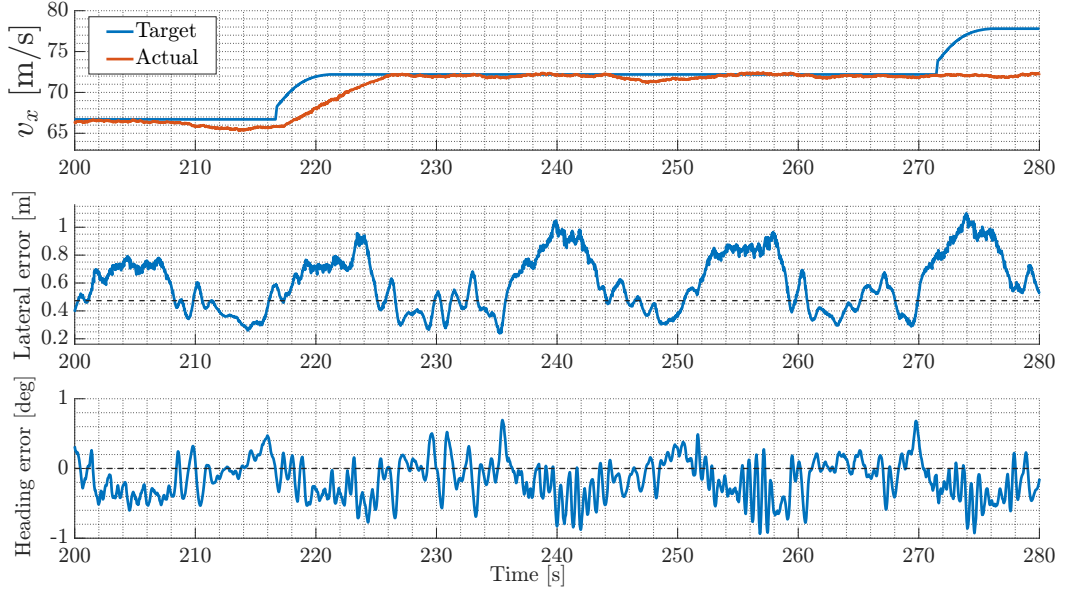


Figure 4.4: Experimental results for path tracking during high speed laps. See <https://youtu.be/ERTffn3IpIs?t=2013> for the time trial laps at LVMS.

and in other autonomous racing platforms in which the controller has been tested. The positive lateral error could be related to an inaccurate force offset Sh_{v_i} used in (4.1.1). Figure 4.5 presents the g-g diagram, showing that the racecar reached lateral accelerations up to 25 m/s^2 .

In both events, the top speed has been limited by hardware and engine malfunctions. In particular, Figure 4.6 shows the case in which the research vehicle was not able to reach the target speed despite a fully saturated throttle command due to a detached cable in the powertrain wiring.

During the high-speed laps at LVMS, when the speed varied from 72 m/s to 75.5 m/s , the MPC reached a maximum lateral error of 1m and an RMS value of 0.5m , while the heading error was between 0.7deg and -1.0deg . On the other side, when the speed varied from 61.45 m/s to 63.16 m/s during the final at IMS, the maximum lateral error was -0.67m and the RMS value -0.29m , while the heading error was between 0.6deg and -0.5deg with a mean of -0.016 .

In Figure 4.7, the tracking performance during the two events at similar speeds, over one lap, is shown. The error on the lateral tracking has been influenced by the choices on the regularization terms, and the mismatch in the tire model. The

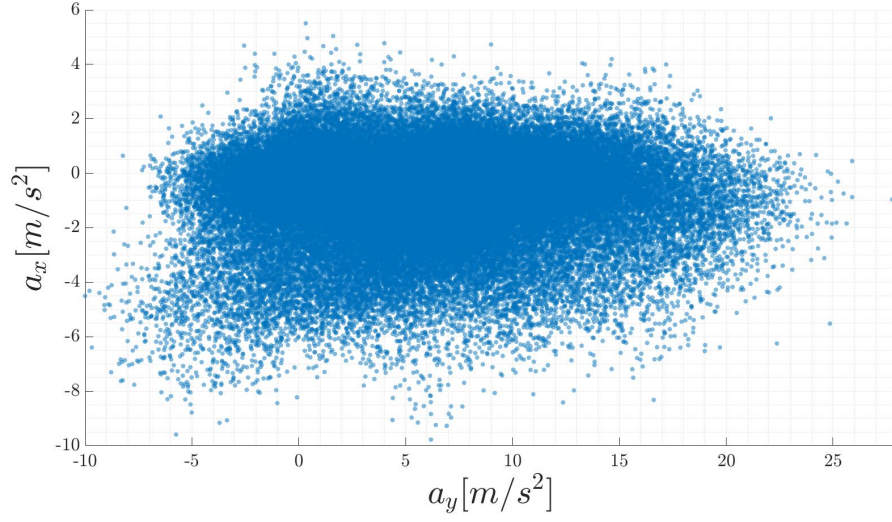


Figure 4.5: The g-g diagram of the fastest lap at LVMS.

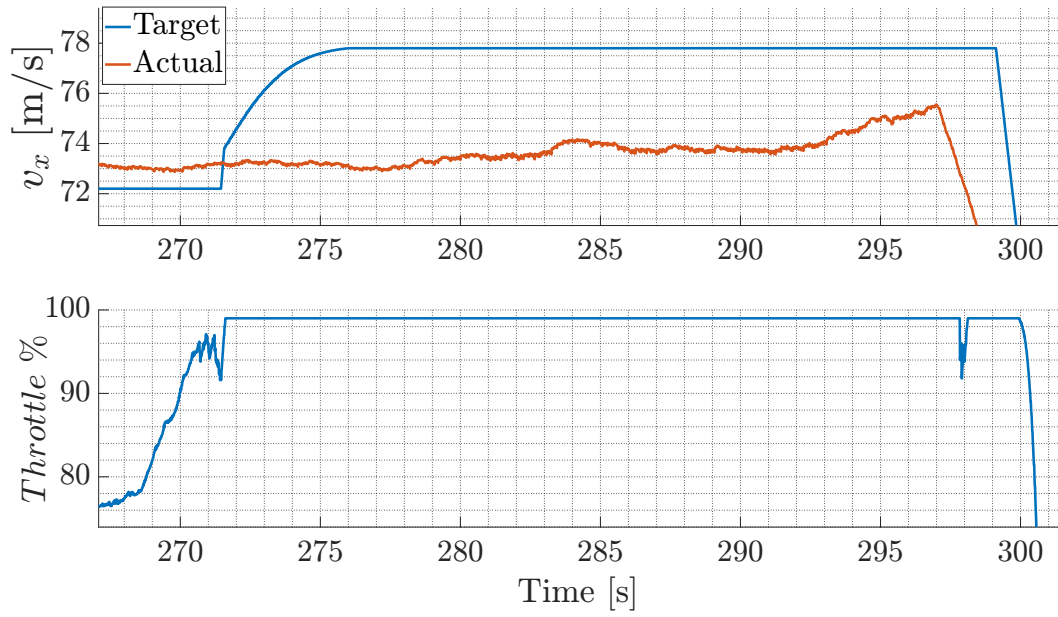
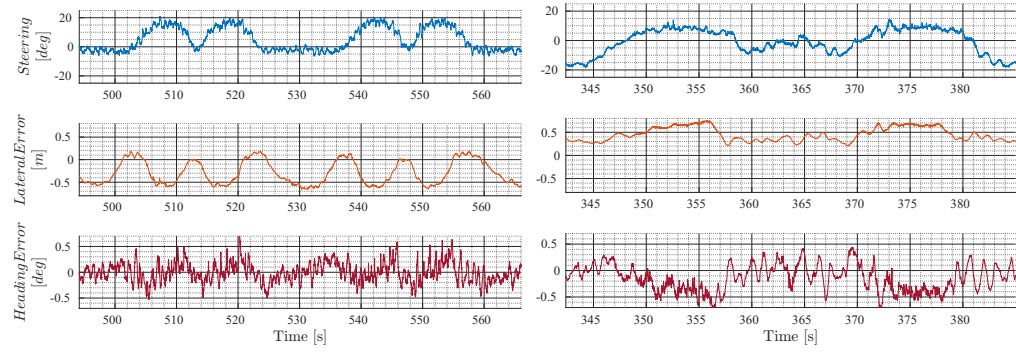


Figure 4.6: From Figure 4.4: powertrain issue during the high speed laps. A top speed of 75.5 m/s has been reached instead of the target speed of 77.7 m/s.

differences, mainly on the lateral error, are caused by changes in the weights' values in the MPC cost function we made, keeping unvaried the parameters of the Pacejka Magic Formula, from the first identification, due to not sufficient time to safely



(a) MPC Tracking Errors during a 215 kph lap at IMS. (b) MPC Tracking Errors during a 220 kph lap at LVMS.

Figure 4.7: Comparing controller tracking performance at a similar speed in different circuits. Despite different tuning values being used, consistent behavior can be observed.

validate on track a set of potentially more accurate coefficients.

The on-track experimental results of the controller using the tricycle model, compared to the single-track model, are presented. Similarly to [67], an Extended Kalman Filter (EKF) is used to filter all the signals of the vehicle and produce the states needed by the controller.

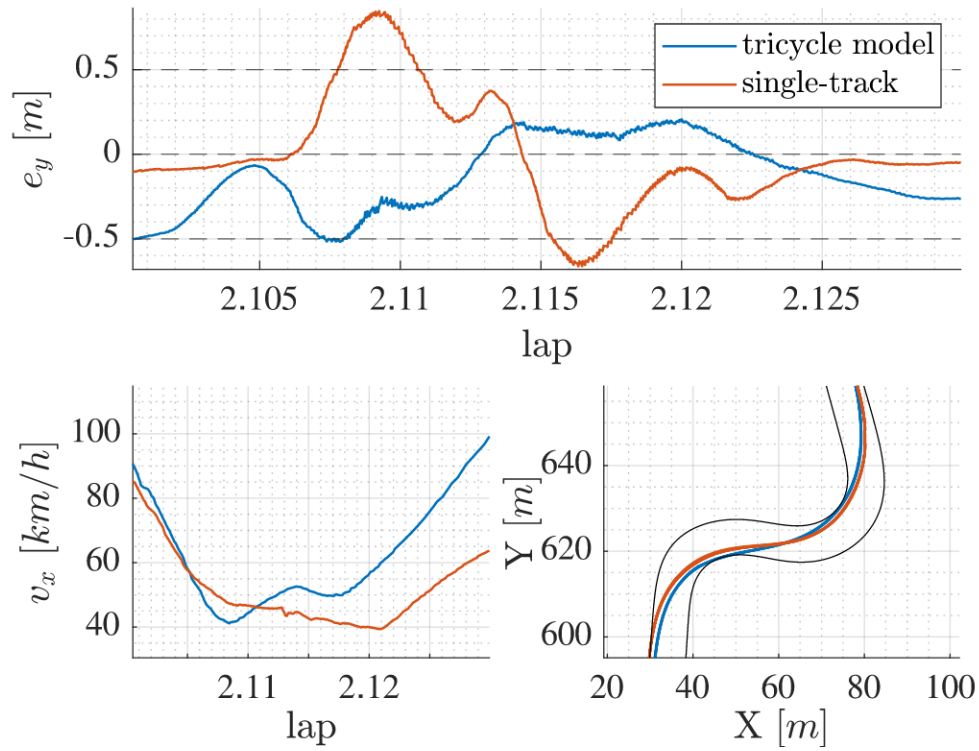


Figure 4.8: Comparison of the two models in the F1 circuit of Monza, Variante 1. See <https://youtu.be/jNc9D9T8inw?si=X4n-57CwF11DXTEE> for the on-board video of the best lap.

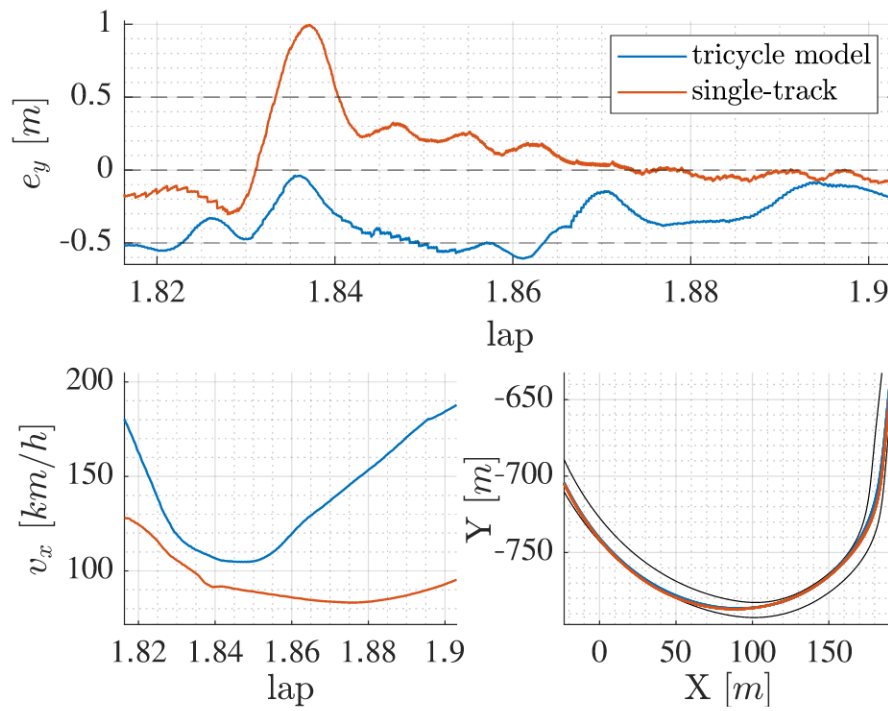


Figure 4.9: Comparison of the two models in the F1 circuit of Monza, Parabolica turn.

In these experiments, the single-track model is defined by the kinematic model in the narrow-radius curves and by the dynamic model in the wide-radius curves as satisfactory and safe tuning could not be found for all the turns of the circuit under consideration.

Figure 4.8 shows the lateral tracking error and longitudinal velocity with the two models. As expected, compared to the kinematic single-track model, the error using the tricycle model is significantly lower. Similar results are presented in Figure 4.9, where the Parabolica turn is shown. With the dynamic single-track model, the error increased beyond 1 m , forcing the car to slow down within safe limits. On the contrary, the tricycle model guaranteed a more consistent and stable behavior. As can be noticed in Figure 4.8 and Figure 4.9, both the models start the turn maneuvers with a not negligible lateral error. This has been caused mainly by the deterioration of the steering actuator performance at angles close to zero and the setup alignment of the front tires. Another limitation has been the usage of the Euler integrator. The decision was initially taken considering mainly the computational burden. However, this caused an inaccurate discretization when the tire utilization moved towards the non-linear region.

4.4.2 Highly Accurate Path Tracking at the Tire Limits

In the following, the results of the controller under good model fitting are presented. Thanks to the experimental data gathered during the first two events and the improvement in the model identification and validation procedures exploiting the Dymola multi-body simulator, it has been possible to improve the tire model and the overall vehicle model parameters. The solution demonstrated highly accurate path tracking at the tire limits both at TMS with the single-track model and at Monza with the tricycle model.

Accurate Path Tracking at TMS

In Figure 4.10, the overall performance of the controller is depicted in an entire lap on the TMS track showing a maximum lateral deviation from the path of 0.4 m. It should be noticed that the peaks are mainly in exiting turn 2 ($t \sim 1075$) and in the D-shaped straight part of the TMS track ($t \sim 1055 - 1065$ and $t \sim 1090 - 1100$). It should be noted that the steering wheel angle presents some oscillations of amplitude around 10° , which should be correctly evaluated considering the high steering ratio value of the vehicle (around 20). Nevertheless, similarly to what has been demonstrated in the previous results at IMS and LVMS, the control produced a limited heading error which is a well-suitable metric in order to verify the smoothness of the resulting maneuvers.

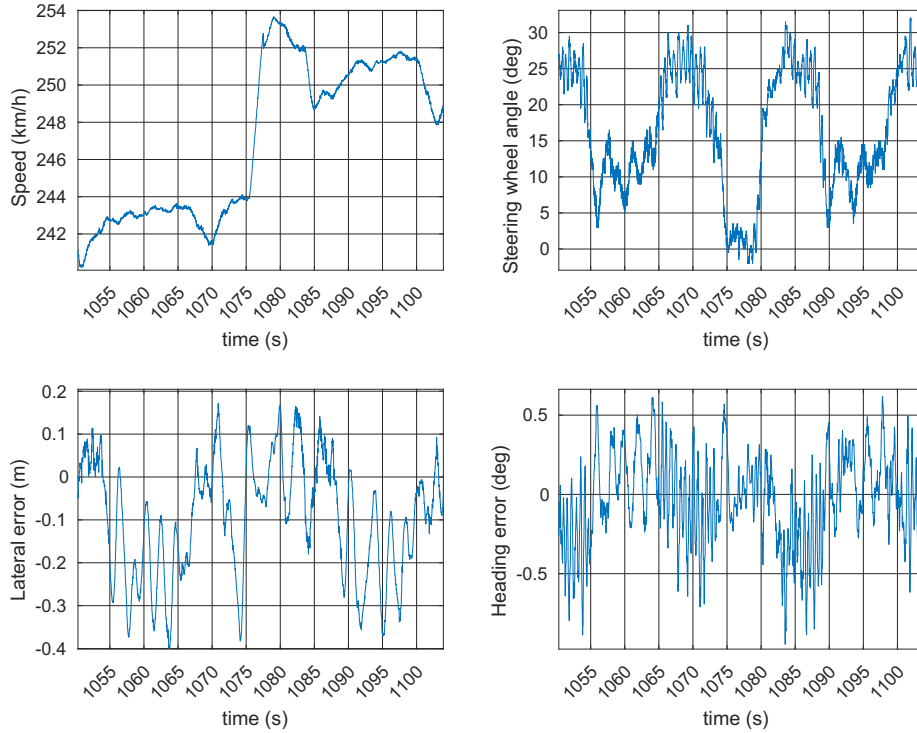


Figure 4.10: Results of a lap at the TMS oval track.

Figure 4.11 focuses in the Turn 1 and 2 of TMS, where the tire usage is close to the limits, as depicted in Figure 4.12, showing how the controller managed this

particular track section with a maximum lateral error lower than 15cm.

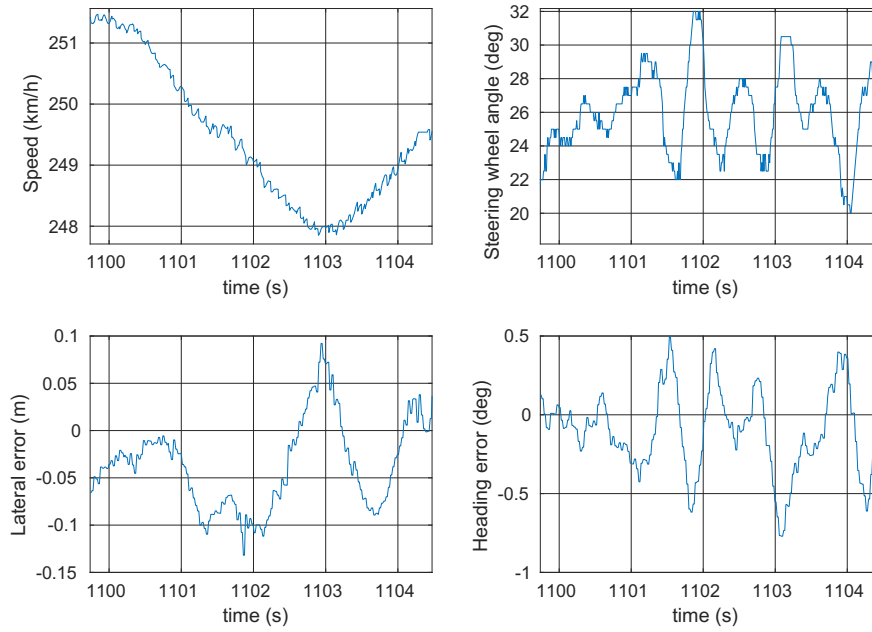


Figure 4.11: Results of the MPC at the turns 1-2 of the TMS oval track.

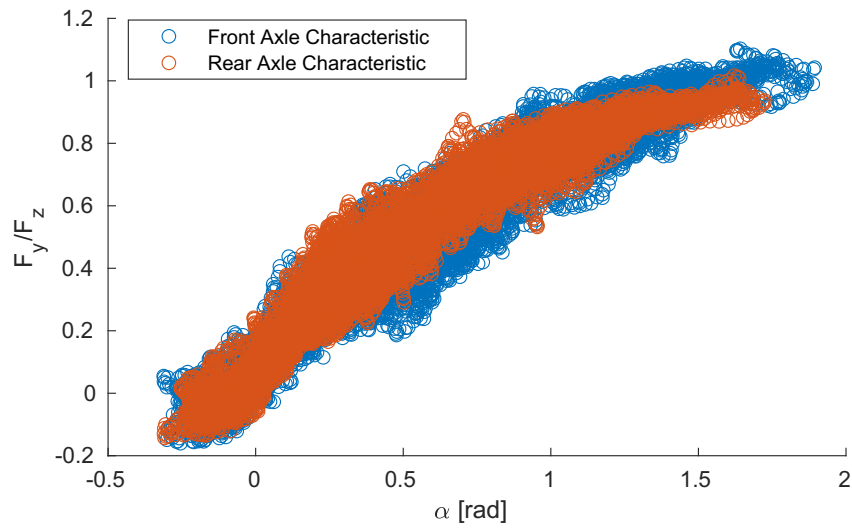


Figure 4.12: The characteristic curves of the tires. It can be seen that after a slip angle of $\sim 1.5^\circ$ the curve approaches the nonlinear region.

Accurate Path Tracking simulating Monza with FMU

Despite the interesting insights presented in Section 4.4.1 regarding the F1 Monza racetrack, due to the described limitations and the different levels of accuracy in their tuning, we do not consider those results satisfactory enough for a correct comparison of the single-track and tricycle models at their best. For this reason, and to demonstrate the performance of the proposed solution close to the very limit of the tire, we present the results in the Dymola multi-body model simulator. These have been produced using the RK4 integrator and after a further improvement of the models' fitting and minor changes on the cost tuning.

In Figures 4.13 and 4.14, a comparison between the MPC using the two different models is shown.

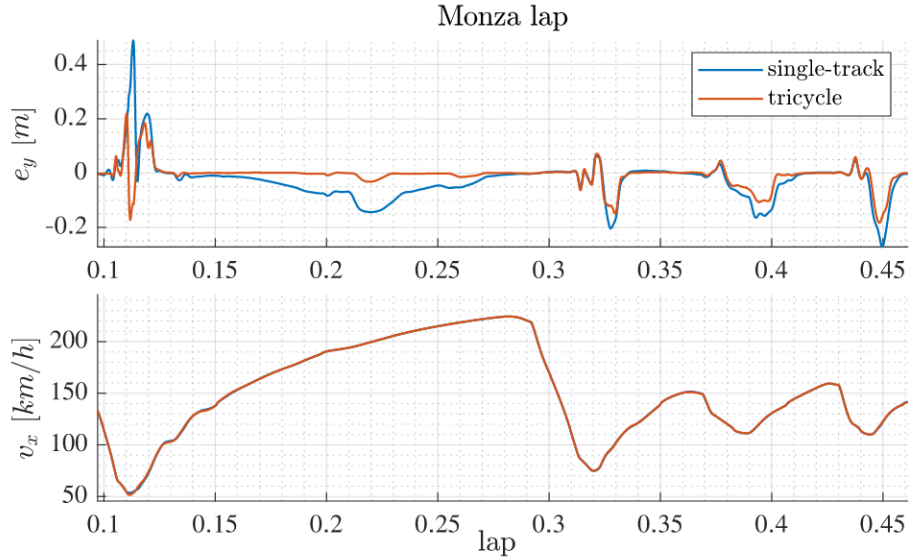


Figure 4.13: Comparison of the two models in a simulation on the 1st half of Monza F1 virtual circuit, driven by the Model Predictive Control (MPC)

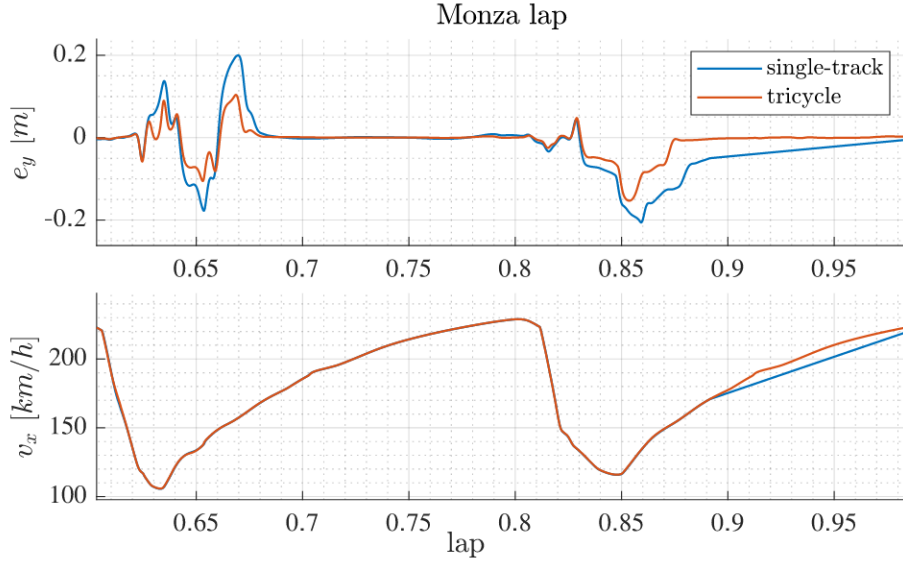


Figure 4.14: Comparison of the two models in a simulation on the 2nd half of Monza F1 virtual circuit, driven by the Model Predictive Control (MPC)

Optimal tuning of effective axle characteristics has been defined for the single-track model to allow an acceptable error throughout the entire track. The different behavior of the single-track model is still evident between slow and fast turns. On the contrary, the tricycle model is capable of making sufficiently accurate predictions over the entire lap and ensuring good consistency in the controller behavior. In particular, the MPC using the tricycle model never exceeded a lateral error of 20cm, whereas with the single-track the peak is around 60cm. Both results can be considered satisfactory depending on the objective. For an edge case application, the ideal raceline could be close to the edge of the track. In this scenario, it is important to reduce the path tracking error as much as possible to avoid touching the gravel or the track barriers, which could lead to a crash or severe instability.

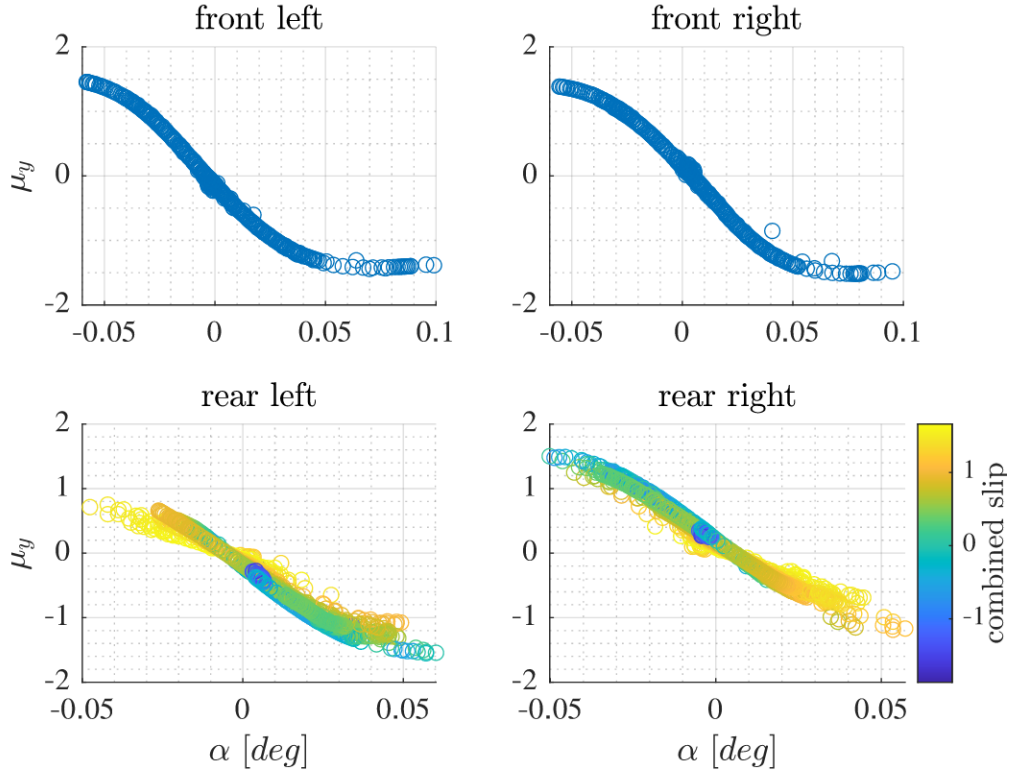


Figure 4.15: The characteristic curves of the tires exported from the telemetry of the multi-body model are displayed. It is interesting to note that the controller, with all its modules, can manage the exit phase of the curves at the traction limit.

In Figure 4.15, it is visible how the controller is able to exploit the non-linear region of the tire usage, demonstrating its effectiveness close to the vehicle handling limit. A colorbar has been included on the rear tires to highlight the effect of the longitudinal forces on the lateral characteristic. Here, positive combined slip values are associated with traction forces, whereas negative values correspond to braking forces.

4.4.3 Exceeding the Tire Limits

In Figure 4.16, the behaviour of the controller while exceeding the tire limits is depicted. Performing Turn 1-2 of TMS at a speed of 252 km/h, differently from the speed of 248 km/h previously shown in Figure 4.11, the car lost traction in the rear producing an evident oversteering. Despite the correct reaction of the

controller, as can be visible between 50s and 54s where the wheel speeds almost stabilized, the human interaction from the pitlane triggered a manual emergency procedure consisting in the maintenance of the last steering wheel angle and fixed brake commands ($t \sim 54s$). This resulted in a crash into the barrier ($t \sim 57s$).

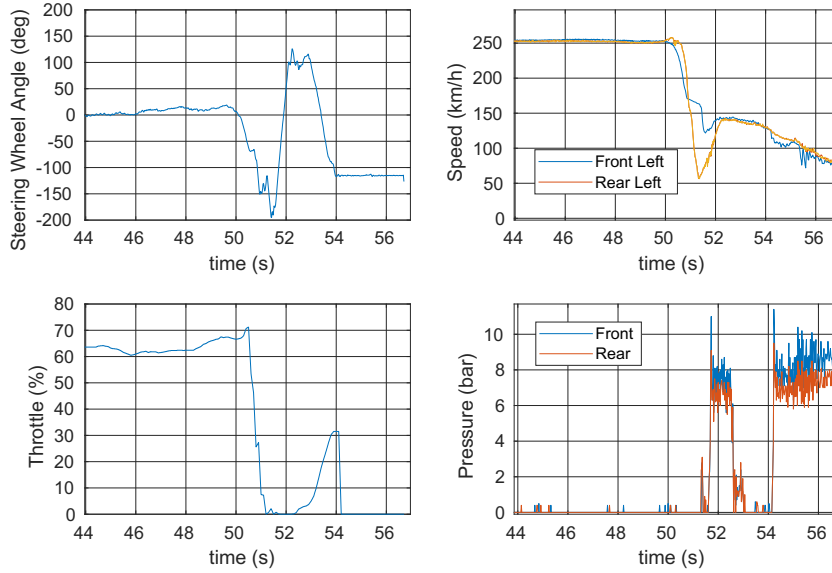


Figure 4.16: Overview of the crash at Turn 2 exit of TMS.

The main lessons learned from this run are double. First, the estimation of vehicle limits should be performed carefully considering that track conditions can vary and therefore an identification made purely offline could no longer be accurate. Second, human interaction with the control of an autonomous racetrack should be limited to speed requests. Due to delays in telemetry and the differences between the real state of the vehicle and the one estimated by the localization module in a similar edge case, visual feedback of the data available online from the pitlane is not always an adequate metric to evaluate the safety of the system.

4.5 Conclusions

This chapter introduced a Model Predictive Control (MPC) strategy for a full-scale racecar, drawing inspiration from [86]. With different models and complexity, the controller demonstrated its effectiveness under model mismatch and achieved

highly accurate path-tracking results at the tire limits after improving the model parameters identification.

Chapter 5

Model Predictive Planning and Control Scheme

5.1 MPC for Planning and Control

As briefly mentioned in Chapter 1, the majority of planning works applied to real vehicles are related to sampling-based and graph-based methods due to their ease of setting and tuning, but they lack in considering the real dynamics of the vehicle and do not scale well in the racing scenario in complex track layouts. Therefore, as a solution to overcome the limits of the Frenet-based planner presented in Chapter 3.1, a hierarchical planning and control scheme based on MPC is proposed.

5.1.1 Hierarchical Planning

The planning scheme involves the integration of a high-level overtake logic, an element designed to identify a feasible running tube. This tube is strategically utilized as a track constraint within an MPC-based Planner (MPP). The overtake logic algorithm follows a similar approach as used in [51] and [41].

The MPP leverages an offline-generated global trajectory as its guiding reference serving as a predefined path to be followed. To ensure both safety and optimal performance, the MPP imposes limitations on the velocity profile across an extensive horizon. These constraints are derived from a combination of predefined physical limits and real-time inputs provided by the user, encompassing both longitudinal and lateral acceleration bounds.

The ultimate output of this hierarchical planning approach is a trajectory that encapsulates the desired path and speed profile. This trajectory becomes the focal point for the subsequent stage of control, where a dedicated MPC takes charge. Tasked specifically with tracking the prescribed path and speed profile, this MPC ensures the execution of the planned trajectory. This hierarchical planning strategy not only refines the trajectory generation process but also establishes a robust link between strategic decision making and trajectory execution and tracking.

5.1.2 Problem Formulation

Both the model and the problem formulation of the MPP follow exactly what has been presented in Chapter 4 for the MPC. The main differences are related to the following:

- The MPP introduces extra soft constraints and allows the online adjustment of bounds associated with longitudinal velocity and acceleration. Users can dynamically update these limits, but to prevent potential infeasibility issues for the solver, soft constraints are initially established. These constraints maintain the existing bounds until the states align with the new values. The comprehensive cost function is expressed as:

$$J_{MPC}(x_t, u_t) = -\dot{s}_t + q_s * s_t^2 + q_n n_t^2 + q_\mu \mu_t^2 + q_v v_t^2 + q_{ay} |s_{v,t}| + q_r + u^T R u + B(x_t) \quad (5.1)$$

where in addition to the terms already described in Chapter 4, a weight q_{ay} is added to the slack variable $s_{v,t}$ of the upper velocity constraint. This constraint is evaluated by calculating the resulting achievable velocity, limiting each step of the horizon to the requested lateral acceleration. Without a state representing the lateral acceleration at the actual status of the developed MPP, mainly to not increase the computational time and complexity, this velocity is calculated using the curvature at each step. To mitigate computational load, the offline global trajectory undergoes modification through a forward and backward iterative algorithm before being used as a reference to the MPP, adjusting the speed profile, and ensuring that it already complies with the required constraints. This is particularly important when the state of the car is distant from the requested profile to be followed.

- The prediction horizon length is set at 150 steps, equivalent to a horizon time of 6 seconds with a sampling time of 0.04 seconds. Empirical experiments have shown that a horizon of at least 5 seconds is necessary to ensure correct compliance with constraints on lateral and longitudinal accelerations. However, the specific requirement depends on the complexity of the trajectory and the maximum achievable speed of the vehicle.
- The MPP employs different weights on the cost function compared to the MPC. In particular, more emphasis is placed on tracking the offline trajectory by penalizing high values of yaw rate. Additionally, significant importance is given to the soft constraints related to the track boundaries and acceleration limits. Furthermore, when the overtake logic does not produce an available tunnel to perform an overtake, it calculate for each step in the horizon the longitudinal progression over the path that should be tracked to avoid the collision and maintain a certain safety distance. Tracking the longitudinal progression is performed by weight q_s . The overall weighting scheme aims to balance the optimization objectives and improve the overall performance of the MPP.

5.2 Results

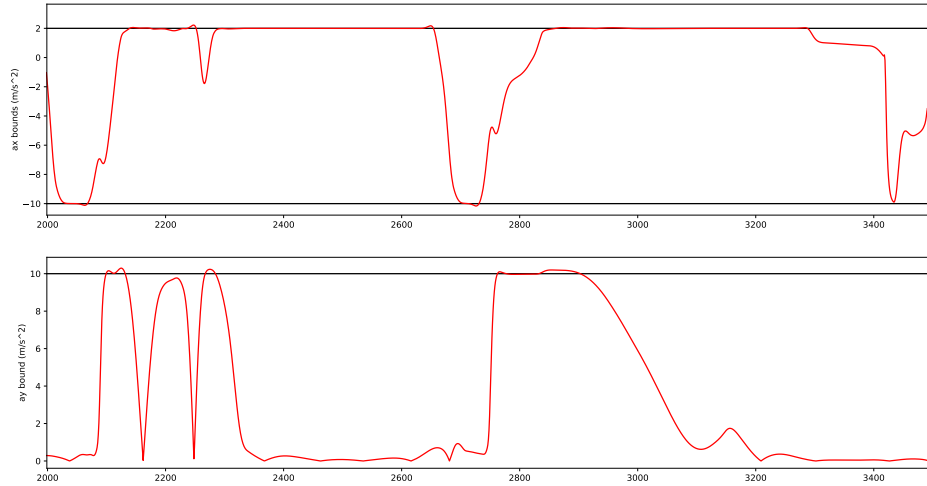


Figure 5.1: Bounds set to the MPP in two portions of the F1 Monza racetrack (Ascari and Parabolica)

The proposed method has been tested in simulation exploiting the FMU interface which includes the accurate multi-body model of the Dallara AV-21 on the Monza F1 track. The opponent racecars are simulated, making them follow the global racing line and providing their groundtruth position and motion forecasting for the next 6 seconds of the horizon.

The ability to limit the speed profile to follow precise lateral and longitudinal acceleration is depicted in Figure 5.1. In particular, the MPP has been limited to a maximum positive and negative longitudinal acceleration of respectively 2 m/s^2 and -10 m/s^2 , and a lateral acceleration of 10 m/s^2 . It can be seen that the lateral acceleration constraint is not always met accurately. The main reason is the absence of this term in the state space and the inaccuracy of the resulting maximum longitudinal velocity used in the soft constraint.

In Figure 5.2, the lateral force produced during the simulation described in this section is visible, demonstrating that the racecar performed at the tire limits where the nonlinear region of the characteristic is notable.

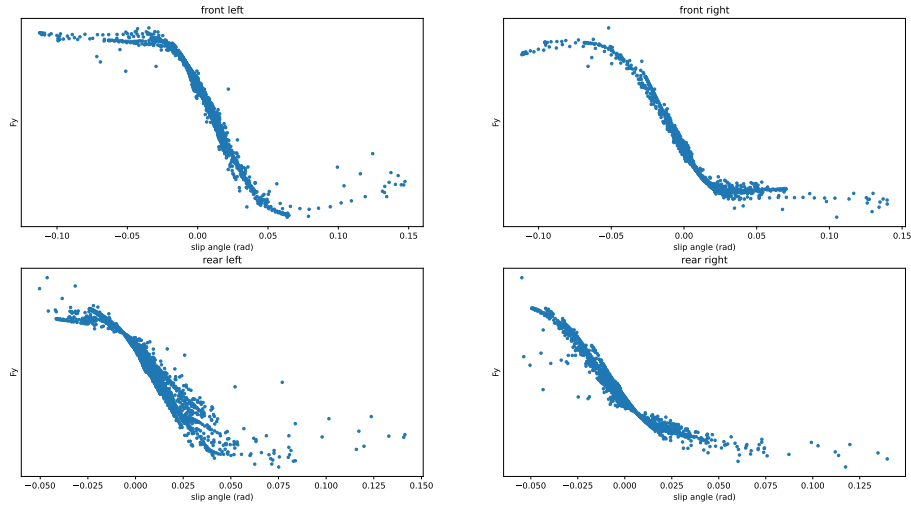


Figure 5.2: Lateral Tire Forces utilization for each wheel. The value of the lateral force is considered confidential, therefore it is not depicted.

In the following sections, three overtaking scenarios will be shown. The first represents a case where the overtake-logic does not produce a feasible tunnel; therefore, the controller should track a longitudinal progress until the overtake is feasible.

The second and third scenarios represent, respectively, an overtake on turn entry and on the middle of a fast turn. In black, the track boundaries are marked, in solid red the tunnel produced by the overtake logic, in dashed red the offline global trajectory and in blue the online trajectory of the MPC. The ego vehicle is depicted in black, while the opponent is depicted in red.

5.2.1 Keeping Distance and Overtaking on Turn Exit

In Figure 5.3, 5.4, 5.5, 5.6, 5.7 and 5.8 are depicted the steps followed by the vehicle in order to maintain a certain distance decided by the overtake logic and then overtaking at the exit of the turn where a feasible tube is produced.

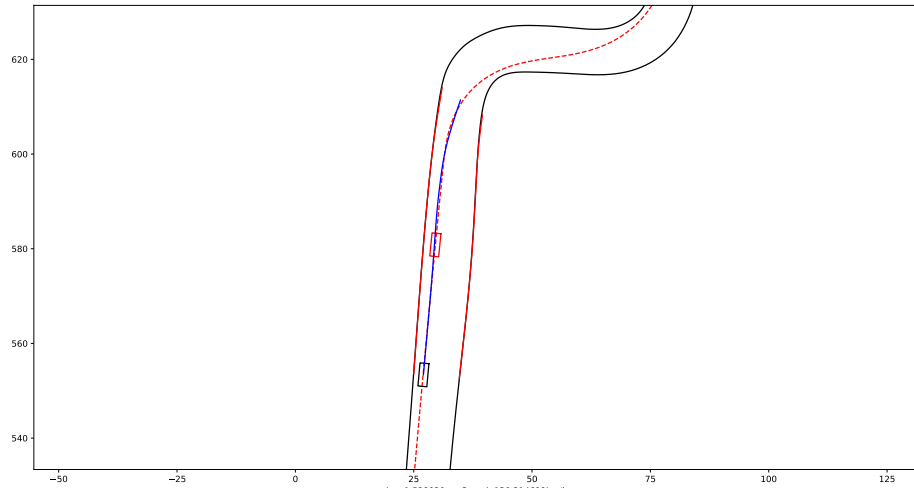


Figure 5.3: Mantaining distance to avoid collision - Variante 1 (1)

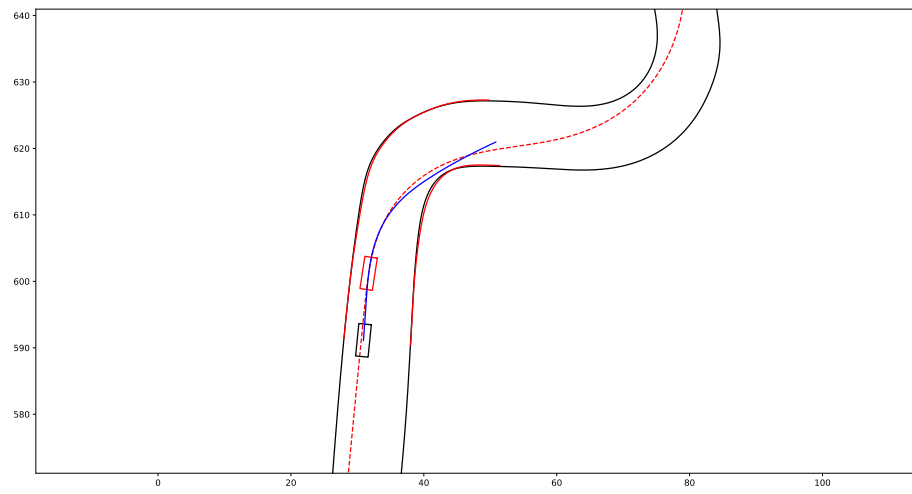


Figure 5.4: Maintaining distance to avoid collision - Variante 1 (2)

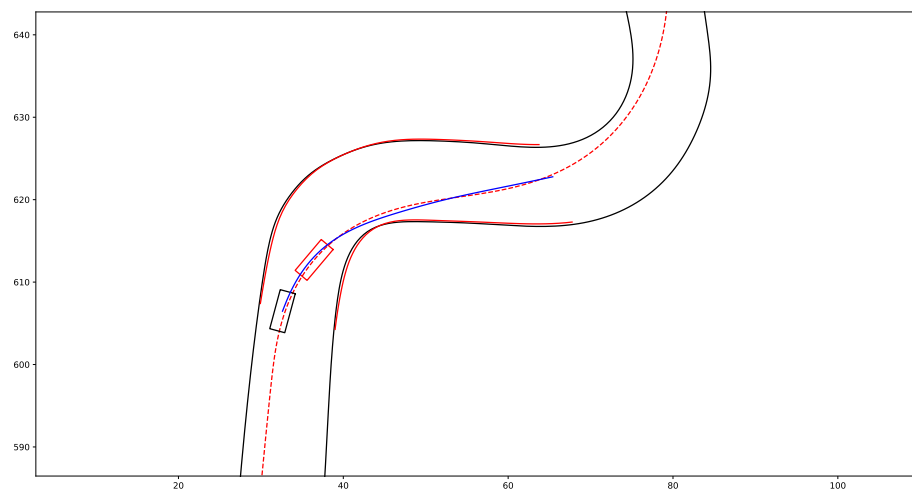


Figure 5.5: Maintaining distance to avoid collision - Variante 1 (3)

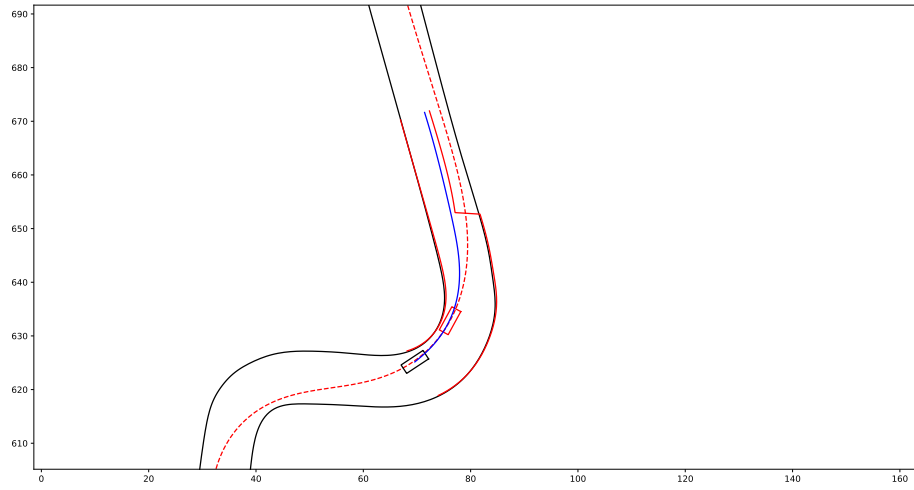


Figure 5.6: The overtake logic produced a feasible tunnel to be followed. The MPC is allowed to follow the maximum speed profile - Variante 1 (4)

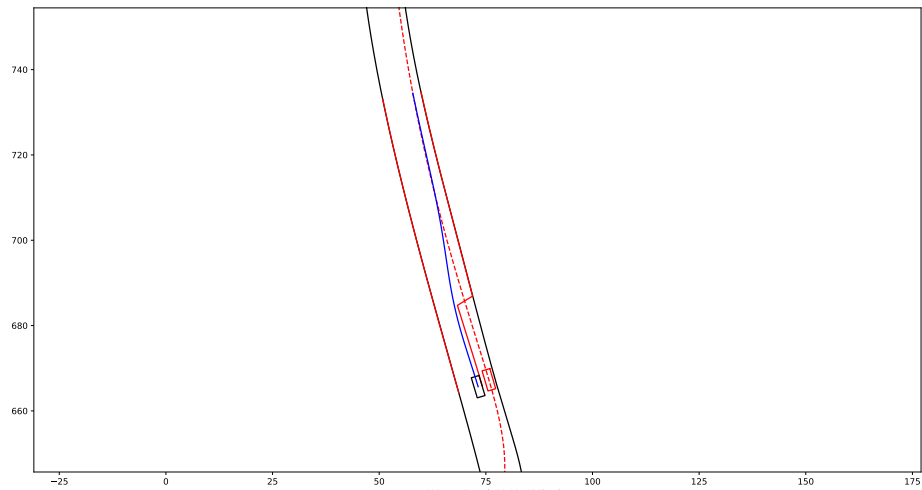


Figure 5.7: Side by side. The MPC is allowed to follow the maximum speed profile and it keeps the correct lateral distance generated by the tunnel - Variante 1 (5)

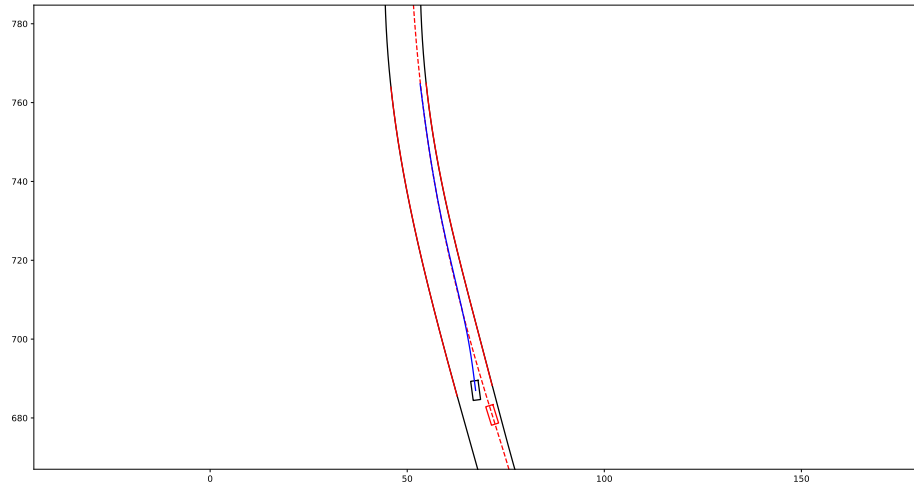


Figure 5.8: Completing the overtake - Variante 1 (6)

5.2.2 Turn Entry Overtake

In Figure 5.9, 5.10, 5.11 and 5.12, are depicted the steps while performing an overtake before the turn entry.

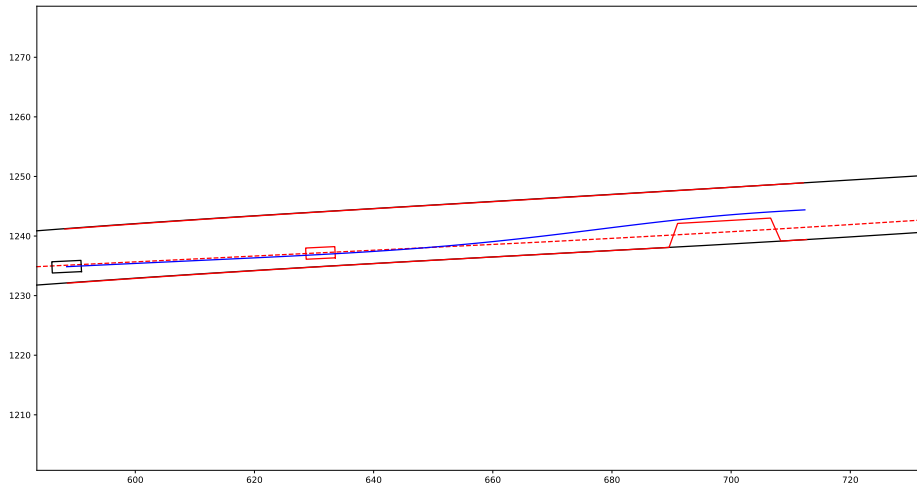


Figure 5.9: Approaching the overtake in Variante 2 entry (1)

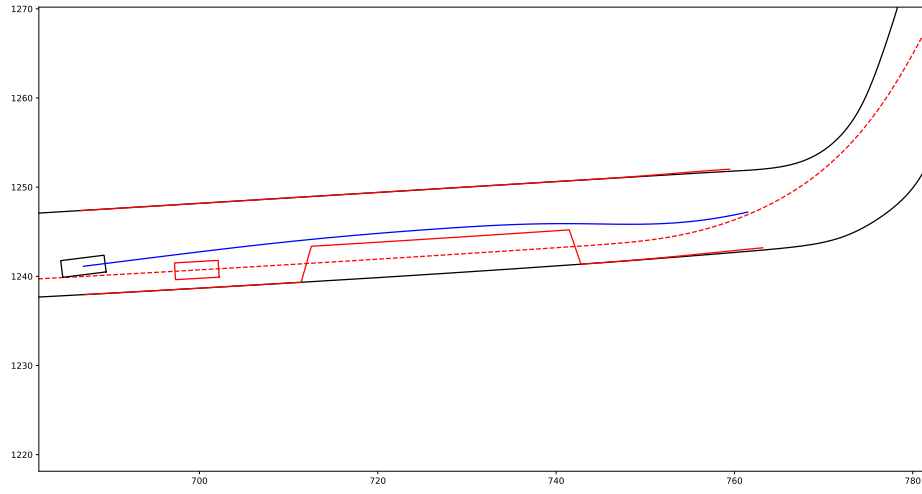


Figure 5.10: Approaching the overtake in Variante 2 entry (2)

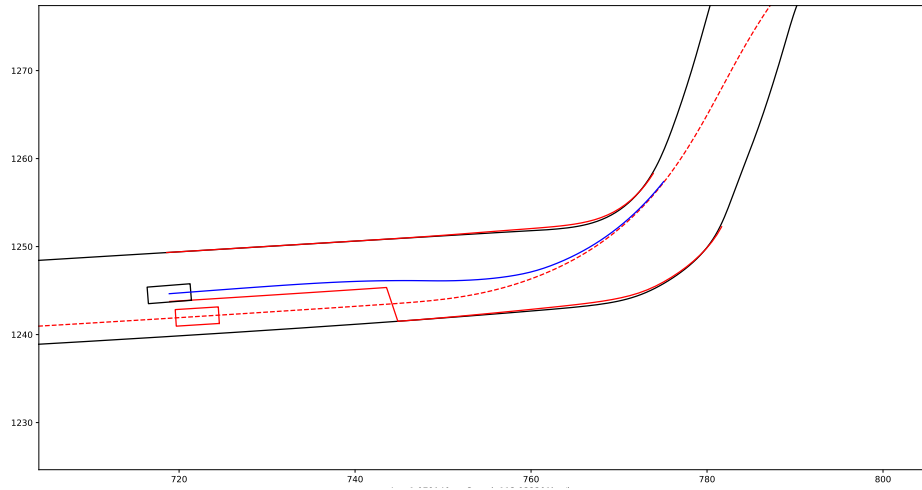


Figure 5.11: Side by side. The MPC keeps the correct lateral distance generated by the tunnel - Variante 2 entry (3)

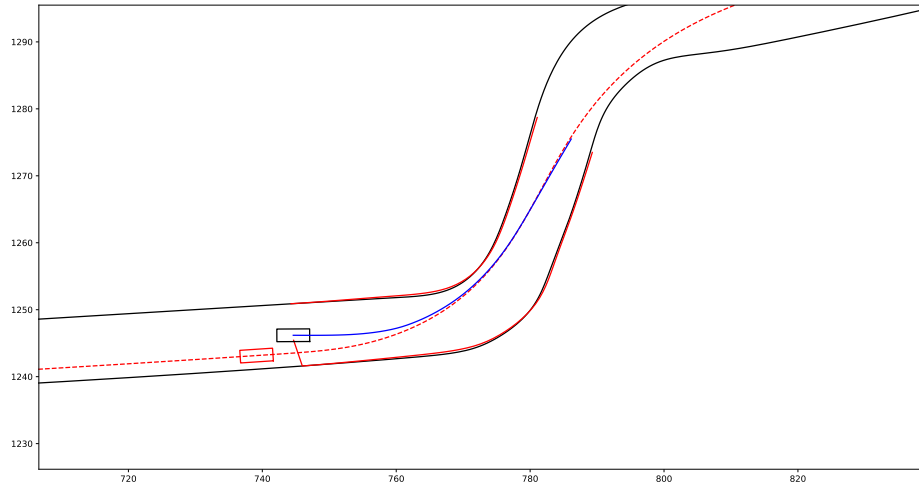


Figure 5.12: Completing the overtake in Variante 2 entry (4)

5.2.3 Overtake on Turn

The capability of performing an overtake in the middle of a turn is depicted in Figure 5.13, 5.14, 5.15 and 5.16.

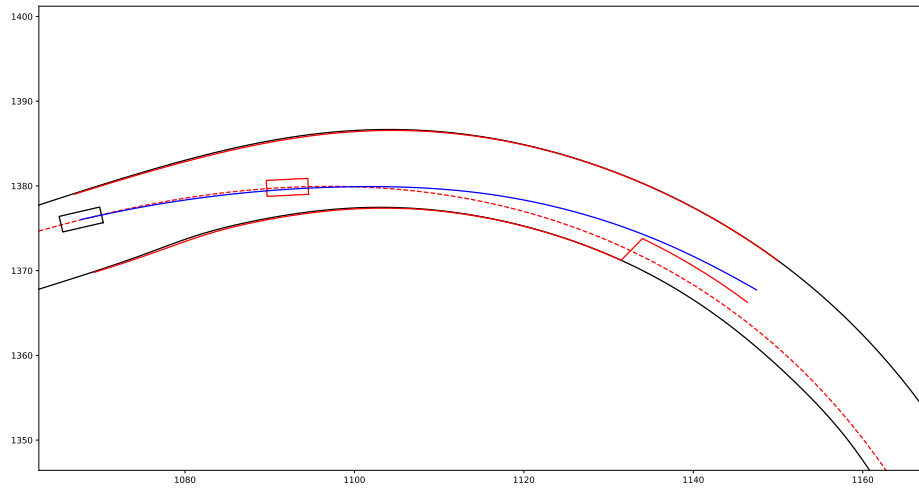


Figure 5.13: Approaching the overtake - Lesmo 1 (1)

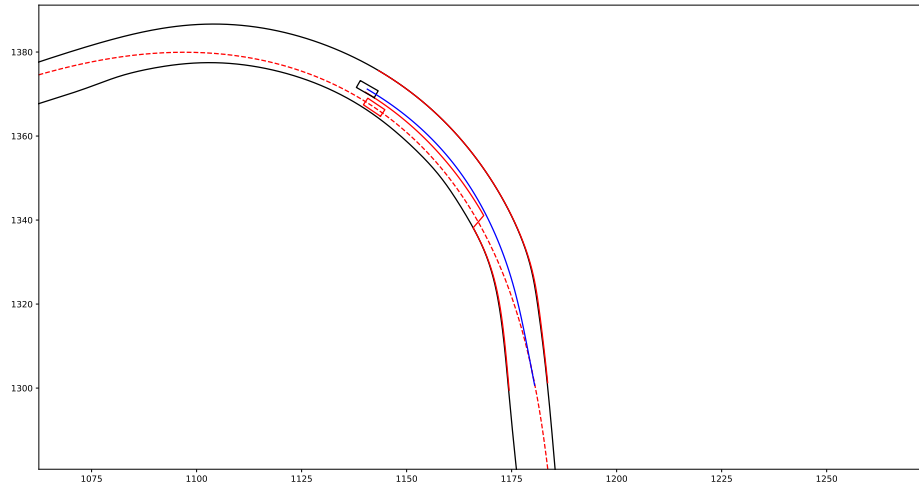


Figure 5.14: Side by side. MPC initializes the overtake maintaining the lateral distance - Lesmo 1 (2)

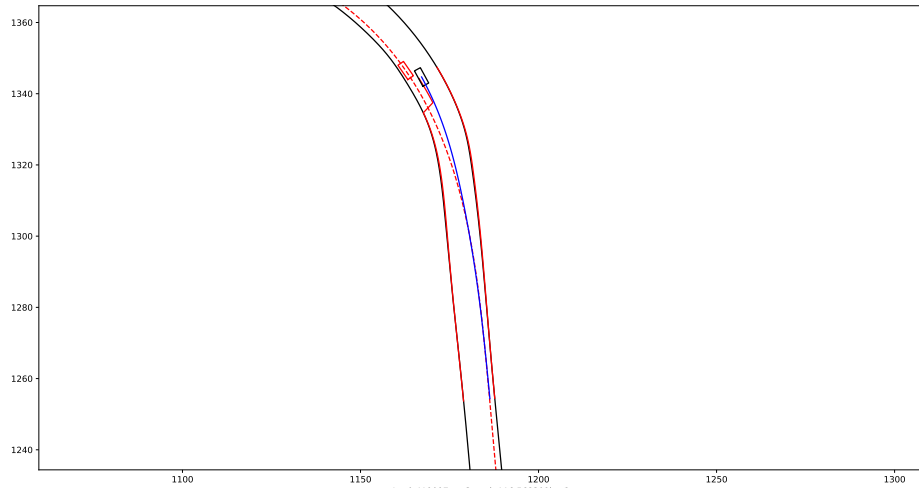


Figure 5.15: Side by side. MPC performing the overtake and maintaining the lateral distance - Lesmo 1 (3)

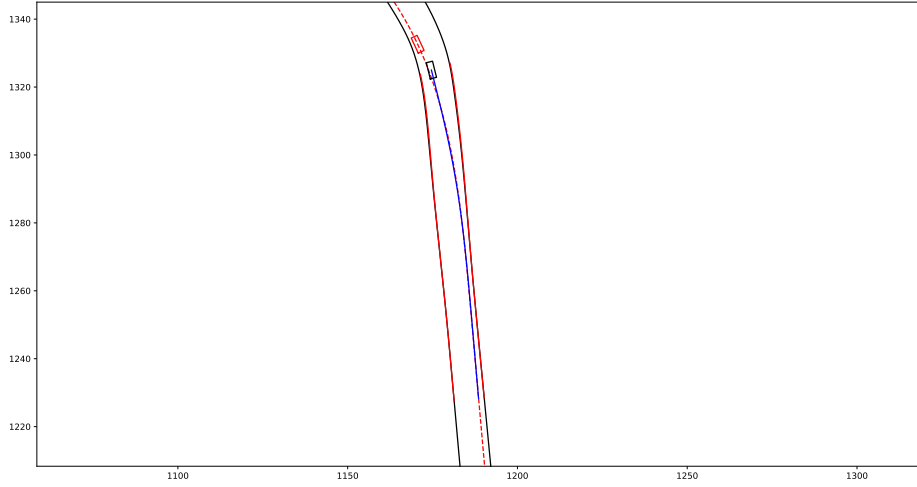


Figure 5.16: Completing the overtake - Lesmo 1 (4)

5.3 Conclusions

The proposed Model Predictive Planning and Control Scheme (MPP-MPC) introduces a hierarchical planning approach with a high-level overtake logic. The simulation results on the Monza F1 track demonstrate the effectiveness of the solution while utilizing the maximum available lateral tire forces. Scenarios, such as maintaining distance, overtaking, and navigating turns, illustrate the versatility of the approach. Although the MPP shows promising performance, future work should focus on improving the accuracy of lateral acceleration constraints and refining the algorithm for deviations from the desired profile. In general, this hierarchical approach could potentially contribute significantly to the advancement of autonomous car racing capabilities with potential applications in fully free racing. The high-level overtake logic makes the solution scalable, maintaining the same complexity and computational effort of the optimization-based modules independent of the number of agents involved in the race.

Chapter 6

Execution Time Analysis

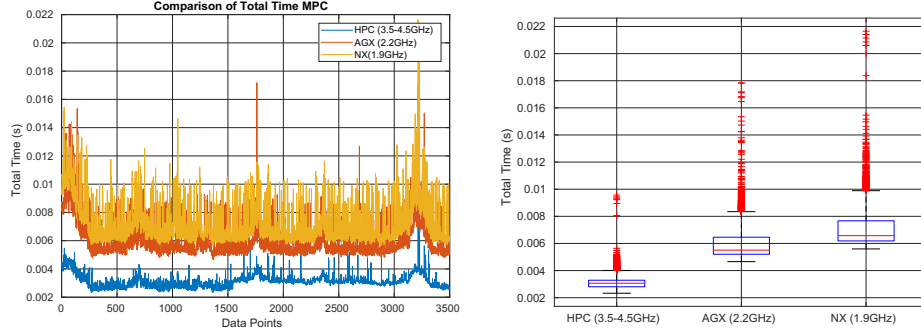
Concerns about MPC execution time usually revolve around the need for real-time responsiveness, adapting to diverse computing platforms, and managing the trade-off between computational accuracy and speed. Addressing these concerns is crucial to the successful deployment of MPC in time-sensitive and resource-constrained applications. To this extent, in the following chapter, a brief analysis of the total computation time of the MPC-base planning (MPP) and MPC proposed in this thesis is presented, considering also two Nvidia embedded boards, the Xavier NX and the Xavier AGX.

The experiments have been performed using in the planner and controller, respectively, a horizon of 63 and 150 steps resulting in 2.5s and 6s time horizon length, aiming at achieving control at 100Hz and plan at 20Hz which is usually the maximum rate of the obstacle detection and forecasting pipelines.

6.1 HPC CPU and Nvidia Embedded Platforms

In Figure 6.1 the computational time of the MPC for the control module is depicted. The time includes all the phases of the algorithm, therefore, the model discretization and the solver time. For the HPC the median value is 0.003s, the minimum value is 0.0023s, and the maximum time is 0.0095s, which as can be seen from the boxplot can be considered part of the outlier group (198 over 3500 points) occurred mainly due to the hardware frequency fluctuation and edge-cases where the states were close to the bounds. The two Nvidia boards resulted in, respectively, a median value of

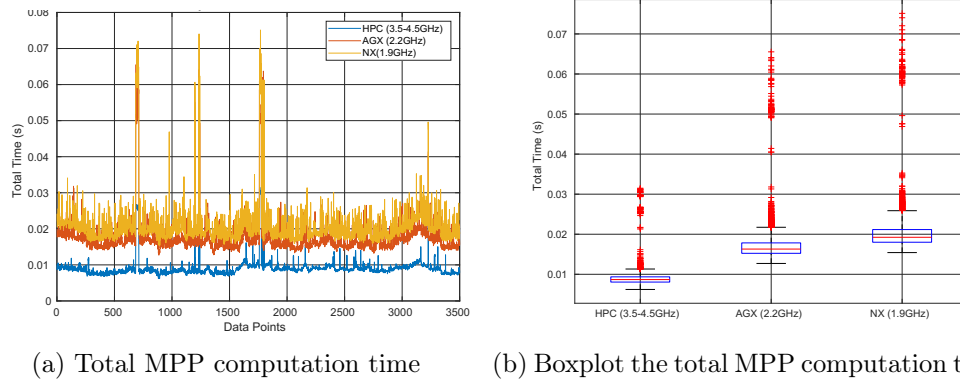
0.0055s and 0.0065s, minimum value of 0.0046s and 0.0056s, and maximum time of 0.018s and 0.021s. The results confirm the feasibility of applying the proposed MPC in real-time on the Nvidia boards.



(a) Total MPC computation time (b) Boxplot the total MPC computation time

Figure 6.1: Comparing the execution time of the MPC in an HPC CPU and two Nvidia Embedded boards.

In Figure 6.2 the computational time of the MPP for the planning module is shown. For the HPC the median value is 0.0086s, the minimum value is 0.0062s and a maximum time of 0.031s, with an outlier group of 138 items over 3500 points. The two Nvidia boards resulted in, respectively, a median value of 0.016s and 0.019s, minimum value of 0.012s and 0.015s, and maximum time of 0.065s and 0.075s. The results confirm the feasibility of applying the proposed planner in real-time on the Nvidia boards and open the possibility of extending the complexity of the used model with additional states and the increment of the horizon length if needed.



(a) Total MPP computation time (b) Boxplot the total MPP computation time

Figure 6.2: Comparing the execution time of the MPP in an HPC CPU and two Nvidia Embedded boards.

6.2 Profiling

A brief analysis of the MPC calls in the main code has been performed using Valgrind [61], a tool for memory debugging and profiling. For simplicity, it has been analyzed a run formed by 200 iterations of the MPC used for the control with a horizon of 63 steps. As can be seen in Figure 6.3, the 16% of the effort is on setting the MPC problem, which consist in setting each stage of the MPC horizon with the correct initial guess, bounds, constraints, and costs. Furthermore, at this step the nonlinear model is discretized and therefore linearized using the RK4 integrator. In particular, the calculation of the Jacobian is the heaviest part at this step.

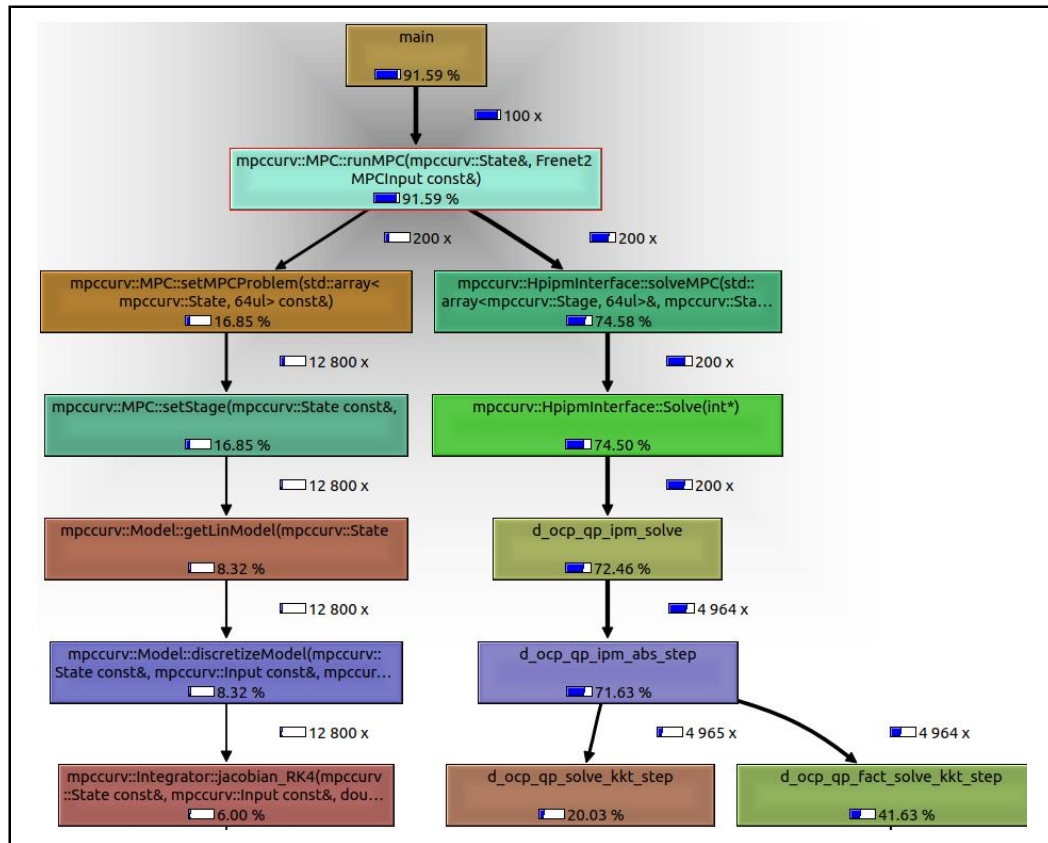


Figure 6.3: First level graph of the calls of the main implementation of the MPC in Valgrind.

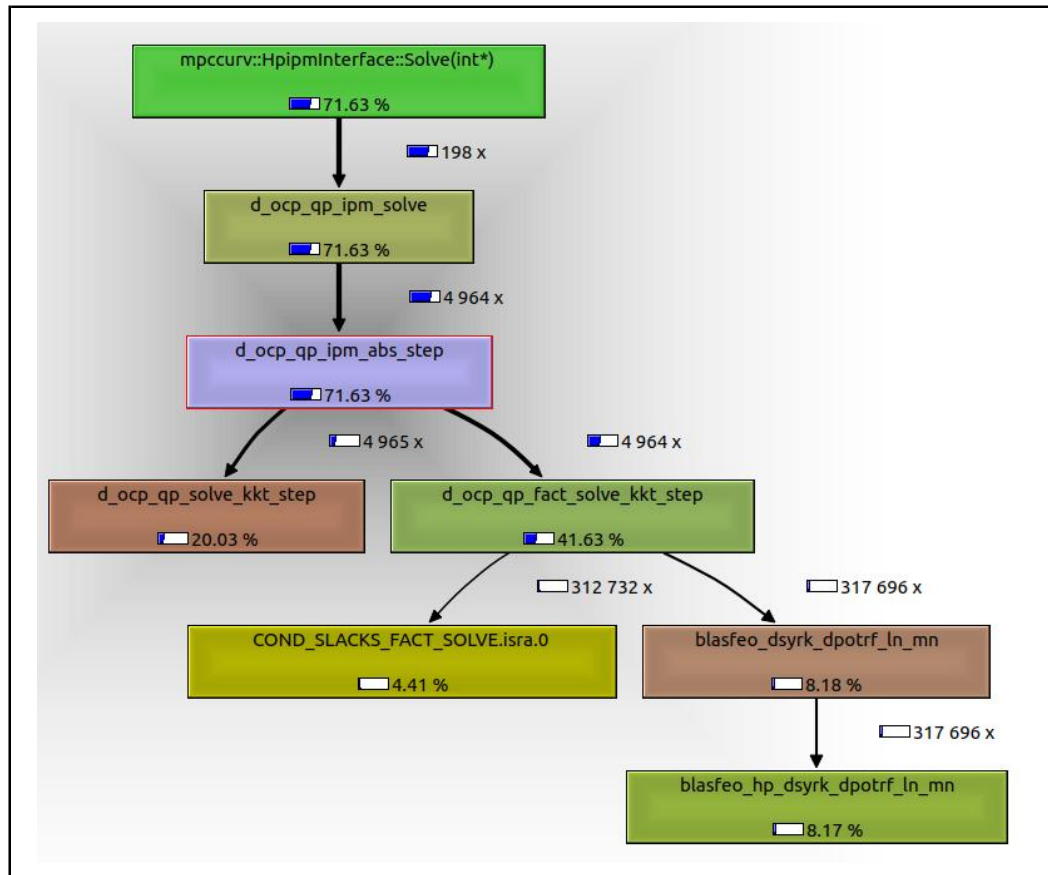


Figure 6.4: Solve call sublevel graph in Valgrind.

As expected, the call to the solver interface is the most expensive. In Figure 6.4, it is shown how the custom interface to the HPIPM solver is formed. In particular, 20% of the entire execution is spent solving the Karush-Kuhn-Tucker (KKT) conditions, while 41% in their factorization. Deeper graphs of the KKT factorization and solve steps are shown, respectively, in Figure 6.5 and Figure 6.6. It is important to notice that these steps produce at each iteration thousands of calls to BLASFE0 routines. Even if this library is tailored for embedded optimization, its subroutines are only single-threaded. Therefore, a further optimization through a parallelization of these subroutines could be possible.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

In this thesis, we introduced various methods for motion planning and control within the domain of autonomous car racing.

Initially, an extension of a popular planning approach, based on the Frenet frame and the creation of a tempospatial graph, was presented. This solution demonstrated the capability to perform active overtakes in a head-to-head racing scenario and static obstacle avoidance on oval racetracks.

For tracking the path and speed profile generated by the planner, we proposed an MPC utilizing a single-track model formulated in curvilinear coordinates. By appropriately modeling combined forces and tuning the controller, the autonomous racecar achieved speeds of up to 270 km/h and lateral accelerations exceeding 2 g, even in the presence of model mismatch and, therefore, lateral error. Further enhancements, including a better-fitted tire model and representation of the yaw moment produced by the locked differential of the Dallara AV-21, enabled the controller to achieve precise path tracking, pushing the limits of tire performance.

To address the limitations of the Frenet frame-based planner in more complex track layouts, we introduced a hierarchical planning and control approach based on MPC. The planning module is divided into a high-level overtake logic that produces track constraints for an MPC-based planner (MPP) to safely overtake another moving racecar. The trajectory generated by MPP is then tracked using the MPC introduced earlier. The primary distinctions between MPP and MPC lie in the

horizon length and constraints, while the model used remains consistent, facilitating interaction and final tracking.

The execution time of both MPP and MPC was analyzed for Nvidia Embedded boards. Leveraging automatic differentiation with ADCodeGen, utilizing the fastest option of HPIPM, and separating high and low-level control, allowed the solutions to meet the 10 ms and 30 ms execution time targets set for the controller and planner, respectively, across all boards. The profiling of the algorithm provided insights into potential further reductions in overall execution time.

Given the current state-of-the-art, the studies presented in this thesis build upon existing knowledge, extending findings related to the capabilities of Model Predictive Control (MPC)-based approaches. The contributions aim to push the boundaries of handling and precise path tracking, marking a significant milestone by showcasing the effectiveness of such approaches in a full-scale autonomous racecar. This contributes novel insights to vehicle modeling and in particular its integration into the MPC problem formulation, which usually limits the complexity of the models. The pipeline based on Hierarchical Planning and Model Predictive Control (MPP-MPC) established in this work serves as a robust foundation, offering a valuable baseline for achieving fully autonomous racing scenarios involving multiple agents.

7.2 Future Work

7.2.1 Online Tire Model Estimation

Despite achieving accurate path tracking, as demonstrated in the results in Chapter 4.4, maintaining consistent performance across varying tracks and racecars requires the inclusion of an online estimation of tire model parameters within the entire planning and control scheme. This is also important to avoid potential crashes similar to the one described in Section 4.4.3. The literature has actively employed model estimation in controllers, as seen in [21], and passively in [75]. The latter's tire model formulation aligns with our work, making it a valuable foundation for extending dynamic online estimation of Pacejka tire parameters. Attention should be paid to selecting metrics to validate the correctness of new estimated values and determining appropriate safety thresholds before updating the model in the controller.

7.2.2 MPP-MPC interaction and Multivehicle Free Racing

Further analysis of the interaction between MPP and MPC is necessary to assess the possibility of extending the planning module with a more detailed model and to adapt to some edge cases such as car instability, going off-track or encountering other agents abruptly blocking the way. These conditions should be considered to allow the solution to adapt to fully autonomous racing scenarios with more than two vehicles on the track. In particular, behavioral planning is a module that has not been thoroughly examined in the literature in the context of autonomous racing.

7.2.3 Planning and Perception Uncertainty

An overlooked aspect in literature, especially for autonomous car racing, is how to consider uncertainties in perception modules within the planning and control scheme. It is crucial to make these modules aware of potential inaccuracies in obstacle detection and motion forecasting of other moving agents. This awareness should be incorporated into a high-level planning module to, for example, adjust the strategy of the MPP and overtake logic to a more conservative maneuver, waiting for the opportune moment to perform the overtake. Recent work addressing this concern is presented in [85], where the authors consider potential occlusion in perception during trajectory generation. Regarding the overall risk of maneuvering, [33] includes this metric in autonomous driving schemes, albeit in urban and highway environments. These findings can be leveraged and adapted to the racing conditions to create a perception-aware planner pipeline and reduce collision risks based on collected data and previous experience.

7.2.4 MPC Acceleration

The results presented in Chapter 6 suggest an opportunity to accelerate portions of the MPC execution. While data exchange between the CPU and GPU and matrix sizes used in optimization might not benefit significantly from parallelization on the GPU, the linear algebra operations performed by BLASFEO[31] can be further analyzed and evaluated for CPU acceleration.

Bibliography

- [1] Eugenio Alcalá, Vicenç Puig, Joseba Quevedo, and Ugo Rosolia. Autonomous racing using linear parameter varying-model predictive control (lpv-mpc). *Control Engineering Practice*, 95:104270, 2020.
- [2] M. Althoff, M. Koschi, and S. Manzingier. Commonroad: Composable benchmarks for motion planning on roads. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 719–726, Los Angeles, CA, USA, 2017.
- [3] Andres Alvarez, Nico Denner, Zhe Feng, David Fischer, Yang Gao, Lukas Harsch, Sebastian Herz, Nick Le Large, Bach Nguyen, Carlos Rosero, Simon Schaefer, Alexander Terletskiy, Luca Wahl, Shaoxiang Wang, Jonona Yakupova, and Haoen Yu. The software stack that won the formula student driverless competition, 2022.
- [4] V. S. Babu and M. Behl. fltenth.dev - an open-source ros based f1/10 autonomous racing simulator. In *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*, pages 1614–1620, Hong Kong, China, 2020.
- [5] B. Balaji, S. Mallya, S. Genc, S. Gupta, L. Dirac, V. Khare, G. Roy, T. Sun, Y. Tao, B. Townsend, E. Calleja, S. Muralidhara, and D. Karuppasamy. Deep-racer: Autonomous racing platform for experimentation with sim2real reinforcement learning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2746–2754, 2020.
- [6] Jonathan Becker, Nadine Imholz, Luca Schwarzenbach, Edoardo Ghignone, Nicolas Baumann, and Michele Magno. Model- and acceleration-based pursuit

- controller for high-performance autonomous racing. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5276–5283, 2023.
- [7] Johannes Betz, Tobias Betz, Felix Fent, Maximilian Geisslinger, Alexander Heilmeier, Leonhard Hermansdorfer, Thomas Herrmann, Sebastian Huch, Phillip Karle, Markus Lienkamp, Boris Lohmann, Felix Nobis, Levent Ögretmen, Matthias Rowold, Florian Sauerbeck, Tim Stahl, Rainer Trauth, Frederik Werner, and Alexander Wischnewski. TUM autonomous motorsport: An autonomous racing software for the indy autonomous challenge, 2022.
- [8] Johannes Betz, Alexander Wischnewski, Alexander Heilmeier, Felix Nobis, Tim Stahl, Leonhard Hermansdorfer, and Markus Lienkamp. A software architecture for an autonomous racecar. In *2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring)*, pages 1–6, 2019.
- [9] Johannes Betz, Alexander Wischnewski, Alexander Heilmeier, Felix Nobis, Tim Stahl, Leonhard Hermansdorfer, Boris Lohmann, and Markus Lienkamp. *What can we learn from autonomous level-5 motorsport?: chassis.tech plus*, pages 123–146. 01 2019.
- [10] Johannes Betz, Hongrui Zheng, Alexander Liniger, Ugo Rosolia, Phillip Karle, Madhur Behl, Venkat Krovi, and Rahul Mangharam. Autonomous vehicles on the edge: A survey on autonomous vehicle racing. *IEEE Open Journal of Intelligent Transportation Systems*, 3:458–488, 2022.
- [11] Tobias Betz, Phillip Karle, Frederik Werner, and Johannes Betz. An analysis of software latency for a high-speed autonomous race car—a case study in the indy autonomous challenge. feb 2023.
- [12] T. Blochwitz, Martin Otter, Johan Åkesson, Martin Arnold, Christoph Clauss, Hilding Elmqvist, Markus Friedrich, Andreas Junghanns, Jakob Mauss, Dietmar Neumerkel, Hans Olsson, and Antoine Viel. Functional mockup interface 2.0: The standard for tool independent exchange of simulation models. 09 2012.
- [13] Torsten Blochwitz, Martin Otter, Martin Arnold, Constanze Bausch, Christoph Clauß, Hilding Elmqvist, Andreas Junghanns, Jakob Mauss, Manuel Monteiro,

- Thomas Neidhold, Dietmar Neumerkel, Hans Olsson, Jörg-Volker Peetz, and Susann Wolf. The functional mockup interface for tool independent exchange of simulation models. pages 105–114, 03 2011.
- [14] Alexander Buyval, Aidar Gabdulin, Ruslan Mustafin, and Ilya Shimchik. Deriving overtaking strategy from nonlinear model predictive control for a race car. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2623–2628, 2017.
- [15] Danilo Caporale, Alessandro Settimi, Federico Massa, Francesco Amerotti, Andrea Corti, Adriano Fagiolini, Massimo Guiggiani, Antonio Bicchi, and Lucia Pallottino. Towards the design of robotic drivers for full-scale self-driving racing cars. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 5643–5649, 2019.
- [16] Jesús Velasco Carrau, Alexander Liniger, Xiaojing Zhang, and John Lygeros. Efficient implementation of randomized mpc for miniature race cars. In *2016 European Control Conference (ECC)*, pages 957–962, 2016.
- [17] Tairan Chen, Zirui Li, Yiting He, Zewen Xu, Zhe Yan, and Huiqian Li. From perception to control: an autonomous driving system for a formula student driverless car, 2019.
- [18] Eugenio Chisari, Alexander Liniger, Alisa Rupenyan, Luc Van Gool, and John Lygeros. Learning from simulation, racing in reality. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8046–8052, 2021.
- [19] R. Craig Coulter. Implementation of the pure pursuit path tracking algorithm. Technical Report CMU-RI-TR-92-01, Carnegie Mellon University, Pittsburgh, PA, January 1992.
- [20] Jacob Culley, Sam Garlick, Enric Gil Esteller, Petar Georgiev, Ivan Fursa, Isaac Vander Sluis, Peter Ball, and Andrew Bradley. System design for a driverless autonomous racing vehicle. In *2020 12th International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP)*, pages 1–6, 2020.

- [21] J. Dallas, M. Thompson, J. Y. M. Goh, and A. Balachandran. A hierarchical adaptive nonlinear model predictive control approach for maximizing tire force usage in autonomous vehicles. *Field Robotics*, 3:222–242, 2023.
- [22] Nicolò De Rita, Alessandro Aimar, and Tobi Delbruck. Cnn-based object detection on low precision hardware: Racing car case study. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 647–652, 2019.
- [23] M. Dempsey, G. Fish, and J. G. D. Beltran. High fidelity multibody vehicle dynamics models for driver-in-the-loop simulators. In *Proceedings of the 11th International Modelica Conference*, 2015.
- [24] Mike Dempsey. Dymola for multi-engineering modelling and simulation. In *2006 IEEE Vehicle Power and Propulsion Conference*, pages 1–6, 2006.
- [25] D. den Hartog. Parameter estimation for lateral dynamic tire models using empirical inertial data: A comparative case study of gradient-based algorithms vs genetic algorithms. 2018.
- [26] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.
- [27] G. Rong et al. Lgsvl simulator: A high fidelity simulator for autonomous driving. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6, Rhodes, Greece, 2020.
- [28] F. Farroni, R. Lamberti, N. Mancinelli, and F. Timpone. Trip-id: A tool for a smart and interactive identification of magic formula tyre model parameters from experimental data acquired on track or test rig. *Mechanical Systems and Signal Processing*, 102:1–22, 2018.
- [29] G. Fish. Stability of explicit euler solvers. <https://www.claytex.com/tech-blog/>, 2020.
- [30] Gianluca Frison and Moritz Diehl. Hpipm: a high-performance quadratic programming framework for model predictive control, 2020.

- [31] Gianluca Frison, Dimitris Kouzoupis, Tommaso Sartor, Andrea Zanelli, and Moritz Diehl. Blasfeo: Basic linear algebra subroutines for embedded optimization. *ACM Trans. Math. Softw.*, 44(4), jul 2018.
- [32] Joseph Funke, Paul Theodosis, Rami Hindiyeh, Ganymed Stanek, Krisada Krittakirana, Chris Gerdes, Dirk Langer, Marcial Hernandez, Bernhard Müller-Bessler, and Burkhard Huhnke. Up to the limits: Autonomous audi tts. In *2012 IEEE Intelligent Vehicles Symposium*, pages 541–547, 2012.
- [33] Maximilian Geisslinger, Rainer Trauth, Gemb Kaljavesi, and Markus Lienkamp. Maximum acceptable risk as criterion for decision-making in autonomous vehicle trajectory planning. *IEEE Open Journal of Intelligent Transportation Systems*, 4:570–579, 2023.
- [34] E. Ghignone, N. Baumann, and M. Magno. Tc-driver: A trajectory conditioned reinforcement learning approach to zero-shot autonomous racing. *Field Robotics*, 3:637–651, 2023.
- [35] Tianyu Gu, Jarrod Snider, John M. Dolan, and Jin-woo Lee. Focused trajectory planning for autonomous on-road driving. In *2013 IEEE Intelligent Vehicles Symposium (IV)*, pages 547–552, 2013.
- [36] A. Guiggiani. *The Science of Vehicle Dynamics: Handling, Braking, and Ride of Road and Race Cars*. Springer, 2004.
- [37] J. Herman, J.M. Francis, S. Ganju, B. Chen, A. Koul, A.K. Gupta, A. Skabelkin, I. Zhukov, M. Kumskey, and E. Nyberg. Learn-to-race: A multimodal control environment for autonomous racing. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9773–9782, 2021.
- [38] Lukas Hewing, Alexander Liniger, and Melanie N. Zeilinger. Cautious nm-pc with gaussian process dynamics for autonomous miniature race cars. In *2018 European Control Conference (ECC)*, pages 1341–1348, 2018.
- [39] Xuemin Hu, Long Chen, Bo Tang, Dongpu Cao, and Haibo He. Dynamic path planning for autonomous driving on various roads with avoidance of static and moving obstacles. *Mechanical Systems and Signal Processing*, 100:482–500, 2018.

- [40] Sebastian Huch, Luca Scalerandi, Esteban Rivera, and Markus Lienkamp. Quantifying the lidar sim-to-real domain shift: A detailed investigation using object detectors and analyzing point clouds at target-level. *IEEE Transactions on Intelligent Vehicles*, 8(4):2970–2982, 2023.
- [41] Georg Jank, Matthias Rowold, and Boris Lohmann. Hierarchical time-optimal planning for multi-vehicle racing, 2023.
- [42] C. Jung, A. Finazzi, H. Seong, D. Lee, S. Lee, B. Kim, G. Kang, and H. Shim. An autonomous racing system: Design, implementation, and analysis; team kaist at the iac. *Field Robotics*, 3:766–800, 2023.
- [43] Juraj Kabzan, Lukas Hewing, Alexander Liniger, and Melanie N. Zeilinger. Learning-based model predictive control for autonomous racing. *IEEE Robotics and Automation Letters*, 4(4):3363–3370, 2019.
- [44] Juraj Kabzan, Miguel de la Iglesia Valls, Victor Reijgwart, Hubertus Franciscus Cornelis Hendrikx, Claas Ehmke, Manish Prajapat, Andreas Bühler, Nikhil Gosala, Mehak Gupta, Ramya Sivanesan, Ankit Dhall, Eugenio Chisari, Napat Karnchanachari, Sonja Brits, Manuel Dangel, Inkyu Sa, Renaud Dubé, Abel Gawel, Mark Pfeiffer, Alexander Liniger, John Lygeros, and Roland Siegwart. Amz driverless: The full autonomous racing system, 2019.
- [45] Nitin R. Kapania and J. Christian Gerdes. Design of a feedback-feedforward steering controller for accurate path tracking and stability at the limits of handling. *Vehicle System Dynamics*, 53(12):1687–1704, 2015.
- [46] Phillip Karle, Felix Fent, Sebastian Huch, Florian Sauerbeck, and Markus Lienkamp. Multi-modal sensor fusion and object tracking for autonomous racing. *IEEE Transactions on Intelligent Vehicles*, 8(7):3871–3883, 2023.
- [47] Jason Kong, Mark Pfeiffer, Georg Schildbach, and Francesco Borrelli. Kinematic and dynamic vehicle models for autonomous driving control design. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, pages 1094–1099, 2015.
- [48] Vincent A. Laurence and J. Christian Gerdes. Long-horizon vehicle motion planning and control through serially cascaded model complexity. *IEEE Transactions on Control Systems Technology*, 30(1):166–179, 2022.

- [49] Vincent A. Laurence, Jonathan Y. Goh, and J. Christian Gerdes. Path-tracking for autonomous vehicles at the limit of friction. In *2017 American Control Conference (ACC)*, pages 5586–5591, 2017.
- [50] Daegyu Lee, Chanyoung Jung, Andrea Finazzi, Hyunki Seong, and D. Hyunchul Shim. Resilient navigation and path planning system for high-speed autonomous race car, 2022.
- [51] Alexander Liniger, Alexander Domahidi, and Manfred Morari. Optimization-based autonomous racing of 1:43 scale rc cars. *Optimal Control Applications and Methods*, 36(5):628–647, 2015.
- [52] Alexander Liniger and John Lygeros. A viability approach for fast recursive feasible finite horizon path planning of autonomous rc cars. In *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control, HSCC '15*, page 1–10, New York, NY, USA, 2015. Association for Computing Machinery.
- [53] Alexander Liniger and John Lygeros. Real-time control for autonomous racing based on viability theory. *IEEE Transactions on Control Systems Technology*, 27(2):464–478, 2019.
- [54] Alexander Liniger, Xiaojing Zhang, Philipp Aeschbach, Angelos Georgioudis, and John Lygeros. Racing miniature cars: Enhancing performance using stochastic mpc and disturbance feedback. In *2017 American Control Conference (ACC)*, pages 5642–5647, 2017.
- [55] Daniele Lombardo. Motion planning algorithms for autonomous racing in multi-vehicle scenarios, 2023.
- [56] Rohan Kumar Manna, Daniel J. Gonzalez, Vishnu Chellapandi, Manuel Mar, Shyam S. Kannan, Shakti Wadekar, Eric J. Dietz, Christopher M. Korpela, and Aly El Gamal. Control challenges for high-speed autonomous racing: Analysis and simulated experiments. *SAE International Journal of Connected and Automated Vehicles*, 5(1):101–114, jan 2022.

- [57] Federico Massa, Luca Bonamini, Alessandro Settimi, Lucia Pallottino, and Danilo Caporale. Lidar-based gnss denied localization for autonomous racing cars. *Sensors*, 20(14), 2020.
- [58] Matthew McNaughton, Chris Urmson, John M. Dolan, and Jin-Woo Lee. Motion planning for autonomous driving with a conformal spatiotemporal lattice. In *2011 IEEE International Conference on Robotics and Automation*, pages 4889–4895, 2011.
- [59] Nicola Musiu. Identificazione e validazione di modello multibody di dallara f308 e sintesi di single-track per applicazioni autonomous racing, 2022.
- [60] Sherif Nekkah, Josua Janus, Mario Boxheimer, Lars Ohnemus, Stefan Hirsch, Benjamin Schmidt, Yuchen Liu, David Borbély, Florian Keck, Katharina Bachmann, and Lukasz Bleszynski. The autonomous racing software stack of the kit19d. *arXiv preprint arXiv:2010.02828*, 10 2020.
- [61] Nicholas Nethercote and Julian Seward. Valgrind: a framework for heavyweight dynamic binary instrumentation. *SIGPLAN Not.*, 42(6):89–100, jun 2007.
- [62] Matthew O’Kelly, Varundev Sukhil, Houssam Abbas, Jack Harkins, Chris Kao, Yash Vardhan Pant, Rahul Mangharam, Dipshil Agarwal, Madhur Behl, Paolo Burgio, and Marko Bertogna. F1/10: An open-source autonomous cyber-physical platform, 2019.
- [63] Matthew O’Kelly, Hongrui Zheng, Dhruv Karthik, and Rahul Mangharam. F1tenth: An open-source evaluation environment for continuous control and reinforcement learning. In Hugo Jair Escalante and Raia Hadsell, editors, *Proceedings of the NeurIPS 2019 Competition and Demonstration Track*, volume 123 of *Proceedings of Machine Learning Research*, pages 77–89. PMLR, 08–14 Dec 2020.
- [64] Hans B. Pacejka and Egbert Bakker. The magic formula tyre model. *Vehicle System Dynamics*, 21:1–18, 1991.
- [65] Brian Paden, Michal Čáp, Sze Zheng Yong, Dmitry Yershov, and Emilio Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on Intelligent Vehicles*, 1(1):33–55, 2016.

- [66] Pascal Puchtler and René Peinl. Evaluation of deep learning accelerators for object detection at the edge. In *KI 2020: Advances in Artificial Intelligence: 43rd German Conference on AI, Bamberg, Germany, September 21–25, 2020, Proceedings*, page 320–326, Berlin, Heidelberg, 2020. Springer-Verlag.
- [67] A. Raji, D. Caporale, F. Gatti, A. Giove, M. Verucchi, D. Malatesta, N. Musiu, A. Toschi, S. R. Popitanu, F. Bagni, M. Bosi, A. Liniger, M. Bertogna, D. Morra, F. Amerotti, L. Bartoli, F. Martello, and R. Porta. er.autopilot 1.0: The full autonomous stack for oval racing at high speeds. *Field Robotics*, 4:99–137, 2024.
- [68] Ayoub Raji, Alexander Liniger, Andrea Giove, Alessandro Toschi, Nicola Musiu, Daniele Morra, Micaela Verucchi, Danilo Caporale, and Marko Bertogna. Motion planning and control for multi vehicle autonomous racing at high speeds. In *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, pages 2775–2782, 2022.
- [69] Ayoub Raji, Nicola Musiu, Alessandro Toschi, Francesco Prignoli, Eugenio Mascaro, Pietro Musso, Francesco Amerotti, Alexander Liniger, Silvio Sorrentino, and Marko Bertogna. A tricycle model to accurately control an autonomous racecar with locked differential. In *2023 IEEE 11th International Conference on Systems and Control (ICSC)*, pages 782–789, 2023.
- [70] Eric J. Rossetter. A potential field framework for active vehicle lanekeeping assistance. 2003.
- [71] Matthias Rowold, Levent Ögretmen, Tobias Kerbl, and Boris Lohmann. Efficient spatiotemporal graph search for local trajectory planning on oval race tracks. *Actuators*, 11(11), 2022.
- [72] A. Saba, A. Adetunji, A. Johnson, A. Kothari, M. Sivaprakasam, J. Spisak, P. Bharatia, A. Chauhan, B. Duff Jr., N. Gasparro, C. King, R. Larkin, B. Mao, M. Nye, A. Parashar, J. Attias, A. Balciunas, A. Brown, C. Chang, M. Gao, C. Heredia, A. Keats, J. Lavariega, W. Muckelroy III, A. Slavescu, N. Stathas, N. Suvarna, C. T. Zhang, S. Scherer, and D. Ramanan. Fast and modular autonomy software for autonomous racing vehicles. *Field Robotics*, 4(1):1–45, 2024.

- [73] Andrew Saba. Development and testing of a software stack for an autonomous racing vehicle. Master's thesis, Pittsburgh, PA, December 2023.
- [74] M. Schratter, J. Zubaca, K. Mautner-Lassnig, T. Renzler, M. Kirchengast, S. Loigge, M. Stolz, and D. Watzenig. Lidar-based mapping and localization for autonomous racing. In *2021 International Conference on Robotics and Automation (ICRA 2021) - Workshop Opportunities and Challenges With Autonomous Racing*. IEEE, 2021. [Online]. Available: <https://linklab-uva.github.io/icra-autonomous-racing/contributedpapers/paper4.pdf>.
- [75] Hyunki Seong, Chanyoung Chung, and David Hyunchul Shim. Model parameter identification via a hyperparameter optimization scheme for autonomous racing systems. *IEEE Control Systems Letters*, 7:1652–1657, 2023.
- [76] Joshua Spisak, Andrew Saba, Nayana Suvarna, Brian Mao, Chuan Tian Zhang, Chris Chang, Sebastian Scherer, and Deva Ramanan. Robust modeling and controls for racing on the edge, 2022.
- [77] Sirish Srinivasan, Sebastian Nicolas Giles, and Alexander Liniger. A holistic motion planning and control solution to challenge a professional racecar driver. *IEEE Robotics and Automation Letters*, 6(4):7854–7860, 2021.
- [78] Tim Stahl, Alexander Wischnewski, Johannes Betz, and Markus Lienkamp. Multilayer graph-based trajectory planning for race vehicles in dynamic scenarios. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 3149–3154, 2019.
- [79] Stahl, Tim, Wischnewski, Alexander, Betz, Johannes, and Lienkamp, Markus. Ros-based localization of a race vehicle at high-speed using lidar. *E3S Web Conf.*, 95:04002, 2019.
- [80] Ganymed Stanek, Dirk Langer, Bernhard Müller-Bessler, and Burkhard Huhnke. Junior 3: A test platform for advanced driver assistance systems. In *2010 IEEE Intelligent Vehicles Symposium*, pages 143–149, 2010.
- [81] John K. Subosits and J. Christian Gerdes. Impacts of model fidelity on trajectory optimization for autonomous vehicles in extreme maneuvers. *IEEE Transactions on Intelligent Vehicles*, 6(3):546–558, 2021.

- [82] J. Talbot, M. Brown, and J. C. Gerdes. Shared control up to the limits of vehicle handling. *IEEE Transactions on Intelligent Vehicles*, 2023.
- [83] Hanqing Tian, Jun Ni, Zirui Li, and Jibin Hu. Autonomous formula racecar: Overall system design and experimental validation, 2020.
- [84] Alessandro Toschi. Integration of model predictive control for autonomous racing, 2022.
- [85] Rainer Trauth, Korbinian Moller, and Johannes Betz. Toward safer autonomous vehicles: Occlusion-aware trajectory planning to minimize risky behavior. *IEEE Open Journal of Intelligent Transportation Systems*, 4:929–942, 2023.
- [86] José L. Vázquez, Marius Brühlmeier, Alexander Liniger, Alisa Rupenyan, and John Lygeros. Optimization-based hierarchical motion planning for autonomous racing. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2397–2403, 2020.
- [87] Niclas Vödisch, David Dodel, and Michael Schötz. FSOCO: The formula student objects in context dataset. *SAE International Journal of Connected and Automated Vehicles*, 5(1), jan 2022.
- [88] Trey P. Weber and J. Christian Gerdes. Modeling and control for dynamic drifting trajectories. *IEEE Transactions on Intelligent Vehicles*, pages 1–11, 2023.
- [89] Moritz Werling, Julius Ziegler, Sören Kammel, and Sebastian Thrun. Optimal trajectory generation for dynamic street scenarios in a frenét frame. In *2010 IEEE International Conference on Robotics and Automation*, pages 987–993, 2010.
- [90] Alexander Wischnewski, Thomas Herrmann, Frederik Werner, and Boris Lohmann. A tube-mpc approach to autonomous multi-vehicle racing on high-speed ovals. *IEEE Transactions on Intelligent Vehicles*, 8(1):368–378, 2023.
- [91] Alexander Wischnewski, Tim Stahl, Johannes Betz, and Boris Lohmann. Vehicle dynamics state estimation and localization for high performance race cars.

- IFAC-PapersOnLine*, 52(8):154–161, 2019. 10th IFAC Symposium on Intelligent Autonomous Vehicles IAV 2019.
- [92] Haoru Xue, Edward L. Zhu, and Francesco Borrelli. Learning model predictive control with error dynamics regression for autonomous racing, 2023.
- [93] Tantan Zhang, Yueshuo Sun, Yazhou Wang, Bai Li, Yonglin Tian, and Fei-Yue Wang. A survey of vehicle dynamics modeling methods for autonomous racing: Theoretical models, physical/virtual platforms, and perspectives. *IEEE Transactions on Intelligent Vehicles*, pages 1–24, 2024.