UNIMORE
UNIVERSITÀ DEGLI STUDI DI
MODENA E REGGIO EMILIA

University of Modena and Reggio Emilia

PhD Program in
Computer and Data Science for Technological and Social Innovation

Cycle XXXVIII

Decision-Making at the Edge of Vehicle Dynamics: Advanced Planning Strategies for Autonomous Racing Cars

Candidate **Alessandro Toschi**

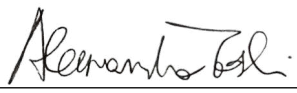
Supervisor: **Prof. Marko Bertogna**

PhD Program Coordinator: **Prof. Andrea Marongiu**

Declaration of Authorship

I, Alessandro TOSCHI, declare that this thesis titled, "Decision-Making at the Edge of Vehicle Dynamics: Advanced Planning Strategies for Autonomous Racing Cars" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: 

Date: 26/03/2026

UNIVERSITY OF MODENA AND REGGIO EMILIA

Abstract

Computer and Data Science for Technological and Social Innovation

Doctor of Philosophy

Decision-Making at the Edge of Vehicle Dynamics: Advanced Planning Strategies for Autonomous Racing Cars

by Alessandro TOSCHI

The rapid advancement of autonomous driving technology has pushed research boundaries from urban navigation to the extreme environment of autonomous racing. In this domain, decision-making systems must manage aggressive maneuvers, complex multi-agent interactions, and strict competition rules at high speeds, where vehicles operate at the limits of handling and available reaction times are minimal. This thesis presents a novel, modular planning framework designed to bridge the gap between high-level strategic reasoning and low-level control. The core contribution is the *Tunnels* architecture, which acts as a multi-stage semantic abstraction layer. It progressively translates discrete spatio-temporal inputs - including nominal track limits, ego-vehicle trajectories, and opponent predictions -into tactical states, such as opponent roles and right-of-way priorities. This semantic translation enables the decision-making module to strategically determine when to safely follow an opponent or how to initiate a rule-compliant overtaking maneuver, subsequently mapping these choices into dynamic, convex spatial corridors. By decoupling the complex, non-convex geometry of multi-agent interactions from trajectory optimization, the framework simplifies the problem landscape for the downstream Model Predictive Control (MPC), ensuring real-time feasibility and robust collision avoidance. Operating on this tactical representation allows for a highly configurable and resilient planning process. The system incorporates tunable parameters to adjust safety margins, yielding behaviors, and competitive aggressiveness based on the context. Furthermore, the framework features adaptive boundary management, seamlessly deciding when to strictly adhere to nominal track limits or when to actively exploit paved run-off areas to evade critical, emergency situations. The proposed system has been extensively validated on full-scale autonomous racecars, demonstrating its effectiveness in two international competitions: the Indy Autonomous Challenge (IAC) with the Dallara AV-24 on high-speed ovals, and the Abu Dhabi Autonomous Racing League (A2RL) with the Dallara Super Formula EAV25 on technical road courses. Experimental results confirm the framework's ability to execute safe, high-speed overtaking maneuvers in adversarial multi-agent scenarios, providing a scalable foundation for future safety-critical autonomous systems. Additional resources, including demonstration videos and the forthcoming open-source code, are available on the project website: https://atoschi.github.io/tunnels-framework/.

Contents

List of Figures

List of Tables

List of Abbreviations

IAC	**I**ndy **A**utonomous **C**hallenge
A2RL	**A**bu Dhabi **A**utonomous **R**acing **L**eague
ROW	**R**ight **O**f **W**ay
MPC	**M**odel **P**redictive **C**ontrol
MPP	**M**odel **P**redictive **P**lanner
OCP	**O**ptimal **C**ontrol **P**roblem
CoG	**C**enter **o**f **G**ravity
OTM	**O**ver**T**ake **M**argin
LVMS	**L**as **V**egas **M**otor **S**peedway
IMS	**I**ndianapolis **M**otor **S**peedway
HPIPM	**H**igh **P**erformance **I**nterior **P**oint **M**ethod
ADAS	**A**dvanced **D**river **A**ssistance **S**ystems

Chapter 1

Introduction

1.1 Planning in Autonomous Driving

The widespread introduction of Autonomous Vehicles (AVs) and Advanced Driver Assistance Systems (ADAS) is primarily driven by the objective of reducing motor vehicle crashes and fatalities. While current commercial systems have successfully mitigated risks in standard driving conditions, a significant number of accidents remain caused by factors such as loss of control, harsh weather, or complex multi-agent interactions that exceed current capabilities. In this context, the motion planning module acts as the critical bridge between perception and actuation, serving as the ultimate guarantor of vehicle safety.

The fundamental role of the planner is to calculate a trajectory that transitions the vehicle from its current state to a desired goal while strictly satisfying dynamic constraints and avoiding collisions. However, ensuring safety is not merely about obstacle avoidance; it requires the ability to reason about uncertain environments. In real-world scenarios, the planner must navigate interactions where the future behavior of other road users, whether human-driven or autonomous, is not deterministic. Consequently, a robust planning framework must balance the efficiency of the maneuver with safety margins that account for prediction uncertainty and perception noise.

To manage the computational complexity inherent in these safety-critical systems, the planning architecture is typically organized hierarchically:

- **Global Planning:** This layer operates at a macroscopic level, calculating an optimal route or reference path offline based on static map data. Its primary goal is to provide a long-term strategic guide.
- **Local Planning:** This layer operates online in real-time. It is responsible for generating the immediate trajectory, handling high-frequency updates to avoid dynamic obstacles and ensure vehicle stability.

This thesis focuses on the challenges associated with the local planning layer. Specifically, it addresses the problem of making safe, high-speed decisions in dynamic environments where the interaction with other agents is a dominant factor. By developing frameworks that can operate at the limits of handling, we aim to validate safety concepts that are transferable to the broader context of autonomous driving.

1.2 The Context of Autonomous Racing

Autonomous racing serves as the ultimate proving ground for high-speed robotics, pushing the boundaries of perception, planning, and control algorithms in safety-critical environments. While the foundations of autonomous driving were laid by milestones such as the DARPA Grand Challenges (2004-2007), which focused on navigating static unstructured environments, and later by Roborace, which introduced the concept of electric autonomous motorsport, the field has recently evolved towards complex multi-agent scenarios. Modern competitions require vehicles not only to drive at the limits of handling but also to interact strategically with opponents. From a research perspective, autonomous racing constitutes a valuable testbed for validating planning algorithms under extreme conditions. Solutions developed in this domain must explicitly handle multi-agent interactions and operate under tight spatial and temporal constraints, remaining robust to prediction errors. Despite the performance oriented nature of racing, both standard autonomous driving and motorsport share the fundamental requirement of minimizing collision risks while complying with externally imposed rules. In urban scenarios, these constraints are dictated by traffic laws and safety standards; in racing, they are defined by competition regulations and the code of conduct. Consequently, planners must adhere to these rules while implicitly cooperating with other agents to avoid accidents, even within a competitive setting. This unique dualism forces the planning framework to continuously balance the ego vehicle's performance objectives with the necessity of safe interaction. This thesis focuses on the challenges and solutions developed within the context of two major international competitions: the Indy Autonomous Challenge (IAC) and the Abu Dhabi Autonomous Racing League (A2RL). Additionally, the proposed framework has undergone preliminary testing on the Roboracer platform, although results from this specific domain are reserved for future work.

1.2.1 Indy Autonomous Challenge (IAC)

The Indy Autonomous Challenge[1] (IAC) is the first international competition to feature full-scale autonomous open-wheel racecars. Launched in 2021 with an event at the Indianapolis Motor Speedway, the league utilizes the **Dallara AV-24** (an evolution of the initial AV-21), a platform based on the Indy Lights IL-15 chassis. This platform is optimized for high-speed sustained running, capable of reaching top speeds exceeding 300 km/h. While the IAC has explored road courses (e.g., Monza), its primary focus remains on high-speed ovals and tri-ovals such as Indianapolis, Las Vegas, and Texas Motor Speedway. The competition format is predominantly structured around "passing battles": head-to-head scenarios where two vehicles take turns attempting to overtake each other at progressively increasing speeds. Although multi-vehicle events with four cars have been conducted, they have typically been confined to specific exhibition formats rather than free racing. This structure forces the vehicles to operate in close proximity at extreme velocities.

[1] https://www.indyautonomouschallenge.com/

1.2.2 Abu Dhabi Autonomous Racing League (A2RL)

The Abu Dhabi Autonomous Racing League[2] (A2RL) represents the newest frontier in autonomous motorsport. It represents a distinct approach to autonomous motorsport, focusing on "road course" racing at the Yas Marina Circuit. The official vehicle is the EAV24 based on the Dallara Super Formula SF23. Compared to the IAC platform, the SF23 features higher downforce and more performant tires, resulting in significantly greater lateral and longitudinal acceleration capabilities, albeit with a lower top speed of approximately 250 km/h; performance figures are second only to Formula 1 cars. The A2RL format aims to replicate traditional human racing dynamics more closely. The inaugural season featured four car races, while the second season expanded to a simulation championship followed by six simultaneous agents on track. Unlike the structured passing rounds of the IAC, this competition emphasizes "free racing" where overtaking maneuvers, and multi-agent interactions occur organically over the course of a race. This unstructured environment places a premium on strategic decision-making and the ability to handle complex track layouts.

1.2.3 Roboracer

In parallel to these major leagues, the framework presented in this thesis has been adapted for the Roboracer[3] platform. While this environment provided valuable data for validating the modularity of the decision-making architecture, the specific results are considered preliminary and will be the subject of future publications.

1.2.4 Competition Rules and Constraints

A critical aspect of planning in autonomous racing environments is strict adherence to sporting regulations. While specific rules vary across competitions, both the Indy Autonomous Challenge (IAC) and the Abu Dhabi Autonomous Racing League (A2RL) are grounded in concepts inherited from human motorsport, such as *right of way*, limitations on defensive maneuvers, and clearly defined responsibilities during overtaking.

Since the latest experimental validation presented in this thesis was conducted within the A2RL framework, this section focuses on those regulatory constraints that directly influence local trajectory planning. These rules introduce explicit geometric and logical conditions that determine how interaction responsibilities evolve during wheel-to-wheel racing.

Defender obligations The concept of *right of way* governs which vehicle is entitled to occupy the optimal racing line at any given time. In general, the leading vehicle may choose its trajectory freely, but is subject to strict limitations when defending its position:

- **One-move rule:** The defending vehicle is permitted to make only a single change of direction to defend a position.
- **No weaving:** Repeated changes of direction on a straight, intended to disrupt the attacker's slipstream or trajectory, are prohibited.
- **No reactive blocking:** Defensive maneuvers must be proactive; reacting to the attacker's movement to block its path is not allowed.

[2] https://a2rl.io/

[3] https://roboracer.ai/

Right of way and yielding Racing interactions are characterized by a gradual transfer of responsibility from the attacking vehicle to the defending one. Initially, the attacker is fully responsible for avoiding collisions and must choose an appropriate location and timing to engage. This responsibility changes once specific geometric conditions are met:

- **Longitudinal proximity:** The attacker gains effective right of way when its front is within $15\,m$ of the defender's rear and it is laterally committed to one side. An attacker aligned directly behind the defender does not have right of way.
- **Yielding behavior:** Once right of way is established, the defender must yield space by constraining its trajectory, rather than by braking.
- **Leaving space:** The defender is required to maintain a lateral safety distance of at least $3.5\,m$ from the track boundary on the side where the attacker has gained right of way.

Overtaking engagement The regulations further define precise geometric thresholds that characterize the different phases of an overtaking maneuver:

- **Engagement point:** An overtaking attempt is formally recognized when the attacking vehicle's front wing becomes adjacent to the defending vehicle's rear wheels. Prior to this point, collision avoidance remains the sole responsibility of the attacker.
- **Shared responsibility:** Once significant overlap is established, both vehicles share responsibility for avoiding contact.
- **Lateral safety margin:** Throughout the engagement, a minimum lateral distance must be maintained to allow for safe maneuvering.
- **Crowding and track limits:** Forcing an opponent off the track is strictly forbidden. If overlap exists, the defender must leave at least one car width plus an additional safety margin between its vehicle and the track edge.

Collectively, these rules transform the planning problem from pure geometric obstacle avoidance into a constrained decision-making task, in which the set of admissible trajectories dynamically depends on the relative state of surrounding agents and on the current interaction phase.

1.3 Planning in Racing: From Global to Local

In the specific domain of autonomous racing, the hierarchical planning paradigm introduced in Section 1.1 takes on a specialized form. The primary objective shifts from general "safety and compliance" to the minimization of lap time, subject to the constraints of vehicle dynamics and track boundaries.

1.3.1 Global Planning: The Optimal Raceline

The foundation of any racing stack is the **Global Planner**. This module operates offline to compute the theoretical optimal trajectory, commonly referred to as the "raceline". The problem is typically formulated as an Optimal Control Problem (OCP) that

minimizes the time T required to complete a lap:

$$J = \min \int_0^L \frac{1}{v_x(s)} \, ds \tag{1.1}$$

where s is the curvilinear abscissa along the track and v_x is the longitudinal velocity.

This optimization considers the static limits of the vehicle (e.g., the friction circle, engine power, and aerodynamic drag) and the geometric constraints of the track boundaries (left and right margins). As detailed in Raji et al., 2022, advanced global planners utilize high-fidelity vehicle models (dynamic bicycle or tricycle models) to generate a reference path and a velocity profile that exploit the full potential of the tires. This static "ideal" solution serves as the reference for the online system.

1.3.2 Local Planning: Handling Interactions

While the global planner provides an optimal racing line for a solo run, it is inherently unaware of both the presence of other agents and short-term deviations required during execution. Even in the absence of interactions, strict adherence to a precomputed raceline is often impractical due to disturbances, execution errors, or the need to perform line changes and smoothly merge back to the nominal trajectory. As a result, the **local planner** plays a critical role not only in multi-agent scenarios, but also in single-vehicle operation.

Operating in real time, typically at frequencies of 20 Hz or higher, the local planner is required to:

- **Reason over feasible motion alternatives**, accounting for track geometry, static and dynamic obstacles, and interaction constraints.
- **Select and execute** a locally optimal maneuver among these alternatives, such as following, defending, or overtaking.
- **Adapt the vehicle motion online**, generating dynamically feasible trajectories and adjusting the longitudinal profile in response to evolving conditions and uncertainties.

This thesis addresses the limitations of traditional local planning approaches by introducing a modular planning framework coupled with an Optimal Control Problem (OCP). The proposed framework explicitly reasons over feasible drivable regions, referred to as *tunnels*, and integrates strategic decision-making for interaction handling. In doing so, it bridges the gap between the static optimality of the global planner and the highly dynamic nature of real world autonomous racing.

1.4 Autonomous Software Stack

This thesis builds upon the autonomous racing software stack developed and employed by the UNIMORE Racing Team for the autonomous racing competitions. The author is an active member of the team and has contributed to the design, development, and experimental validation of several modules, with a particular focus on state estimation, motion planning and control.

1.4.1 UNIMORE Racing Team Overview

The UNIMORE Racing Team originated as a collaboration between the High Performance Real Time Laboratory (HiPeRT Lab) at the University of Modena and Reggio Emilia, the spin-off company HIPERT srl, the Technology Innovation Institute (TII), the University of Pisa, and ETH Zurich. During the initial phases of the Indy Autonomous Challenge, the team competed under the name *TII EuroRacing*.

The complete autonomous software stack developed during this period has been documented in the system paper Raji et al., 2024a, which describe the full architecture adopted for oval racing. After the first event the University of Modena and Reggio Emilia was the only remaining institutional partner; later another system paper Raji et al., 2024b has been published describing the evolution of the stack to handle also road-course circuits.

The team evolved into its current form as *UNIMORE Racing*, now fully led by the University of Modena and Reggio Emilia through HiPeRT Lab, while maintaining algorithmic collaborations with HIPERT srl. Although the current stack significantly differs from the er.autopilot architecture, a dedicated system paper describing the new *UR autopilot* is planned but not yet available at the time of writing.

1.4.2 Software Stack Overview

The autonomous software stack adopted by the UNIMORE Racing Team follows the classical *Sense–Plan–Act* paradigm, and is structured into four main subsystems: perception, localization, planning, and control. While the complete implementation details of the current stack are outside the scope of this thesis, this section provides a high-level overview to contextualize the role and requirements of the planning framework proposed in this work.

Perception and Localization

The perception and localization layers are responsible for providing a consistent and reliable estimate of both the ego vehicle state and the surrounding environment. The system processes data from a heterogeneous sensor suite, including LiDARs, radars, cameras, and inertial sensors, whose outputs are fused and filtered before being forwarded to downstream modules.

On road-course circuits, ego-state estimation primarily relies on SLAM-based localization techniques, which provide accurate and drift-limited pose estimates even in scenarios where GNSS information is unreliable or unavailable. GNSS measurements, when present, play a marginal role and are mainly used for redundancy, validation, or initialization. High-frequency inertial data are fused within state estimation pipelines to ensure smooth and low-latency estimates of vehicle motion.

Perception modules detect and track surrounding agents, providing estimates of their positions, velocities, and relative motion. These outputs are inherently affected by sensor noise, latency, and occasional misdetections. Importantly, from the perspective of the planning module, all perception and localization outputs are already the result of multiple processing stages and thus represent filtered, delayed, and potentially imperfect information about the true system state.

Planning

The planning module constitutes the core focus of this thesis and acts as the central interface between perception and control. It is responsible for transforming estimated states and predictions into dynamically feasible motion commands while accounting for interaction constraints, race regulations, and vehicle limits.

Planning is organized hierarchically:

- **Global planning:** Computed offline, this layer generates a nominal minimum-time racing line and an associated reference velocity profile, based on track geometry, vehicle dynamics, and friction limits. The global planner is agnostic to other agents and represents the optimal solution for a solo run.
- **Local planning:** Operating online at high frequency, this layer adapts the reference trajectory to the current racing context. It reasons over feasible motion alternatives, handles interactions with other vehicles, and decides when to follow, defend, or overtake. The local planner must explicitly account for uncertainties, perception delays, and regulatory constraints, while ensuring that the resulting motion remains trackable by the controller.

Because the planning module sits between estimation and actuation, it inherently accumulates the effects of upstream errors and delays while directly influencing downstream vehicle behavior. This makes planning particularly sensitive to imperfections in perception and localization, and motivates the need for robust decision-making and trajectory generation strategies. The novel contributions of this thesis specifically target the local planning layer and are detailed in the following chapters.

Control

The control layer is responsible for executing the planned trajectory with high fidelity while maintaining vehicle stability near the dynamic limits. The primary control strategy is based on Model Predictive Control (MPC), which solves a constrained optimization problem at high frequency (typically 100 Hz) using a dynamic vehicle model.

The controller tracks the trajectory produced by the planning module and computes steering, throttle, and braking commands that respect actuator limits and vehicle dynamics. A low-level actuation layer then converts these commands into signals for the drive-by-wire system. The effectiveness of the control layer has been extensively validated in both simulation and on-track experiments, allowing the planning module to rely on its robustness when generating aggressive yet dynamically feasible trajectories.

1.5 Contributions and Outline

The primary objective of this thesis is to bridge the gap between theoretical motion planning algorithms and their practical deployment in high-speed, safety-critical multi-agent scenarios. While existing literature offers numerous solutions for trajectory optimization, few have been validated on full-scale vehicles operating at the limit of handling in adversarial environments.

1.5.1 Main Contributions

The specific contributions of this work to the state of the art in autonomous racing and motion planning can be summarized as follows:

- **The "Tunnels" Framework:** A novel, modular approach to local planning that decouples the definition of drivable areas from the trajectory generation. By dynamically shaping the convex constraints (tunnels) based on the surrounding environment, this method simplifies the optimization problem for the Model Predictive Control (MPC), avoiding local minima and ensuring solvability even in complex scenarios.
- **Hierarchical Decision-Making Architecture:** The design of a behavioral layer that integrates high-level strategic reasoning (e.g., when to overtake, defend, or yield) with low-level geometric constraints. This includes the development of different logics that translate continuous sensor data into discrete semantic states for rule compliance.
- **Multi-Platform Real-World Validation:** Unlike many studies limited to simulation or scaled platforms (e.g., F1Tenth), the proposed framework has been extensively validated on two distinct full-scale racing platforms: the Dallara IAC AV-24 (Indy Autonomous Challenge) on oval tracks and the Super Formula EAV 25 (A2RL) on complex road courses.
- **Adaptability to Diverse Racing Domains:** The demonstration of the framework's versatility, proving its effectiveness in both structured "passing battles" over 290 km/h (typical of oval racing) and unstructured "free racing" scenarios (typical of road courses), managing speeds exceeding 250 km/h and dynamic multi-agent interactions.

1.5.2 Thesis Outline

The remainder of this thesis is structured as follows:

- **Chapter 2** reviews the related work in the field of motion planning for autonomous driving and racing, analyzing the limitations of current approaches and positioning the proposed solution within the literature. Moreover, it provides important technical background useful for understanding the subsequent chapters.
- **Chapter 3** presents the theoretical formulation of the *Tunnels Framework*. It details the algorithms for drivable area generation, the *egoLoc* logic for situational awareness, the decision making process and the integration with the Model Predictive Control.
- **Chapter 4** explores the strategic trade-offs inherent in multi-agent racing, justifying the specific design choices regarding forecasting and geometric constraints. It analyzes the balance between trust in opponent predictions and risk management, detailing how to tune the system's aggressiveness.
- **Chapter 5** provides a comprehensive analysis of the experimental results. It compares performance data from high-fidelity simulations with telemetry recorded during the A2RL and IAC competitions, highlighting the system's behavior in critical scenarios such as overtaking and collision avoidance.

- **Chapter 6** concludes the dissertation, summarizing the key findings and discussing limitations and future research directions, including potential applications to non-racing autonomous driving contexts.

1.5.3 List of Publications

The research related to this dissertation has led to several peer-reviewed publications. The core contributions of the proposed planning framework, together with extensive experimental validation of its integration within a complete autonomous racing stack, have been disseminated in the following works, which are partially incorporated into this text. In particular, these publications report not only on high-level decision-making and trajectory planning, but also on the achieved control performance, including accurate trajectory tracking near the dynamic limits, as well as on the robustness of the underlying state estimation and localization pipelines in real-world racing conditions.

Core Publications (Author / Co-Author) The primary scientific contributions of this dissertation are directly reflected in the following publications, in which the author is either the first author or a co-author. These works cover decision-making and planning, state estimation, and the design and validation of complete autonomous racing software stacks:

- **A. Toschi**, F. Prignoli, and M. Bertogna, "Modular Decision-Making and Drivable Areas for Multi-Agent Autonomous Racing," in *Proceedings of the 2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2025.
- **A. Toschi**, N. Musiu, F. Gatti, A. Raji, F. Amerotti, M. Verucchi, and M. Bertogna, "Guess the Drift with LOP-UKF: LiDAR Odometry and Pacejka Model for Real-Time Racecar Sideslip Estimation," in *Proceedings of the 2024 IEEE Intelligent Vehicles Symposium (IV)*, Jeju Island, Korea, 2024.
- **A. Toschi**, *Integration of Model Predictive Control for Autonomous Racing Cars*, Master's thesis, University of Modena and Reggio Emilia, Modena, Italy, 2023.
- F. Prignoli, **A. Toschi**, N. Musiu, P. Falcone, and M. Bertogna, "Bridging Predictive Optimization and Real-World Racing: Motion Planning and Control for Head-to-Head Autonomous Racing," *Submitted to IEEE Transactions on Intelligent Vehicles*, 2026.
- A. Raji, D. Caporale, F. Gatti, **A. Toschi**, et al., "er.autopilot 1.0: The Full Autonomous Stack for Oval Racing at High Speeds," *Field Robotics*, vol. 4, no. 1, pp. 99–137, 2024.
- A. Raji, D. Caporale, F. Gatti, **A. Toschi**, et al., "er.autopilot 1.1: A Software Stack for Autonomous Racing on Oval and Road Course Tracks," *IEEE Transactions on Field Robotics*, vol. 1, pp. 332–359, 2024.
- A. Raji, A. Liniger, A. Giove, **A. Toschi**, et al., "Motion Planning and Control for Multi-Vehicle Autonomous Racing at High Speeds," in *Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2022.
- A. Raji, N. Musiu, **A. Toschi**, et al., "A Tricycle Model to Accurately Control an Autonomous Racecar with Locked Differential," in *Proceedings of the IEEE International Conference on Systems and Control (ICSC)*, 2023.

Related Team Publications This dissertation also builds upon the collective engineering effort of the UNIMORE Racing team and its research partners. The following publications and theses describe complementary work in vehicle modeling, control, state estimation, and simulation infrastructure that support the planning framework developed in this dissertation:

- N. Musiu, *Vehicle Dynamics and Friction-Adaptive Model Predictive Control in Autonomous Racing: A Comprehensive Modeling Approach*, Ph.D. dissertation, University of Modena and Reggio Emilia, 2025.
- F. Prignoli, *Predictive Motion Planning and Control for Autonomous Racing Under Regulations: From Compliance to Strategic Awareness*, Ph.D. dissertation, University of Modena and Reggio Emilia, 2026.
- A. Raji, *Model Predictive Planning and Control for Autonomous Racing, from HPC to Embedded Platforms*, Ph.D. dissertation, University of Parma, 2024.
- N. Musiu, E. Mascaro, A. Raji, et al., "A Comprehensive Benchmark of Vehicle Dynamics Models for Autonomous Racing: a Deep Dive into MPC," *Vehicle System Dynamics*, 2025.
- G. Lambertini, M. Pini, E. Mascaro, et al., "Fast and Realistic Automated Scenario Simulations and Reporting for an Autonomous Racing Stack," in *Proceedings of the 2025 IEEE Intelligent Vehicles Symposium (IV)*, forthcoming, 2025.
- A. Remonda, N. Hansen, A. Raji, N. Musiu, et al., "A Simulation Benchmark for Autonomous Racing with Large-Scale Human Data," *arXiv preprint*, 2024.
- S. Akouan'ha Ngoune, *Hybrid Kalman and Neuro-Enhanced Moving-Horizon Estimation of Vehicle Sideslip for High-Performance Autonomous Driving*, M.Sc./Thesis work, supervised by M. Bertogna, co-supervised by F. Amerotti, A. Toschi, and F. Hyso, 2025.

Chapter 2

Background and Related Work

This chapter provides the contextual foundation for the proposed planning framework. It first surveys the state-of-the-art in autonomous racing, highlighting the gap between simulation-based learning methods and the deterministic approaches currently deployed on track. Subsequently, it retraces the evolution of the UNIMORE Racing planning stack, analyzing the specific failures that motivated the shift towards the modular *Tunnels* architecture. Finally, it provides a comprehensive theoretical background on Model Predictive Control (MPC), Vehicle Dynamics, and Grip estimation, detailing the mathematical models that underpin the algorithms presented in Chapter 3 and better understand the result showed in Chapter 5.

2.1 Overview of Planning Strategies

Local trajectory planning has been widely explored in the literature, but in autonomous racing applications, particularly those tested on hardware Betz et al., 2022, the diversity of applied planning techniques is considerably narrower.

Neural network-based planners have demonstrated performance exceeding that of human experts in simulation Wurman et al., 2022, but their effectiveness in real-world racing scenarios remains largely unverified Chung, Seong, and Shim, 2024; Trumpp et al., 2024; Raji et al., 2022. In contrast, sampling-based Thrun et al., 2006; Jung et al., 2023 and graph-based Stahl et al., 2019; Lee et al., 2022 approaches, or hybrid combinations of both Ögretmen et al., 2022; Ögretmen* et al., 2023, remain the most commonly employed techniques for full-scale autonomous racing. These methods are well-established, computationally predictable, and effective in structured scenarios. However, they exhibit limitations: overtaking maneuvers are often restricted to straight sections of the track Na et al., 2024, or require additional physical constraints to be explicitly imposed Ögretmen et al., 2024, as the original sampling algorithm does not inherently account for them, leading to enforced limitations within the planner and necessitating further smoothing steps to fully exploit vehicle dynamics.

Model Predictive Control (MPC) is widely considered a strong candidate for racing due to its predictive capabilities and ability to incorporate dynamic constraints. Various MPC-based strategies have been explored, including combinations with game theory, both for full-scale Jung et al., 2021 and small-scale Liniger, 2018 autonomous cars, as well as approaches that exploit track layout characteristics Bhargav et al., 2021. Additionally, nonlinear MPC has been demonstrated to be effective (in simulation), even when integrating decision-making directly within the optimization process Buyval et al., 2017; Li et al., 2022. However, incorporating complex collision-avoidance mechanisms or dynamically changing track-bound constraints within MPC can make real-time performance challenging.

To mitigate these challenges, our framework follows a modular approach inspired by Liniger, Domahidi, and Morari, 2015 where drivable area computation is decoupled from trajectory optimization. However, in that work, opponent predictions are not considered, as it was tested on static small-scale vehicles. State-machine-based solutions Suresh Babu and Behl, 2022 can be effective when dealing with a single opponent but do not scale well in multi-agent settings, where the number of transitions increases significantly, making the system prone to errors. More recent works propose hierarchical planning architectures Jank, Rowold, and Lohmann, 2023, leveraging MPC for merging precomputed racing lines Ticozzi et al., 2023 or incorporating learned opponent behavior into planning decisions Baumann et al., 2025. However, none of these methods explicitly address how multiple dynamic interactions can be considered simultaneously, especially in real-time experiments.

This thesis addresses this gap by demonstrating the effectiveness of a modular framework that combines drivable area computation and high-level decision-making to handle multi-agent interactions dynamically, even at high speeds and accelerations, while ensuring compliance with externally imposed rules. Furthermore, the proposed architecture is designed to be flexible and extensible, providing a foundation for future advancements in autonomous racing planning Toschi, Prignoli, and Bertogna, 2025.

2.2 Team's Planning Evolution

The planning architecture presented in this thesis is the result of an iterative development process driven by the specific requirements of the IAC and the A2RL competitions. Previous iterations of the team's software stack Raji et al., 2024a highlighted critical limitations in handling multi-agent scenarios, which directly motivated the design of the modular *Tunnels Framework*.

2.2.1 Limitations of Frenet-based Planning

In the early stages of the IAC (2021-2022), the team employed a sampling-based planner operating in the Frenet frame (s, d) Raji et al., 2022. This approach generated a candidate set of polynomials to minimize deviation from the optimal global raceline while avoiding static obstacles. While effective for single-vehicle time trials or simple static avoidance, this method proved insufficient for head-to-head racing:

- **Feasibility Issues:** Converting Frenet polynomials back to Cartesian space does not guarantee dynamic feasibility. At high speeds, a geometrically valid polynomial might imply lateral accelerations exceeding the friction limits, leading to loss of control.
- **Reactive Nature:** The sampling approach is inherently reactive. It evaluates trajectories based on the current snapshot of the environment, lacking the predictive foresight required to set up complex overtaking maneuvers or to handle strategically the *Right of Way (RoW)* rules.
- **Implementation Complexity vs. Flexibility:** As noted in Section 2.1, sampling strategies can be augmented with additional dynamic checks or post-processing steps to ensure physical feasibility. However, such solutions are not "plug-and-play" and require significant engineering effort to tune for edge

cases. Consequently, the team chose to transition to an MPC-based architecture. This decision allowed us to better exploit the team's established expertise in vehicle dynamics and optimal control, providing a mathematically rigorous and inherently more flexible foundation for handling the complex constraints of multi-agent racing.

2.2.2 Scalability Issues with Finite State Machines

To introduce behavioral logic (e.g., Overtake vs. Follow), earlier versions integrated the planner with a Finite State Machine (FSM). The FSM would trigger specific parameter sets or target offsets based on the relative position of the opponent. However, as the complexity of the scenarios increased, particularly with the introduction of multi-agent races in A2RL, the FSM approach faced severe scalability issues:

- **State Explosion:** Handling interactions with 3 or 4 opponents required an exponential number of transitions to cover all possible permutations of relative positioning (e.g., "Car A inside, Car B outside, approaching turn").
- **Oscillations (Flickering):** FSMs are prone to rapid switching between states (e.g., Overtake $\leftrightarrow$ Abort) due to sensor noise or minor prediction updates, resulting in erratic vehicle behavior.

2.2.3 The Need for Convex Decomposition

Initial attempts to solve these issues using a single MPC formulation incorporating collision avoidance constraints proved computationally intractable for real-time applications. The non-convex nature of obstacle avoidance constraints creates local minima and requires heavy solving times. This evolution clarified the need for a hierarchical decomposition: a dedicated module to solve the non-convex geometry problem (finding the *Tunnel*) and a downstream convex solver (MPC) to handle the dynamics.

2.3 Theoretical Concepts

This section details the fundamental mathematical models and control strategies that underpin the implementation of the proposed framework. Specifically, it covers the Model Predictive Control (MPC) formulation utilized for trajectory tracking, the dynamic bicycle model describing vehicle motion, and the tire models essential for understanding the vehicle limits.

2.3.1 Model Predictive Control (MPC)

Model Predictive Control (MPC) has become the de facto standard for high-performance autonomous racing due to its ability to handle Multi-Input Multi-Output (MIMO) systems while explicitly respecting physical and operational constraints. Unlike classical control methods (e.g., PID or LQR), MPC solves an Optimal Control Problem (OCP) at each sampling instant t, computing a sequence of future control inputs to minimize a cost function over a finite prediction horizon N Toschi, 2023.

Optimal Control Problem Formulation

In the context of this work, the MPC is designed to track a reference trajectory (raceline) while maintaining the vehicle within the safe drivable area (generated *Tunnels*). The problem is formulated in discrete time as follows:

$$\min_{u_0,\dots,u_{N-1}} \quad J(x,u) = \sum_{k=0}^{N-1} l(x_k, u_k, \Delta u_k) + V_f(x_N) \tag{2.1a}$$

$$\text{s.t.} \quad x_{k+1} = f_d(x_k, u_k) \quad \forall k \in [0, N-1] \tag{2.1b}$$

$$u_{min} \leq u_k \leq u_{max} \quad \forall k \in [0, N-1] \tag{2.1c}$$

$$\Delta u_{min} \leq \Delta u_k \leq \Delta u_{max} \quad \forall k \in [0, N-1] \tag{2.1d}$$

$$h(x_k) \leq 0 \quad \forall k \in [1, N] \tag{2.1e}$$

where $x_k \in \mathbb{R}^{n_x}$ is the state vector, $u_k \in \mathbb{R}^{n_u}$ is the control input vector, and $\Delta u_k = u_k - u_{k-1}$ represents the rate of change of the inputs. The term $f_d(\cdot)$ denotes the discretized vehicle dynamics, while $h(x_k)$ encapsulates state constraints such as track boundaries.

Cost Function Design

The cost function J is typically quadratic, designed to balance tracking accuracy against control smoothness. It can be expanded as:

$$J = \sum_{k=0}^{N-1} \left(\|x_k - x_{ref,k}\|_Q^2 + \|u_k - u_{ref,k}\|_R^2 + \|\Delta u_k\|_{R_\Delta}^2 \right) + \|x_N - x_{ref,N}\|_P^2 \tag{2.2}$$

- **Tracking Term (Q):** Penalizes deviations from the reference path and velocity profile. In racing, high weights are assigned to lateral error (to stay on the racing line) and heading error.
- **Actuation Term (R):** Penalizes the magnitude of control inputs, though in racing applications this is often minimized to allow full exploitation of the vehicle's power and steering authority.
- **Regularization Term (R_Δ):** Crucial for high-speed stability, this term penalizes aggressive changes in control inputs (e.g., steering jerk), preventing high-frequency oscillations that could destabilize the vehicle dynamics.
- **Terminal Cost (P):** Approximates the infinite-horizon cost to ensure stability and convergence of the closed-loop system.

Real-Time Solving Strategy

To achieve the update rates required for racing (typically 100 Hz), the Non-Linear Programming (NLP) problem is often approximated. The team employs a Linear Time-Varying MPC (LTV-MPC) approach. At each step, the non linear vehicle dynamics $f(x,u)$ are linearized around the current reference trajectory, yielding a Quadratic Programming (QP) problem:

$$x_{k+1} = A_k x_k + B_k u_k + g_k \tag{2.3}$$

This QP is then solved using high-performance solvers such as High Performance Interior Point Method (HPIPM), which exploits the block-sparse structure of the optimal control problem to deliver solution times in the order of milliseconds.

2.3.2 Vehicle Dynamics Modeling

Accurate modeling of vehicle dynamics is a prerequisite for high-performance planning and control in autonomous racing. While kinematic models may suffice for low-speed urban driving, racing maneuvers involve large tire slip angles, significant lateral accelerations, and operation close to the physical limits of adhesion. Under these conditions, a dynamic formulation is required to correctly capture the coupling between longitudinal and lateral motion. A comprehensive analysis and comparison of vehicle models for autonomous racing applications is provided in Musiu et al., 2026.

Dynamic Bicycle Model

The proposed framework relies on a Dynamic Bicycle Model (also referred to as the single-track model), which aggregates the left and right wheels of each axle into a single equivalent wheel. This representation captures the dominant vehicle dynamics while maintaining a level of complexity compatible with real-time optimization.

The state vector is defined as

$$x = [X, Y, \psi, v_x, v_y, \dot{\psi}]^T,$$

where (X, Y) denotes the vehicle position in the inertial frame, ψ is the yaw angle, v_x and v_y are the longitudinal and lateral velocities expressed in the body frame, and $\dot{\psi}$ is the yaw rate.

The equations of motion are derived from the conservation of linear and angular momentum:

$$\dot{X} = v_x \cos\psi - v_y \sin\psi \tag{2.4a}$$

$$\dot{Y} = v_x \sin\psi + v_y \cos\psi \tag{2.4b}$$

$$\dot{v}_x = \frac{1}{m}(F_{x,r} + F_{x,f}\cos\delta - F_{y,f}\sin\delta) + \dot{\psi} v_y \tag{2.4c}$$

$$\dot{v}_y = \frac{1}{m}(F_{y,r} + F_{y,f}\cos\delta + F_{x,f}\sin\delta) - \dot{\psi} v_x \tag{2.4d}$$

$$\ddot{\psi} = \frac{1}{I_z}\left(l_f(F_{y,f}\cos\delta + F_{x,f}\sin\delta) - l_r F_{y,r}\right) \tag{2.4e}$$

where m is the vehicle mass, I_z is the yaw moment of inertia, l_f and l_r are the distances from the center of gravity to the front and rear axles, δ is the front steering angle, and $F_{x,i}$, $F_{y,i}$ are the longitudinal and lateral tire forces acting on axle $i \in \{f, r\}$.

Tire Forces and Slip Modeling

In racing conditions, tire forces constitute the primary limiting factor of vehicle performance. Lateral forces arise when the tire velocity vector is not aligned with the wheel heading, a phenomenon quantified by the slip angles:

$$\alpha_f = \delta - \arctan\left(\frac{v_y + l_f\dot{\psi}}{v_x}\right), \quad \alpha_r = -\arctan\left(\frac{v_y - l_r\dot{\psi}}{v_x}\right). \tag{2.5}$$

Similarly, longitudinal forces depend on the slip ratio κ, which captures the mismatch between wheel rotational speed and ground speed.

To model the nonlinear relationship between slip and force, the framework adopts a Pacejka Magic Formula tire model. In the case of pure lateral slip α (front or rear), the lateral force is expressed as:

$$F_y(\alpha) = \mu F_z \cdot D \sin\left(C \arctan\left(B\alpha - E(B\alpha - \arctan(B\alpha))\right)\right), \tag{2.6}$$

where F_z is the vertical load and the coefficients B, C, D, and E define the stiffness, shape, peak, and curvature of the force–slip relationship.

Friction Limits and Dynamic Feasibility

The total force that a tire can generate is bounded by the available road–tire friction, commonly represented by the friction circle (or ellipse):

$$\left(\frac{F_x}{F_{x,\max}}\right)^2 + \left(\frac{F_y}{F_{y,\max}}\right)^2 \leq \mu^2. \tag{2.7}$$

Respecting this constraint is essential to prevent loss of control. In the context of planning and control, the friction limit directly constrains the admissible longitudinal and lateral accelerations and therefore defines the dynamically feasible region of motion.

2.3.3 Tire Grip and State Estimation

While the vehicle and tire models described above define the physical limits of motion, their practical effectiveness strongly depends on the accuracy of the underlying parameters and state estimates. In real racing scenarios, the available grip and several vehicle states cannot be assumed constant or directly measurable.

Need for Online Grip Adaptation

The friction coefficient μ is not constant over time. Tire temperature, wear, rubber deposition, and track evolution cause significant variations even within a single race stint. As shown in Chapter 6 of Musiu, 2026, relying on a fixed friction estimate leads either to overly conservative behavior or to loss of control when operating near the limits.

For this reason, the autonomous racing stack incorporates online mechanisms to adapt the effective friction limits used by planning and control. These mechanisms observe discrepancies between the predicted vehicle response and the measured behavior, allowing the system to scale force and acceleration constraints in real time.

Role of Lateral Velocity and Sideslip Estimation

Accurate estimation of lateral vehicle states is a key enabler for both grip adaptation and stable control. In particular, lateral velocity v_y and sideslip angle α are critical variables when operating close to the handling limits. Since these quantities are not directly measured by onboard sensors, they must be estimated online.

As demonstrated in Toschi et al., 2024, robust estimation of lateral velocity using LiDAR odometry combined with tire model predictions significantly improves the accuracy of sideslip estimation under aggressive driving conditions. Moreover, now

also information from the radar and the Kistler are integrated in the same filter, further increasing the robustness. This information enhances the reliability of both the vehicle model and the grip estimation process.

Impact on Planning and Control

State estimation errors directly affect the accuracy of predicted vehicle behavior and, consequently, the feasibility of planned trajectories. Since the planning module operates as an interface between perception and control, it inherently accumulates estimation errors and delays from upstream modules.

Reliable estimation of grip and lateral states therefore plays a fundamental role not only for low-level control but also for high-level planning. By enabling tighter yet safe constraints, accurate state estimation allows the planner to exploit the available grip more effectively, improving performance while maintaining robustness near the dynamic limits.

Chapter 3

Planning Framework

3.1 Architecture Overview

The proposed planning workflow is illustrated in Fig.3.1 and follows a classical *Perceive-Plan-Act* structure. The *Perception* stage provides environment constraints, detected opponents and their predicted behavior, which are then processed in the *Planning* stage to generate a feasible trajectory. Finally, the *Actuation* stage executes the selected trajectory via the vehicle control system. At the core of the framework lies the *Tunnel Framework*, which computes the drivable areas and selects the best option when multiple choices exist. These *Tunnels* are regions in Frenet coordinates, derived by deforming the given *Environment Bounds*. The Frenet frame is especially useful because it separates longitudinal progress from lateral position, allowing track boundaries to be directly expressed as state constraints in the MPC. The framework operates based on *Parameters* categorized into three main groups:

- Overtaking and obstacle avoidance: tunnel size, safety margins relative to vehicles and environment bounds, and overtaking aggressiveness.
- Being overtaken: space allocation for other agents and thresholds to recognize yielding scenarios.
- External signals: behavior directives from the autonomous stack, such as overtaking permissions (e.g., green flags in racing contexts) or Adaptive Cruise Control (ACC) following distances.

The *Tracker* module assigns unique identifiers to detected opponents, ensuring consistency across consecutive detections, while the *Forecasting* module predicts opponent trajectories. It is important to highlight that predicted behaviors are influenced by opponent interactions via a dedicated planner for each agent. This ensures that forecasting takes into account mutual collision avoidance between opponents, rather than treating them as independent moving obstacles. More details on the forecasting module will be provided in the section 4.1. The *MPC Instances* operate both as an input and an output: they use tunnel margins as constraints while also leveraging the prediction horizon to refine ego-vehicle trajectory forecasting. The proposed framework has been validated with both single and multiple MPC instances, as detailed in section 3.5 and demonstrated in Chapter 5.

3.1.1 Tunnels Framework

The framework consists of four main blocks, as illustrated in Fig.3.2, each representing a key stage of the planning pipeline:

1. *Interactions Manager*: Identifies opponents interactions and determines track occupancy constraints.

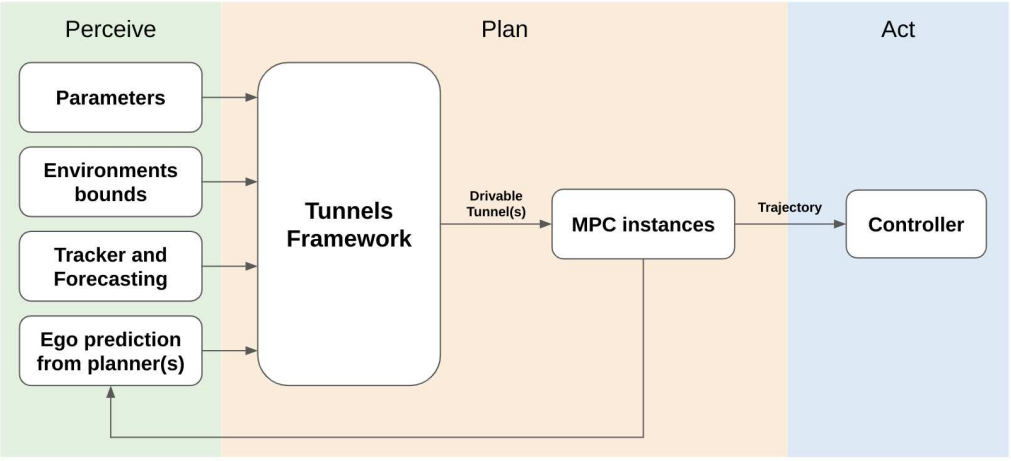


FIGURE 3.1: Simplified block scheme of planning modules inside the autonomous stack.

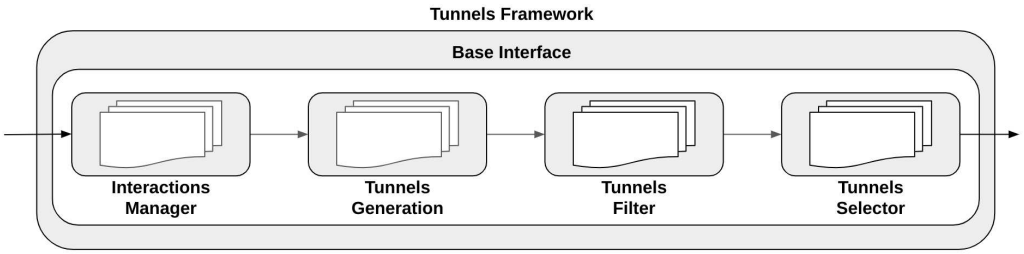


FIGURE 3.2: Tunnels framework: pipeline divided into four blocks.

2. *Tunnels Generator*: constructs potential drivable corridors based on opponents interactions.
3. *Tunnels Filter*: validates and refines the generated tunnels, enforcing feasibility constraints.
4. *Tunnels Selector*: selects the optimal tunnel(s) for trajectory optimization, using single or parallel selection strategies.

These blocks are implemented as C++ virtual classes, ensuring a standardized interface while allowing flexible customization. New implementations can be seamlessly integrated, and thanks to a modular design, users can configure the system at runtime without recompilation. The subsequent subsections provide a detailed explanation of each component.

3.2 Interactions Manager

The *Interactions Manager* is the first block in the pipeline. First, it interpolates track boundaries, wall boundaries, the ego-vehicle prediction, and opponent predictions over a shared time horizon. Being in Frenet coordinates, the track and wall boundaries are represented as lateral offsets from the raceline at each longitudinal step, so they give the information about how much lateral space is available at each point along the track and how far is it from the nominal raceline. The ego vehicle prediction consists of a sequence of Frenet states (longitudinal position, lateral offset, and longitudinal velocity) over the prediction horizon, representing the expected trajectory of the ego vehicle created by the open loop MPP with the speed profile created

by the longitudinal planner (that ignores the opponents and always overtake). Each opponent's prediction is similarly represented as a sequence of Frenet states over the same horizon but the predictions are created by a dedicated module that we will refer as Forecasting and whose details will be given in the next Chapter.

Then, a map of opponents identifiers is updated based on the latest input data. It updates existing opponents and adds new ones or removes them as necessary, ensuring that each opponent retains a consistent identifier across time steps. This step is important to maintain consistent tracking of opponents across consecutive detections so that info and flags assigned to each opponent can be preserved over time. Moreover, in this way it is possible to handle the flag through hysteresis logic, avoiding rapid change of behavior that could be induced by the different opponent classification.

Among these info, the following are computed for each opponent:

- ***Overtake Margins - OTMs***: lateral and longitudinal safety distances are calculated to be additional buffer zones over vehicle dimensions to ensure safer overtaking and account for eventual imprecisions and delays: detection a bit shifted or delayed, wrong predictions and controller tracking errors. These margins are parameterized, with a minimum and maximum value for each side, and are linearly scaled based on the ego-vehicle speed. Higher speeds lead to larger margins, enhancing the possibility to get closer in the turns and approach while braking and avoiding staying too close in the straights of fast turns when unnecessary.

- ***Attacker-Defender***: this flag determines whether an opponent is overtaking or being overtaken. The front and back *Overtake Margins* discriminate three regions as depicted in Fig.3.3. An agent in the front area is always considered a defender, and one in the back always an attacker; a positional hysteresis define the flag inside the longitudinal margins. It's unlikely to happen that a vehicle is spawning inside the hysteresis zone (detection should see far enough to avoid this), but in this case the opponent is initialized as an attacker by default.

- ***Right of ROW flag***: determines whether the ego-vehicle has priority or should yield to specific opponents. This is based on the longitudinal ROW distance and the commitment to one side, as per racing rules explained in 1.2.4. Being sure of the opponent intention to overtake on a specific side is not trivial, in this case it's checked whether the opponent's lateral position is offset enough to have its left tires on the right of ego-vehicle right tires (for overtaking on the right) or vice-versa for overtaking on the left. If the opponent is inside the hysteresis zone, no ROW is assigned.

- ***Granted Borders***: the first time that an attacker opponent gains its ROW, the current lateral distance of the ego vehicle from the border in which the opponent is overtaking is stored as a lateral limit. The granted left or right border is the maximum between that stored value and the current ego-vehicle distance from the specific track border. In this way, the initial distance is always the minimum border even if the controller is overshooting while the border is even increasing if the vehicle is already getting further to the other side; this is used as non blocking strategy as explained in 3.3. This border is updated only when the opponent is no longer classified as an attacker.

- ***Heading error***: as the yaw of the vehicles is not directly available from the Frenet representation, the heading of the agent should be used to compensate

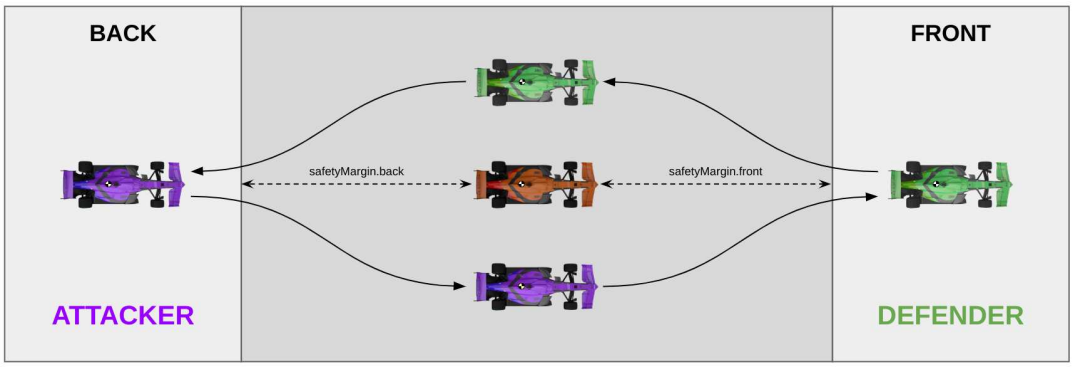


FIGURE 3.3: Attacker-Defender flag: hysteresis-based classification of opponent status.

the vehicle dimension using its orientation. Actually, this is not implemented yet, but it will be added in the future. Not using it we are assuming that each agent is always following a line that is parallel to our raceline, that is true in most of the cases but not always, especially in the corners entry and exit. This is even more crucial when a vehicle is spinning or going sideways, because in this case the bounding box used for collision checking is very different of the real occupied space. The *Heading error* flag was used to compensate this effect, using the car length also in lateral direction when the heading error (current heading compared to the one of our raceline) is high. This feature is available and was tested in simulation, but not yet implemented in the real car because the perception module does not provide the heading of the opponents yet.

- ***Identifier***: unique ID assigned to each opponent from the tracker module.
- ***Previous Index***: saved position in the internal data structure in the previous iteration for logics that need consecutive iterations.

3.2.1 Ego Vehicle Relative Positions

Once these opponent-specific parameters are computed, this class calculate the ego-vehicle's relative position with respect to each opponent; these will be referred to as *EgoLocs*. Each *EgoLoc* consists of six boolean flags indicating whether the ego-vehicle is:

- *Left*: opponent is on the right
- *Right*: opponent is on the left
- *Front*: opponent is behind
- *Back*: opponent is ahead
- *FrontCoG*: opponent CoG is behind ego CoG
- *Critical*: opponent is too close

The importance of these flags for the decision making part will be clarified in sections 3.4-3.5 but how these are calculated is showed here because is where they are computed, in the first step pipeline of the tunnels framework as they are used by all the classes.

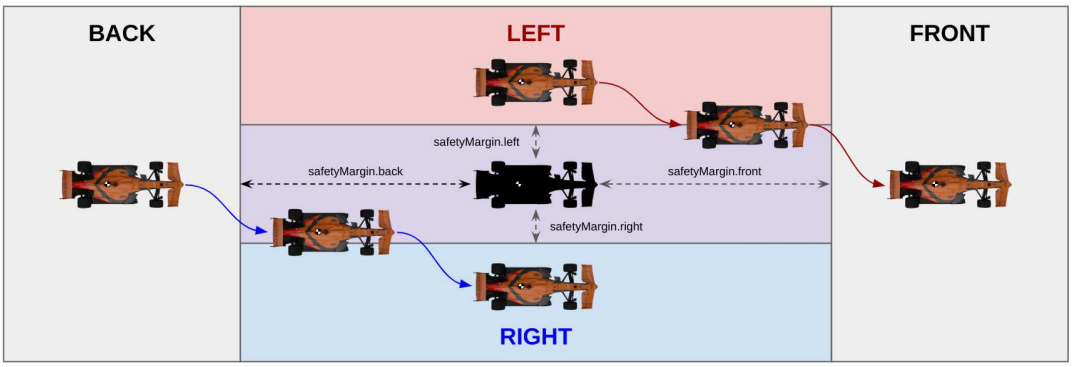


FIGURE 3.4: EgoLoc first implementation: structure of four flags indicating ego vehicle (orange) relative position to opponents (black).

In the first implementation, the *Overtake margins* defined four discrete regions Fig.3.4 outside the margins, where the purple zone inside instead was handled through an hysteresis to decide if the vehicle should be considered on the left or on the right. It is important to observe that the controller should adhere to the tunnel boundaries (that are computed using the same *Overtake Margins*) to avoid accessing the purple zone.

However, as the MPC is always tracking a reference trajectory when this happens to be inside the tunnel, the optimization will stay as close as possible to the tunnel limits. Due to this, during testing it was observed that the controller could slightly overshoot the tunnel limits that for him are just soft constraints to stay close as possible to the raceline or to prioritize vehicle stability; especially at high speeds or during aggressive maneuvers, due to tracking errors (such as an overshoot on the cruise control) or the other agent moving toward the ego vehicle (controller doesn't exit from the tunnel but may find itself out), this become more relevant. This enlighten a limitation of this approach, as the ego-vehicle could be easily classified either left or right when it is close to the center of the purple zone, leading to an erroneous influence to the decision making progress. To address this, the *EgoLoc* logic was modified to dynamic cones instead of the previous squared regions, as shown in Fig.3.5. Four lines are defined originating from the opponent's CoG, with slopes determined by the maximum and minimum relative speeds between the ego-vehicle and the opponent; forming in Frenet four cones-like regions that adapt over time. These boundaries are calculating using the standard linear equation of a line passing through the origin $y = mx$ [1]. In this context, these coordinates translate to $LateralDistance = LongitudinalDistance \times DynamicSlope$, therefore, the boundary lines n_{left} and n_{right} are calculated as:

$$n(\Delta s) = \Delta s \times slopeparameter(\Delta Vx) \tag{3.1}$$

In this way, the further we are from the opponent, the wider the cones become, the less constrained we are from being classified left or right. On the other side, the dynamic slopes ensure that when the delta speed is high, the cone in the back become narrower, making it more probable to be classified as left or right earlier and avoiding a change of direction that could lead to unsafe situations with high relative speeds. On the front, the cone become wider with high relative speed because in this case the ego-vehicle can return to the center without risking blocking

[1] These lines are straight in Frenet coordinates but when reported in Cartesian they will be more curved when the raceline will be more curved (that is a straight line in n = 0 in Frenet)

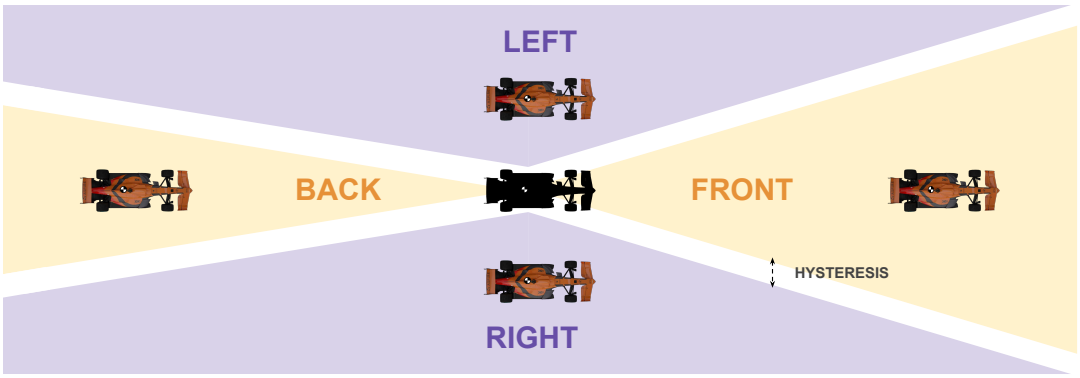


FIGURE 3.5: EgoLoc latest implementation: four lines divide the region in four cones that vary dynamically. The white zone in between cones is not defined as an hysteresis buffer; it is filled with the previous cone.

the other. For the same reasons on the contrary, when the relative speed is low the back cone become wider and the front one narrower; giving more freedom in choosing the overtake side as there is more time to react and less risk of collision in the front when concluding an overtake. This approach provides a more continuous and accurate representation of the ego-vehicle's relative position, reducing misclassifications when near the center zone and avoid possible flickering of the flags (that could happen in the previous approach when the two agents were on the longitudinal hysteresis limits). The vehicle is considered to be in the **Front** or **Rear** cone if its lateral position Δn at CoG is strictly between the two boundary lines.

$$n_{right} \leq \Delta n \leq n_{left} \tag{3.2}$$

To be considered in a **Side** cone, the check is stricter than simply crossing the line. The Ego vehicle is not considered "Left" or "Right" until its Center of Gravity (CoG) has crossed the boundary line **plus** half the vehicle width. This work as buffer term H (Hysteresis) to prevent flickering. Inside H you are not specifically inside a cone but it's filled with the cone you were in the previous iteration. This effectively means the vehicle must commit to the side zone with the whole car outside the boundary (at CoG) before the state changes, preventing rapid state oscillation near the boundary.

$$(\Delta n \leq n_{right} - H) \quad \vee \quad (\Delta n \geq n_{left} + H) \tag{3.3}$$

In both *EgoLocs* implementations, the *Critical* flag is set when an opponent is inside a ego-vehicle critical bubble. The latter size is defined using the *critical margins* that, similarly to the *OTMs*, are a group of four values (back/front/left/right) but here they are fixed and referred to the ego-vehicle. When an opponent is within one of these distances, the *Critical* flag is activated. This proximity indicates a potentially hazardous proximity that may require immediate attention or evasive action; the effects of this flag on the drivable areas will be explained in the next section.

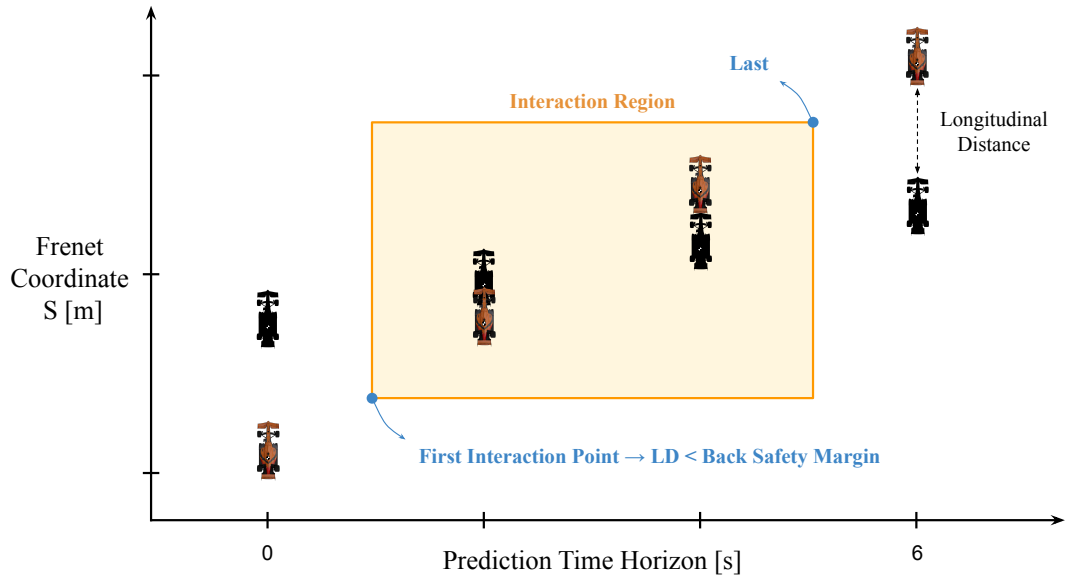


FIGURE 3.6: Interaction region formed by the overlap of ego and opponent predictions. Each interaction point is a boolean flag indicating longitudinal proximity at each time step, while ignoring lateral positions.

3.2.2 Interactions calculation

This is the part the gives the name to the class, where the core input to the next block is calculated: the *Interactions* between ego and each opponent over the prediction horizon. These interaction points will determine where tunnels deviate from the nominal environment bounds Fig. 3.6. For each detected agent, the longitudinal ego prediction is compared with the longitudinal opponent prediction over the entire horizon. As touched upon in the previous section, the longitudinal ego prediction used is the one with the speed profile created by the longitudinal planner, that is always overtaking; in other words, it calculates what the ego would do ignoring the opponents. This doesn't correspond to what the MPP and MPC are really doing, but is a conservative approximation that works because the real longitudinal behavior will always be (equal or) more restrictive due to many factors such as: the curvature of the overtaking maneuvers, the need to follow if ACC is active, the need to slow down for handling the car, etc. So in this way we are considering interactions before the ones that would result if we are using the MPP prediction, so the controller will actually have more time to react. Moreover, using this speed profile we are gaining also on the stability side. It has been tested to use at least the MPP curvature instead of the racing line one, to have this speed profile closer to the real one while deviating from the racing line, but the results were not significantly different in the nominal cases while even worse in the other cases: using the MPP curvature of the previous iteration is inherently influenced by the previous tunnels choice and to what is happening in the closed loop controller; so in case of change of decision or when the MPP is covering after recovering from an oversteer or a wrong behavior, the curvature could vary significantly, leading to flickering or unstable interactions detection. Moreover, using the ideal speed profile is more fair for the tunnels calculation because every choice has the same, when using the curvature of the previous choice we are influencing the interactions - so the tunnels shape - also of the other side(s). Regarding the opponents predictions, these are the ones created by the Forecasting

module explained in 4.1, and the same longitudinal planner (another instance) is used to calculate their speed profile but is not an "always overtaking" profile but follow what the *Forecasting* module decided for them. In each step of the horizon, the longitudinal distance between ego and opponent is calculated. If the opponent is in front and this delta distance is less than a car length plus the *back OTM*, or if the opponent is behind and the delta distance is less than a car length plus the *front OTM*, then an overlap is detected. This indicates that the two vehicles are close enough to potentially interact, either by overtaking or being overtaken. Once all these interaction points are identified over the horizon for all the opponents, these are re ordered based on who is interacting first in time. This is necessary because then the number of opponents are clamped to a maximum number (e.g. 6) for computational reasons, so we want to keep the most relevant ones; moreover having them re ordered make the pipeline potentially more efficient for the checks explained in 3.4.

At this stage, only the *track tunnel* (with also the wall bounds), an interpolated segment of (eventually) tightened track bounds, is computed. The interpolated ego and opponent predictions, the interpolated track and wall boundaries, the *egoLocs*, the interaction vectors and all the opponent related flags are passed to the subsequent module.

Section Summary

The **Interactions Manager** transforms raw predictions into a semantic description of the racing scenario. It computes velocity-scaled safety margins, assigns attacker/defender and right-of-way roles with hysteresis, and introduces memory mechanisms to prevent blocking behavior. A key contribution is the **dynamic cone-based EgoLoc**, which stabilizes relative positioning under aggressive maneuvers. Finally, interactions are detected using a conservative open-loop ego prediction to ensure early and robust constraint generation for subsequent planning stages.

3.3 Tunnels Generator

The `Tunnels Generator` module processes the interpolated inputs and interaction points provided by the previous stage of the pipeline. Its primary function is to construct all possible drivable corridors, referred to as *tunnels*, by recursively reshaping the environment boundaries in response to surrounding opponents. Before initiating the generation process, the module determines the active boundaries of the drivable area by assigning a specific `environment` variable. This can be defined in two ways: the **Nominal Scenario**, where boundaries are strictly the "track" edges (white lines with an eventual safety margin), or the **Emergency Scenario**, where boundaries expand to the *walls*. The latter are not strictly walls but run-off areas that correspond to paved regions such as curbs and extended asphalt zones outside the nominal track limits, where the vehicle can physically drive without encountering grass, gravel, or unsafe surfaces. These areas do not represent nominal racing lines but provide additional maneuvering space in non-standard situations.

The switch to the escape scenario is triggered when the ego vehicle enters a critical condition. This is determined by specific criteria:

1. the *egoLoc* is set to "critical" for any opponent (as described in 3.2)
2. the car is physically outside the track boundaries but within the walls
3. mayday request from the controller: in case the MPC would need more space (thus more freedom) in complicated cases i.e. handling is compromised because of high longitudinal or lateral slips (implemented just on the planning side, the controller is never requesting this yet)

When run-off areas are active, the system not only switch the boundaries but also adapts the overtake margins to prioritize maneuverability. In particular, overtaking margins switch from dynamically computed values to *critical margins*, which represent the smallest admissible thresholds. Although dynamic margins normally vary between minimum and maximum values, critical margins are always smaller, allowing the ego vehicle to accept closer proximity to opponents in constrained or hazardous situations. At the same time, even if in particular sections of the circuit the wall and the track boundaries coincide (no runoff areas available) the safety offsets from the white lines—are removed, enabling the planner to exploit the full available surface up to the physical limits when necessary [2].

Once the environment configuration is defined, tunnel generation proceeds through a recursive process that is the core of this module. The recursion starts from a single *base tunnel*, representing the fully interpolated environment, and iterates through one opponent at a time to reshape the boundaries based on the type of interaction. For each detected opponent, the current set of tunnels is duplicated resulting in two variants per tunnel. Consequently, for N opponents, a total of 2^N tunnels are generated. Among them, two represent extreme strategies—fully leftmost and fully rightmost manoeuvres—while the remaining $2^N - 2$ are *mixed tunnels*, combining overtakes on different sides depending on opponent ordering (Fig. 3.7 – Algorithm 1).

The complexity grows exponentially, and at the moment up to 8-10 opponents can be handled maintain the planner's intended frequency (details in 5.3). The

[2] In racing scenarios, these additional safety margins are typically set to zero by default. Typically, these are used when you are not certain about track bounds. Instead, maintaining a fixed distance from the borders during the race can drastically remove the available area

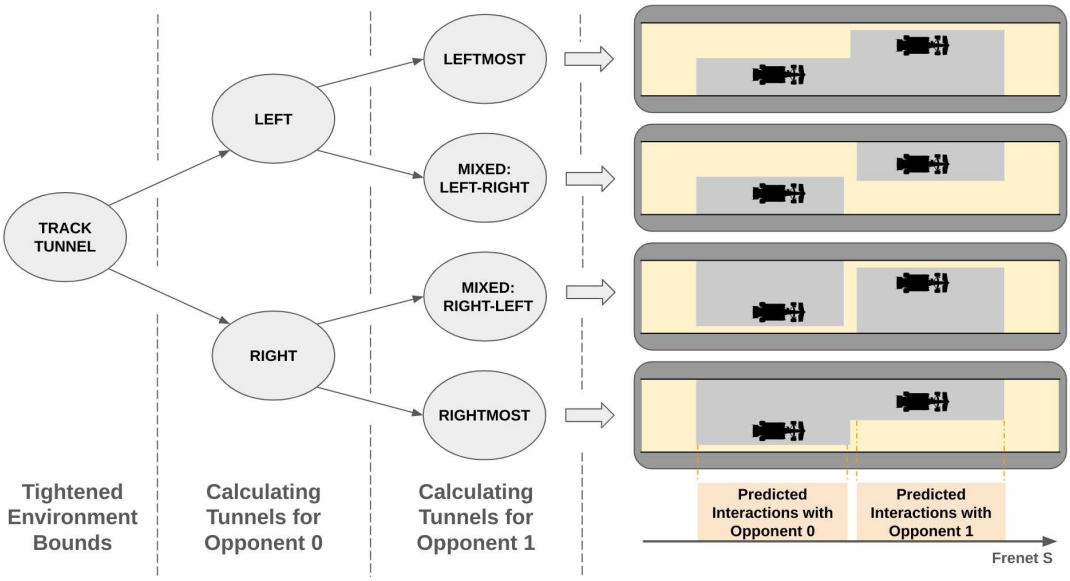


FIGURE 3.7: Recursive tunnel generation: an example with two static opponents resulting in four tunnels. The number of calculated tunnels doubles at each iteration.

pipeline could be further optimized but this doesn't represent a limit at the moment: the number of processed opponents is clamped to a smaller value and in racing scenario is very unlikely to interact with more than 8 vehicles at a time[3].

The reshaping logic depends heavily on the role of the opponent:

- **Opponent as Defender (Ego Attacking):** The system uses the forecasted interaction points proactively. At each deviation point, predicted interaction from previous class, the lateral overtaking margin is applied to shape the drivable area. By applying lateral margins on either side in each point of the horizon where an interaction is predicted, the two new tunnels represent overtaking maneuvers on the left and right sides of the defender.

- **Opponent as Attacker (Ego Defending):** Conversely, forecasted interactions are ignored, and the system behaves reactively, enforcing rule-based margins required to comply with racing regulations (see rules in 1.2.4). The system first determines if the opponent has gained right of way (see commitment in 3.2). If commitment is detected, the system attempts to apply a strict **rules margin** along the entire planning horizon on the respective side. However, this application is conditional on available space. If the ego vehicle is already positioned closer to the track edge than the rules margin allows (e.g., during corner entry), the system does not abruptly widen the gap. Instead, it adheres to the **granted border** (see previous section). This ensures that the vehicle maintains the previously calculated valid spacing rather than destabilizing the trajectory with sudden lateral deviations to satisfy the full margin. The two new tunnels thus represent the ego vehicle yielding on the same side and thus are equal.

[3]Given the tracks usual width and vehicle dimensions, meaningfully a maximum of three cars can stay longitudinally aligned. The worst case for the ego-vehicle would be to have three opponents aligned on the front, three aligned on the back, while being in the middle with one opponent aligned in each side

Algorithm 1: Recursive Tunnel Generation

Input: List of base tunnels (starting from track tunnel)
Output: Updated list of tunnels

calcTunnels$(baseTunnels)\{$
if $size(baseTunnels) = nTotalTunnels$ **then**
  return;
$calculatedTunnels = baseTunnels$;
if $opponent.defender$ **then**
  $calculatedTunnels = applyMarginsToOvertake$;
else
  $calculatedTunnels = applyMarginsToLetSpace$;
calcTunnels$(calculatedTunnels)$;
$\}$

To ensure robustness beyond strict rule enforcement, several additional safety overrides are applied. First, an attacker may be temporarily treated as a defender if the predicted trajectories indicate a likely collision and the opponent is *slower* than the ego vehicle. This check is not based solely on alignment: a faster attacker, even if longitudinally aligned with the ego vehicle, is not treated as a defender because no collision risk is predicted. This override prevents unsafe situations where an attacker overtakes, moves in front, and brakes unexpectedly within the overtaking zone, which would otherwise risk being ignored if treated purely as an attacker. Second, when another vehicle is side by side with the ego—specifically when its nose is aligned with the ego wheels until its rear clears the ego front—an additional margin is applied directly relative to the opponent rather than the track boundary. This side-by-side buffer ensures that if the opponent exhibits erratic lateral motion, the ego vehicle retains sufficient space to react safely. Once the opponent is fully ahead (no longer longitudinally aligned), it is ignored unless it is (or becomes) a defender under the above conditions.

All generated tunnels, representing the complete set of feasible drivable areas under the current conditions, are finally forwarded to the subsequent module for feasibility validation and trajectory selection.

Section Summary

This module receives interpolated inputs and determines whether to activate *run-off areas*, setting critical margins and selecting the appropriate environment boundaries (track or walls). It applies lateral overtaking margins when dealing with defenders at each interaction point and rule-based margins when handling attackers in the whole horizon. Predictive approach while attacking, reactive while defending. Safety logic adapts the treatment of attackers depending on predicted collision risk. A recursive expansion generates all possible drivable tunnels (2^N), which are then forwarded to the next stage of the planning framework.

3.4 Tunnels Filter

The primary objective of the *Tunnels Filter* module is to validate the set of tunnels generated by the previous stage, ensuring their feasibility for trajectory optimization. Each candidate tunnel undergoes a rigorous sequence of checks and adjustments before being considered for selection.

Due to the recursive nature of the generation process, the resulting drivable areas may initially exhibit crossed margins or insufficient width in constrained environments, such as near track edges or in scenarios with densely packed opponents. Consequently, the filter first clamps the tunnel borders to the environment bounds and enforces a minimum width constraint. These adjustments ensure that tunnels remain strictly within track limits, borders do not intersect, and each corridor provides sufficient spatial clearance for navigation. It is important to note that this stage guarantees *geometric* feasibility—that the area is drivable in terms of space—rather than *dynamic* feasibility. Further checks concerning vehicle dynamics and maximum acceleration capabilities are performed during the final selection phase.

The refinement of these boundaries is crucial for the stability of the Model Predictive Control (MPC). By eliminating impossible areas—such as those where the maximum boundary is strictly less than the minimum boundary—the filter ensures that the optimization problem remains well-posed. Furthermore, each tunnel is evaluated for overall feasibility, with specific flags assigned to guide the decision-making process in the subsequent module.

3.4.1 Consistency with Relative Positions

Initially, the *EgoLoc* flags calculated in Section 3.2.1 are utilized to determine if a tunnel is *allowed* based on the ego-vehicle's current relative position to the opponents. A consistency check is performed to prevent hazardous maneuvers; for instance, if a tunnel necessitates overtaking on a side opposite to the ego-vehicle's current commitment, it is marked as not *allowed*. This prevents trajectories that would require crossing an opponent's path abruptly. While this check is iterated for each opponent defining the tunnel, it is set to `false` as soon as a single violating opponent is detected, and no further checks are required since the *allowed* flag is defined at the tunnel level.

3.4.2 Boundary Refinement and Minimum Width

Following the consistency checks, the tunnel borders are clamped to the global environment bounds. A parameterized minimum width is strictly enforced to ensure navigability:

- **Track Edges:** If a corridor is too narrow due to proximity to a track edge, the inner border is expanded towards the track center to meet the minimum width requirement.
- **Opponent Squeeze:** If the area is compressed between two or more opponents, the available space is divided equally relative to the midpoint between the closest borders.

While these adjustments guarantee a minimum width, they may result in borders that overlap with the physical bounding box of an opponent. To address this, the *emSuitable* (Emergency Suitable) flag is updated. If any refined border overlaps

with an opponent, the tunnel is marked as not *emSuitable*, indicating it is unsafe even for emergency maneuvers. Conversely, if the tunnel is geometrically clear but flagged as not *allowed* due to other logic (e.g., rule constraints), it remains *emSuitable*. This distinction allows the planner to utilize these tunnels as fallback options in hazardous conditions where the ego-vehicle must escape a critical situation.

3.4.3 Right of Way and Aggressiveness

Once the geometric boundaries are fixed, a final allowance check is performed based on the available width and Right of Way (ROW) logic. The planner evaluates whether the ego-vehicle holds the ROW over specific opponents throughout the prediction horizon. The extent of this temporal evaluation is governed by the *rowAggressiveness* parameter, which dictates how assertively the ego-vehicle behaves (when attacking) based on predicted future states:

- **-1 (Conservative):** ROW is ignored; only the current drivable area width is checked against the safety threshold.
- **0 (Reactive):** Standard rule compliance; ROW is claimed only if valid at the current time step. The same behavior as in defensive mode.
- **1 (Short-Horizon Prediction):** ROW is claimed if the ego-vehicle's Center of Gravity (CoG) is predicted to be ahead of the opponent within the first second of the horizon.
- **2 (Predictive Rule Compliance):** Combines rule logic with predictions, considering ROW valid if gained within the first second.
- **3 (Mid-Horizon Prediction):** Considers ROW valid if gained within the first half of the prediction horizon.
- **4 (Aggressive):** Considers ROW valid if gained at any point within the entire prediction horizon.

In this context, "considering" implies that if the planner predicts the ego-vehicle will gain ROW before the specified time horizon, the tunnel remains *allowed* even if the current width is narrow, under the assumption that the opponent will yield as expected. Conversely, if ROW is not held or predicted, the tunnel is *allowed* only if its width exceeds a parameterized *allowed width*[4].

This parameterization allows for tunable aggressiveness in overtaking maneuvers; its effect on the results are displayed in sec. 4.2. This mechanism enables more aggressive overtaking strategies by allowing planning even when *ROW* is not currently held, but is predicted to be acquired in the near future. Increasing the aggressiveness parameter raises the likelihood of initiating overtakes that rely on long-horizon predictions. While this increases the probability of successful overtaking, it also amplifies the risk associated with inaccurate predictions or unexpected opponent behavior. However, since the planner is updated every 40 ms, it can rapidly adapt to new information.

To mitigate this trade-off, an additional safety check is applied. Even if *ROW* is granted, a tunnel is marked as *too tight* and not *allowed* if the first interaction point is

[4] At the code level, it is enforced that the *allowed width* parameter must be strictly greater than the *minimum width* parameter. Without this constraint, clamped tunnels would essentially bypass the width check, rendering all geometrically valid tunnels as allowed.

already excessively constrained or if a collision is predicted at any interaction point. In cases where a tunnel is only partially *too tight* or partially not *allowed*, a longitudinal constraint is introduced into the tunnel structure. This constraint forces the ego-vehicle to remain behind the specific opponent that cannot be safely overtaken and is defined by the opponent's longitudinal position at the first interaction point where the tunnel becomes infeasible. As a result, even if a tunnel is not fully drivable, the ego-vehicle is prevented from attempting unsafe overtaking maneuvers.

3.4.4 Longitudinal Constraints

In scenarios where a tunnel is partially blocked or deemed *too tight* for safe passage, a longitudinal constraint is imposed. This constraint restricts the ego-vehicle's longitudinal progress to the position of the specific blocking opponent at the point of infeasibility. This mechanism ensures that if a full overtake is not viable, the ego-vehicle defaults to a safe following behavior behind the opponent, rather than colliding or attempting a risky pass. Section 3.6 will elaborate on how this longitudinal constraint integrates with the Adaptive Cruise Control (ACC) system.

The finalized tunnel data, including feasibility flags and blocking opponent information, is then passed to the *Tunnels Selector* module for the final decision.

Section Summary

This module refines tunnel boundaries and assigns feasibility flags to each tunnel, most notably the *allowed* flag.

- **Geometric Feasibility:** Tunnels are clamped to environment bounds and enforced to meet minimum width requirements.
- **Rule Compliance:** If a tunnel's width falls below a safe threshold, it is disallowed unless the *ROW* logic (modulated by an aggressiveness parameter) indicates the opponent is expected to yield.
- **Fallback Logic:** When a tunnel is partially infeasible, a longitudinal constraint is applied to ensure the ego-vehicle follows the opponent safely rather than attempting an unsafe overtake.

3.5 Tunnels Selector

The *Tunnels Selector* determines which tunnel(s) are ultimately provided to the trajectory optimization layer. Two implementations are currently available and were developed sequentially. The first one, the *Parallel Selector* (Section 3.5.1), was designed to exploit multiple Model Predictive Control (MPC) instances to evaluate different tunnels concurrently, leveraging the resulting MPC cost functions for decision-making.[5] Subsequently, the *Single Selector* (Section 3.5.2) was introduced and paired with the MPC discussed in this thesis Ayoub Raji, 2024, where a single MPC instance is used also in multi-vehicle scenarios.

A quantitative comparison between the two approaches has not been performed yet under fully controlled conditions (i.e., identical planning and control modules combined with the same tunnels framework). Nevertheless, qualitative evidence from multiple racing scenarios indicates that both selectors are able to reliably identify appropriate corridors. The main differences between the two approaches are described in the following subsections, while broader considerations on advantages, limitations, and empirical outcomes are deferred to the experimental chapters.

Before detailing each implementation, it is useful to highlight their shared foundations and their main architectural differences. Both selectors are cost-based. However, in the *Parallel Selector* the cost is the one directly produced by each MPC instance, whereas the *Single Selector* uses a custom cost function computed within the selector itself. Moreover, since the *Parallel Selector* requires MPC optimization to be executed before making a decision, the final decision-making logic is effectively moved outside the *Tunnels Framework*. In contrast, the *Single Selector* performs the tunnel decision internally and provides a single corridor to the optimizer. The availability of multiple MPC instances enables a simpler (and typically less error-prone) arbitration logic, but the specific implementation may be riskier or less performant in complex scenarios due to the reduced explicit modeling of decision structure and interaction-specific constraints.

3.5.1 Parallel Selector

When the *Parallel Selector* is used, the output of the *Tunnels Framework* is a set of three tunnels: two tunnels selected from the validated candidates, plus the track tunnel. The track tunnel is an interpolated representation of the full environment (with optional safety margins) augmented with a longitudinal constraint to remain behind opponents. It is always provided to one MPC instance as a fallback option in case the two selected corridors result infeasible. Consequently, at least three MPC instances are required when adopting this selector.

The two candidate tunnels are selected within the *Tunnels Framework* based on the *allowed* flag. The leftmost and rightmost *allowed* tunnels are selected whenever they exist. Otherwise, if on one side no *allowed* corridor is available, the corresponding outermost tunnel on that side is kept regardless of feasibility. When no *allowed* tunnel exists at all, the two outermost tunnels are still selected, but the track tunnel becomes the only viable option in practice.

This strategy prioritizes the first feasible corridor on each side, implicitly encouraging maneuvers with fewer side changes. As a consequence, mixed tunnels are indirectly penalized, especially for opponents that interact earlier in the horizon (consistent with the opponent reordering and the recursive generation process).

[5] The MPC formulation adopted in the first implementation, along with the whole motion planning and control scheme, follows the approach presented in Prignoli et al., 2026.

Once the three tunnels are selected, they are forwarded to independent MPC instances running in parallel. Each MPC computes a trajectory, adapts the speed profile within its assigned corridor, and outputs a cost value reflecting the quality of the resulting solution. The specific cost structure depends on the MPC implementation; in general, it includes terms promoting adherence to the racing line, smoothness, and vehicle stability.[6]

In addition to the cost value, the MPC exit status (i.e., the solver termination flag) is used to assess whether a valid numerical solution has been found within the assigned tunnel. This is critical because a tunnel may be geometrically valid but still infeasible once vehicle dynamics and optimization constraints are considered. If no solution is found, or if the solution is deemed suboptimal according to the solver status, the corresponding tunnel is treated as infeasible for decision-making.

The arbitration logic follows. If exactly one of the two selected tunnels is *allowed* and yields a valid MPC solution, that tunnel is selected directly (without comparing costs), so that a safe option is always prioritized when available. If both tunnels are *allowed* and both MPC optimizations succeed, their costs are compared and the tunnel with lower cost is selected for execution. To reduce frequent switching due to minor cost differences, a hysteresis mechanism is applied: once a tunnel is selected, its cost is scaled by a factor before being compared with the alternative, reducing the likelihood of switching unless a significant advantage is observed. Overall, this approach exploits the MPC optimization outcome to select the most suitable maneuver when multiple feasible corridors exist.

3.5.2 Single Selector

The *Single Selector* outputs exactly one tunnel, thus requiring only one MPC instance. To select the most suitable corridor among the validated candidates, the selector evaluates a cost function for each tunnel and combines it with tunnel-level feasibility flags. The decision process follows three main stages: (i) a trajectory is approximated within each candidate corridor, (ii) the tunnel cost is evaluated, and (iii) a tunnel is selected (or forced) depending on the availability of *allowed* corridors.

Trajectory generation

The trajectory computed within each tunnel is not model-based and is only used as a rough estimate of curvature, intended to quantify how strongly the maneuver deviates from the racing line. A full optimization-based trajectory generation would be computationally prohibitive given the number of tunnels to evaluate. Instead, a lightweight procedure is adopted to approximate the lateral deviation required to remain within the corridor.

The first and last point of the horizon, together with each interaction change, define a set of waypoints. These N waypoints partition the horizon into $N-1$ longitudinal sections. Within each section, if a lateral offset n_{tunnel} is required to remain inside the tunnel starting from the current lateral position n_0, a scaled hyperbolic tangent function is used to generate a smooth transition.

Given the amplitude $\Delta n = n_{\text{tunnel}} - n_0$, the longitudinal position within the section s_i, the section inflection point s_p (defaulting to the section midpoint), and a

[6] Details of the cost function used in our MPC implementation when testing the *Parallel Selector* are reported in Section 5.2.1.

curvature factor k, the normalized distance z is defined as:

$$z = \frac{s_i - s_p}{k}. \tag{3.4}$$

The lateral profile $y(s)$ is mapped to the range $[0, \Delta n]$ using the sigmoid form:

$$y(s) = \frac{\Delta n}{2} \left(\tanh(z) + 1\right). \tag{3.5}$$

Although this provides a smooth lateral transition between waypoints, the resulting trajectory may violate tunnel borders depending on the corridor shape and on the raceline curvature. Therefore, the generated profile is validated against tunnel limits; if violations occur, the inflection point s_p and then the curvature factor k are iteratively adjusted until the trajectory remains within the tunnel.

To compute curvature, the first and second derivatives are required. Using $\frac{d}{dz}\tanh(z) = 1 - \tanh^2(z)$ and the chain rule $\frac{dz}{ds} = \frac{1}{k}$:

First derivative:

$$y'(s) = \frac{\Delta n}{2k} \left(1 - \tanh^2(z)\right). \tag{3.6}$$

Second derivative:

$$y''(s) = -\frac{\Delta n}{k^2} \tanh(z) \left(1 - \tanh^2(z)\right). \tag{3.7}$$

The instantaneous curvature κ of the resulting plane curve is computed as:

$$\kappa(s) = \frac{y''(s)}{\left(1 + [y'(s)]^2\right)^{3/2}}. \tag{3.8}$$

This curvature is not meant to approximate the exact curvature that the vehicle will follow during the maneuver. In Frenet coordinates, it represents only a local deviation from the reference raceline (which has zero curvature in the Frenet representation). Nevertheless, it provides a practical proxy for maneuver aggressiveness over a short horizon without reconstructing a full raceline. To obtain the total curvature, the value computed here is summed with the reference curvature provided as input to the *Tunnels Framework*.[7]

The resulting curvature is then used to evaluate the dynamic feasibility of the maneuver inside the corridor by comparing the required lateral acceleration against tire physical limits. Given instantaneous curvature κ and longitudinal velocity v_x, the lateral acceleration a_y (adjusted for banking angle ϕ) is:

$$a_y = v_x^2 \cdot \kappa \cdot \cos(\phi). \tag{3.9}$$

The available acceleration depends on the total vertical load F_z, which includes static load, aerodynamic downforce, and banking-induced load:

$$F_{z,\text{total}} = mg\cos(\phi) + \frac{1}{2}\rho S C_l v_x^2 + m a_y \tan(\phi). \tag{3.10}$$

Load distribution between axles depends on the center-of-gravity position. Let l_f and l_r be the distances from the center of gravity to the front and rear axles, and

[7] All quantities required by the following checks (e.g., banking, maximum acceleration, longitudinal speed) are provided as inputs to the framework.

$L = l_f + l_r$ be the wheelbase. The load distribution factor cog is:

$$cog = \frac{l_r}{l_f + l_r}. \tag{3.11}$$

Consequently, the vertical loads F_{zf} and F_{zr} are computed as:

$$F_{zf} = (F_{z,0} + F_{z,\text{bank}}) \cdot cog + F_{z,\text{aero}} \cdot s_{\text{front}}, \tag{3.12}$$
$$F_{zr} = (F_{z,0} + F_{z,\text{bank}}) \cdot (1 - cog) + F_{z,\text{aero}} \cdot (1 - s_{\text{front}}). \tag{3.13}$$

Using a simplified Pacejka Magic Formula model, the maximum lateral force for each axle is:

$$F_{yf,\max} = F_{zf} \cdot (D + s_v) \cdot e + mg \sin(\phi) \cdot cog, \tag{3.14}$$
$$F_{yr,\max} = F_{zr} \cdot (D + s_v) \cdot e + mg \sin(\phi) \cdot (1 - cog). \tag{3.15}$$

A trajectory is considered dynamically feasible (and the tunnel thus *allowed*) only if the required lateral acceleration does not exceed the maximum achievable acceleration:

$$a_{y,\text{ego}} \leq \frac{F_{yf,\max} + F_{yr,\max}}{m}. \tag{3.16}$$

If the condition is violated at any point, the maneuver is flagged as unsafe due to lateral demand exceeding the available grip. The exploration factor e scales slightly above the estimated grip peak to avoid overly conservative rejection due to the approximate nature of the constructed trajectory. For the same reason, and to avoid aborting committed overtaking maneuvers, this dynamic check is performed only when the *egoLoc* is *back* (Section 3.2.1) with respect to a specific opponent. When approaching an opponent, lateral demand typically increases and the hyperbolic-tangent profile become unrealistically aggressive in this phase, leading to unnecessary rejection of feasible tunnels.

Cost function

The tunnel cost (C_{tunnel}) in Eq. 3.17 combines a term encouraging continuity with the previously selected tunnel (C_{prev}), a drivable-area term (C_{area}), and a trajectory aggressiveness term (C_{traj}):

$$C_{\text{tunnel}} = C_{\text{prev}} + C_{\text{area}} + C_{\text{traj}}. \tag{3.17}$$

The term C_{prev} biases the selection toward the previous decision to reduce undesired oscillations, unless a tunnel becomes *not allowed*. In this context, "best match" with the previous tunnel does *not* mean selecting the same tunnel index across iterations. Instead, it refers to matching the previous decision at the level of each interacting opponent: for every opponent, the selector compares whether the ego-vehicle was previously planning to overtake on the left or on the right (as encoded by the tunnel branch generated for that opponent in the recursive construction). The current tunnel is therefore penalized if, for one or more opponents, it implies an overtake side different from the one selected in the previous iteration.

To avoid over-constraining the decision, this continuity penalty is weighted more strongly for opponents that interact earlier in the horizon (and are thus more critical for immediate maneuver consistency), while allowing progressively more flexibility for opponents that interact later. This is implemented through an exponential decay

weighting. Let N be the total number of opponents and i be the index of the current opponent in the reordered list (with smaller i corresponding to earlier interactions). The decay factor δ_i is defined as:

$$\delta_i = \exp\left(\lambda \cdot (N - i)\right), \tag{3.18}$$

where λ is a tunable decay constant controlling how quickly the influence decreases. With this formulation, changes of overtake side for the first interacting opponents are penalized the most (promoting strong short-term continuity), while changes associated with later interactions receive a smaller penalty, enabling the planner to adapt the chosen side further ahead if needed.

The term C_{side} acts as a multiplicative penalty on the area cost, increasing emphasis on mixed tunnels: the larger the number of side alterations, the higher the effective cost. The drivable-area contribution is obtained by accumulating tunnel width along the horizon, so that larger corridors yield smaller costs. For convenience, the corresponding accumulated metric is denoted as C_{area}:

$$C_{\text{area}} = C_{\text{dimension}} \cdot C_{\text{side}}. \tag{3.19}$$

Finally, C_{traj} accumulates the estimated curvature from Eq. 3.8 over the horizon. Higher curvature leads to higher cost, discouraging aggressive maneuvers. In practice, each term in Eq. 3.17 is multiplied by a tunable weight, which can be adjusted to match the desired driving style and performance objectives.

Tunnel selection

At this stage, the selector performs the decision-making. Differently from the *Parallel Selector*, exactly one tunnel is always selected. The track tunnel is returned only when no interactions are predicted.

First, the selector checks whether at least one tunnel is *allowed*. If at least one *allowed* tunnel exists, the selection is straightforward: the costs of all *allowed* tunnels are compared and the tunnel with minimum cost is selected. If no *allowed* tunnels exist, the selector switches to a forced-choice logic.

In the forced-choice case, the first branch depends on whether the ego-vehicle is side-by-side with an opponent. If the ego is side-by-side, two scenarios are considered:

- **All opponents are on the same side:** the outermost tunnel on the opposite side is selected, maximizing separation from the opponent group.
- **Opponents exist on both sides:** one of the mixed tunnels is selected by choosing, for each opponent, the overtake side opposite to the opponent location (e.g., if *egoLoc* is *left*, select the tunnel that overtakes on the right, and vice versa). This check is performed twice: first requiring the tunnel to be *emSuitable*, and, if no candidate is found, repeating the same logic while ignoring *emSuitable*. In both cases, if multiple tunnels satisfy the conditions, the one with the lowest cost is selected.

If the ego-vehicle is not side-by-side with any opponent, the selector first checks whether at least one tunnel is *emSuitable*. If so, the *emSuitable* tunnel with lowest cost is selected. Otherwise, the selection is initialized using the best match of the

previous tunnel as a fallback (in the worst case, the last decision is retained).[8] Then, all tunnels are evaluated to identify candidates that, even if not *emSuitable*, exhibit at least non-overlapping bounds with any opponent when vehicles are longitudinally aligned in the predictions (neglecting longitudinal safety margins and vehicle lateral dimensions). If such tunnels exist, the candidate with minimum cost is selected. This does not guarantee collision avoidance; rather, it provides a principled way of selecting the least hazardous option among infeasible candidates.

It is important to emphasize that, in the forced-choice regime, the selected tunnel is not *allowed*. Therefore, it is likely that at least one longitudinal constraint is active (see the next section). As a consequence, selecting a tunnel via fallback does not necessarily imply a collision, provided that longitudinal constraints are properly enforced.

Tunnel statistics

In the final step, for debugging and analysis purposes, additional statistics are computed and logged. An heuristic probability model evaluates two metrics: *Success Probability* and *Collision Risk*. Both rely on a set of *Probability Terms* T derived from tunnel geometry and environment. Let T_{area} be the normalized width ratio, T_{side} the number of side changes, T_{opps} the number of interacting agents, T_{traj} the normalized trajectory cost, and T_{forecast} the prediction confidence (kept constant at the moment).

The **Success Probability** P_{success} follows a multiplicative decay model, starting from $P_{\max}$ and reduced by exponential penalty terms:

$$P_{\text{success}} = P_{\max} \cdot (1 - T_{\text{area}}) \cdot e^{-(T_{\text{side}})} \cdot e^{-(T_{\text{opps}})} \cdot e^{-(T_{\text{forecast}})} \cdot \Pi_{\text{yield}}, \tag{3.20}$$

where Π_{yield} is a reliability penalty applied only when an opponent yield is required. The final value is clamped such that $P_{\text{final}} \in [P_{\min}, P_{\max}]$.

In contrast, the **Collision Risk** $R_{\text{collision}}$ is computed through an additive model:

$$R_{\text{collision}} = \min\left(\sum_i (T_i) + R_{\text{yield}}, R_{\max}\right). \tag{3.21}$$

Expanding the summation:

$$R_{\text{base}} = T_{\text{area}} + T_{\text{side}} + T_{\text{opps}} + T_{\text{traj}} + T_{\text{forecast}} + R_{\text{yield}} \tag{3.22}$$

If the tunnel is flagged as a *Forced Tunnel*, the risk is doubled to reflect the lack of alternatives:

$$R_{\text{final}} = \begin{cases} 2.0 \cdot R_{\text{base}} & \text{if forced,} \\ R_{\text{base}} & \text{otherwise.} \end{cases} \tag{3.23}$$

Similarly to the cost function, each *Probability Term* T is actually multiplied by a tunable weight, which can be adjusted based on empirical evidence.

[8] Same decision does not necessarily imply the same tunnel instance: the tunnel is recomputed at each iteration and a best-match mapping is applied to account for opponent reordering. If the number of agents changes (e.g., a new opponent appears or disappears), the previous index is matched accordingly.

Section Summary

The *Tunnels Selector* determines which tunnel(s) are provided to the MPC layer. The *Parallel Selector* evaluates two candidate tunnels (plus a fallback track tunnel) by running multiple MPC instances in parallel and selecting the best corridor based on solver feasibility and MPC cost, with an additional hysteresis to prevent frequent switching. The *Single Selector* outputs a single tunnel and relies on a lightweight internal evaluation: a hyperbolic-tangent trajectory is generated to estimate curvature and dynamic feasibility, and a custom cost function combines continuity with the previous decision, drivable area, side-change penalties, and aggressiveness proxies. When no *allowed* tunnel exists, a forced-choice logic selects the least hazardous corridor and typically activates longitudinal constraints to prevent unsafe overtakes.

3.6 Cruise Control Integration

While the trajectory optimization layer operates on the output of the *Tunnels Selector*, an intermediate module is required to integrate longitudinal constraints derived from the tunnel geometry into the Adaptive Cruise Control (ACC) system.

The activation of these longitudinal constraints within a selected tunnel is determined by specific operational scenarios:

- **Overtaking Prohibitions:** Constraints are strictly enforced when the autonomous stack signals a "follow-only" mode. This typically occurs during yellow flag conditions, pit-lane entries, or other scenarios where the ego vehicle is required to maintain position behind other vehicles.
- **Nominal Racing:** During standard racing conditions, constraints are applied if the tunnel is designated as *not allowed*, is deemed *too tight*, or if an opponent is blocking the track without yielding.
  - The system imposes a longitudinal constraint on a specific agent identified as a "defender" (see Fig. 3.3) if the ego vehicle does not possess the right of way.
  - Conversely, the longitudinal constraint is disregarded if the aforementioned conditions are not met, if the ego vehicle is already side-by-side with the opponent, or if the opponent is off-track (preventing the ego vehicle from being blocked by vehicles that are stationary or re-entering the track).

In the Frenet coordinate frame, this constraint is formulated as a maximum longitudinal position profile, denoted as a vector $\mathbf{s}_{\max}$, defined over the prediction horizon. This profile tracks the forecasted longitudinal position of the opponent, adjusted by a safety margin to ensure an adequate following distance.

The objective of the ACC is to achieve the target following distance while matching the opponent's velocity at steady state. The control strategy adapts based on four relative state configurations:

1. **Far distance, lower speed:** The ACC accelerates to simultaneously close the gap and match the target speed.
2. **Far distance, higher speed:** The ACC decelerates while closing the distance to converge to the target speed.
3. **Close distance, lower speed:** The ACC accelerates smoothly to reach the target speed without exceeding the safety distance.
4. **Close distance, higher speed:** The ACC decelerates to compensate for deviations in both speed and distance.

Two distinct integration strategies were developed for the motion planning and control layer. The first approach, utilized with the *Parallel Selector*, incorporates a direct longitudinal limit into the Model Predictive Control (MPC) formulation. The second approach, applied to the *Single Selector*, converts the constraint into a reference speed profile for the MPC to track. These methods are detailed below.

3.6.1 ACC as Longitudinal Soft Constraint

In this configuration, the ACC module integrates the longitudinal constraint into the optimization problem by enforcing that the ego vehicle's longitudinal position remains less than or equal to $s_{\max}$ at each time step. Similar to the lateral constraints within the tunnels, this is implemented as a soft constraint to ensure solver feasibility.

To guarantee safe interaction with a leading defender opponent, denoted as D, the longitudinal planner enforces mixed state–input constraints on the predicted arc-length trajectory $s_{k|t}$. Given a forecast of the defender's trajectory $\{s_{\mathrm{D},k|t}\}_{k=0}^{N}$, a minimum spatial gap is imposed as:

$$s_{k|t} \leq s_{\mathrm{D},k|t} - s_{\mathrm{safe}}, \quad \forall\, k \in \mathcal{I}_0^N, \tag{3.24}$$

where s^{safe} represents a fixed safety margin.

Furthermore, a time-gap constraint ensures a minimum temporal headway proportional to the ego vehicle's velocity:

$$s_{k|t} \leq s_{\mathrm{D},k|t} - t_{\mathrm{gap}}\, v_{x,k|t}, \quad \forall\, k \in \mathcal{I}_0^N, \tag{3.25}$$

where t^{gap} is the desired time headway parameter. Both equations (3.24) and (3.25) are affine with respect to the optimization variables $(s_{k|t}, v_{x,k|t})$, thereby preserving the convexity of the MPC formulation. These constraints realize an ACC behavior at the planning level, enabling the vehicle to adapt its speed and maintain safe distances .

3.6.2 ACC as Target Speed Profile

Alternatively, the **Adaptive Cruise Control (ACC)** module can regulate the ego vehicle's velocity to maintain a safety distance by generating a target speed profile. The control logic transitions through three primary phases: state determination, Feed-Forward speed calculation, and PID-based refinement.

1. ACC Status Determination

The operational state is determined by evaluating the gap error and speed error against configurable thresholds. Let d_{target} be the desired following distance and s_0 be the current longitudinal distance to the lead vehicle.

- **FOLLOWING:** This state is active if the absolute gap error $|d_{\mathrm{target}} - s_0|$ and the absolute speed error $|v_{\mathrm{ego}} - v_{\mathrm{opp}}|$ are below the defined thresholds ϵ_{dist} and ϵ_{vel}, respectively.
- **CLOSING:** If the thresholds are exceeded, the system transitions to a closing state, further categorized by the feasibility of deceleration:
  - **NOMINAL:** Sufficient deceleration is achievable within standard comfort limits.
  - **CRITICAL:** Maximum mission-allowed deceleration is required to ensure safety.
  - **AVOIDANCE:** Maximum deceleration is insufficient, or the distance falls below the avoidance threshold, defined as $d_{\mathrm{avoid}} = k_{\mathrm{avoid}} \cdot d_{\mathrm{target}}$.

2. Feed-Forward (FF) Control

The Feed-Forward velocity, v_{ff}, is derived using uniformly accelerated motion equations to close the gap Δs. The target velocity change, Δv_{target}, is calculated as:

$$\Delta v_{\text{target}} = \sqrt{(v_{\text{opp}} - v_{\text{ego}})^2 + 2 \cdot a_{\text{target}} \cdot |\Delta s|} \tag{3.26}$$

where $a_{\text{target}} = |a_{\max}| \cdot k_{\text{ff}}$ represents the tracking aggressiveness. The resulting reference speed is given by:

$$v_{\text{ff}} = \max\left(k_{\min} \cdot v_{\text{opp}},\, v_{\text{opp}} - \text{sgn}(\Delta s) \cdot \Delta v_{\text{target}}\right) \tag{3.27}$$

This speed is subsequently saturated based on the vehicle's dynamic acceleration limits and the discretization time step Δt:

$$v_{\text{ff}} \in [v_{\text{ego}} + a_{\min}\Delta t,\, v_{\text{ego}} + a_{\max}\Delta t] \tag{3.28}$$

3. PID Correction and Final Velocity

When the Proportional-Integral-Derivative (PID) controller is enabled, the system computes two corrective contributions:

- **Speed PID:** Minimizes the relative velocity error, $\delta v = v_{\text{opp}} - v_{\text{ego}}$.
- **Distance PID:** Minimizes the spatial gap error, $\Delta s = d_{\text{target}} - s_0$.

The final target velocity, v_{acc}, is the sum of the feed-forward and PID terms, bounded by the planner's maximum allowable speed, v_{plan}:

$$v_{\text{acc}} = \min\left(v_{\text{plan}},\, v_{\text{ff}} + v_{\text{pid,speed}} + v_{\text{pid,dist}}\right) \tag{3.29}$$

4. Velocity Profile Integration

Finally, the velocity profile is updated to ensure longitudinal consistency. The new velocity constraints are interpolated along the tunnel coordinates to provide a smooth input for the *Longitudinal Planner*:

$$V_{\text{profile}}(s) = \text{interp1}(S_{\text{planner}}, V_{\text{max\_tunnel}}, S_{\text{long}}) \tag{3.30}$$

In manual mode (where PID control is disabled), the safety distance is directly subtracted from the tunnel's longitudinal bounds to physically restrict the searchable space available to the planner.

Section Summary

This section details the integration of longitudinal constraints from the Tunnels Selector into the Adaptive Cruise Control (ACC) system. It defines activation conditions based on overtaking rules and racing scenarios (nominal vs. defensive). Two implementation strategies are presented: (1) imposing soft longitudinal position constraints within the MPC formulation to maintain spatial and temporal safety gaps, and (2) generating a target speed profile via a multi-stage logic involving state determination, feed-forward control, and PID refinement.

Chapter 4

Strategies to Balance Performance and Safety

The deployment of autonomous racing agents at the edge of vehicle dynamics necessitates a sophisticated balance between competitive performance and operational safety. While the previous chapter detailed the architectural components of the Tunnels Framework, this chapter explores the conceptual insights and strategic trade-offs required to manage multi-agent interactions under uncertainty. Central to this discussion is the role of forecasting and the varying degrees of trust placed in opponent predictions. By analyzing the interplay between aggressive and conservative planning strategies, we justify the design choices that enable the system to navigate complex racing scenarios while adhering to regulatory constraints and physical limits.

4.1 Forecasting of Other Agents: Evolution and Challenges

Although the development of forecasting modules is not the primary focus of this thesis, their accuracy is a critical input for the Tunnels Framework. In high-speed autonomous racing, anticipating the future states of competing agents is paramount. However, forecasting remains one of the most significant challenges due to the inherent unpredictability of opponents. Agents may utilize tactical maneuvers such as *Push-to-Pass* (P2P), exhibit varied braking points, or execute sudden defensive blocks.

To establish a baseline for prediction, the framework employs a racing line matching strategy. This involves a majority voting scheme over a sliding time window to identify the closest global raceline for each agent independently. There is a batch of offline computed racelines. In testing it can happen that the ego or other agents stay on lines that are not optimal but all inner or all outer. In racing scenarios these lines parallel to bounds are removed (because they do not represent valid racing trajectories) and just the racing line is retained; in some cases, multiple lines may be considered to capture different racing strategies (early or late apex, staying on a side in a straight,...). This matched reference allows for the extraction of local curvature and a nominal speed profile. The longitudinal planner then account for the agent's current velocity and merge it to the speed profile that is scaled based on performance. At the moment the performance of the opponent is not estimated online, but a fixed scaling factor is applied.

Throughout the evolution of the UNIMORE Racing software stack, three primary solutions for multi-agent forecasting have been developed and evaluated.

4.1.1 Basic Approach: Constant Velocity and Kalman Filtering

Initially, the system utilized a standard Kalman Filter to track opponents, assuming constant velocity along the racing line. While it can be effective in oval and tri-oval speedways where trajectories are quite predictable and the longitudinal accelerations are low, this approach don't generalize well with more complex maneuvers and ignore the interactions between agents.

4.1.2 Traditional Approach: Frenet-based Polynomial Planning

The subsequent solution relied on a Frenet-frame planner where the matched racing line served as the primary reference. In scenarios with interacting opponents, the system generated a pool of candidate polynomials, selecting the one with the minimum cost based on target lateral offsets (d-coordinates) and velocities. However, it was primarily suited for head-to-head scenarios and lacked the ability to predict complex multi-vehicle interactions.

- **Advantages:** The approach is computationally lightweight and robust to state estimation noise. It provides stable overtaking decisions when longitudinal gaps are large.
- **Limitations:** The framework does not guarantee dynamic feasibility of the output trajectory and often suffers from poor overtake accuracy in high-speed delta scenarios. This is not directly tracked by the ego but having a not feasible trajectory introduce a sure error and influence also the tunnel shape with something not realistic. Furthermore, it tends to clamp aggressively to track boundaries in high-curvature sections and lacks the logic to manage complex "one-vs-many" interactions.

4.1.3 Latest Approach: Stanley and Tunnels Framework Integration

The current implementation utilizes a more sophisticated architecture. Here, a full speed reference is generated, and an independent instance of the *Tunnels Framework* 3.1.1 is instantiated for each opponent to manage mutual interactions. The trajectory calculated within the selected corridor using the *Single Selector* 3.5.2, is used as a reference for a Stanley steering controller, driving a bicycle model and a longitudinal PI controller.

- **Advantages:** Ensures trajectories are physically feasible by running a bicycle model. It manages multi-vehicle interactions and rule compliant behavior through an internal instance of the Tunnels Framework for each opponent. It also enables a seamless transition to Adaptive Cruise Control (ACC) if a tunnel is blocked.
- **Limitations:** The increased complexity requires significant integration and tuning effort. Being heavily dependent on the vehicle model, it is more susceptible to model mismatch may exhibit flickering if the underlying overtaking logic switches references too frequently.

4.2 Methodological Validation: Static Testing

Before proceeding to full-scale simulations and track tests, the framework is validated through deterministic static tests. Unlike dynamic simulations, a static test

isolates a single iteration of the *Tunnels Framework*, allowing for the injection of artificial opponent states and trajectory predictions.

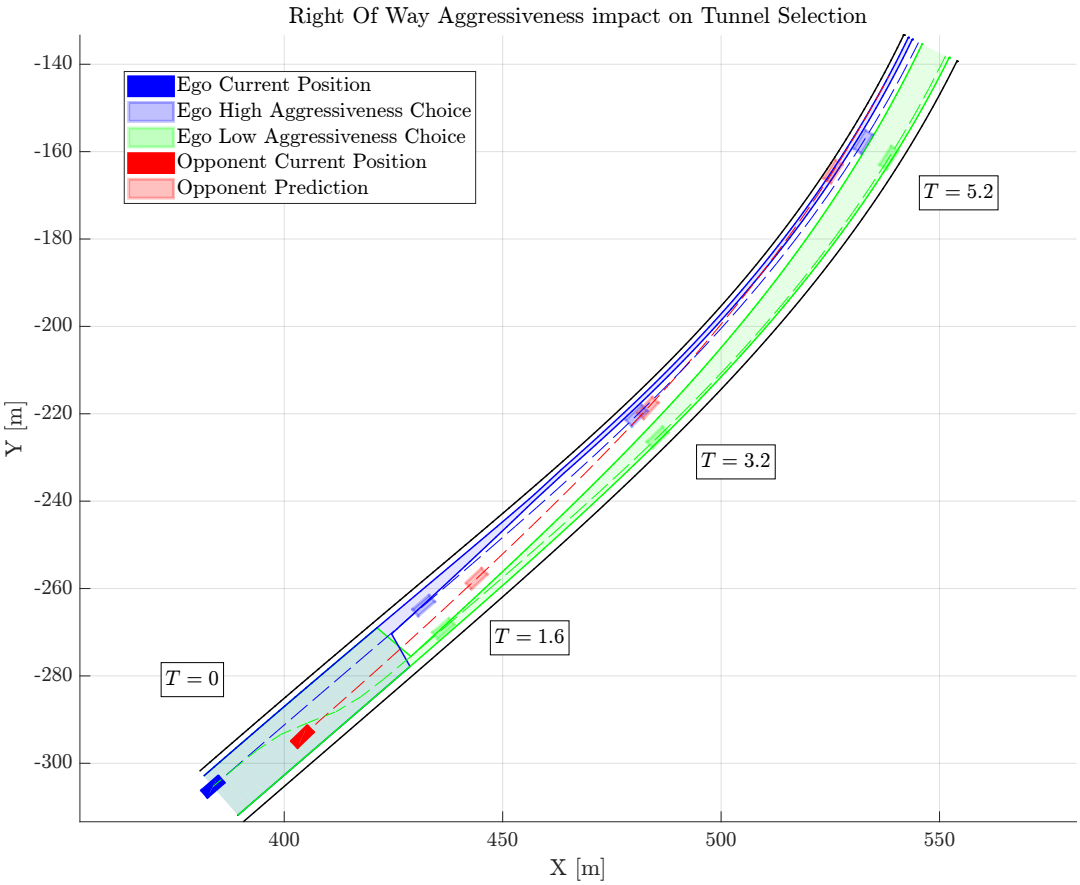


FIGURE 4.1: Overlap of two static tests. The blue tunnel represents maximum aggressiveness (High-Trust), while the green tunnel represents minimum aggressiveness (Low-Trust).

Static scenarios are crucial for code validation and ensuring consistency between C++ implementations. They allow for precise fine-tuning of cost functions and the analysis of prediction errors on tunnel generation. However, it is important to note that static tests are limited to spatial validation; they cannot evaluate the temporal stability (e.g., flickering) between consecutive frames. A static test provides precise control by positioning opponents with set distances and velocity deltas while injecting artificial trajectory predictions. Unlike full simulations, it isolates a single iteration of the *Tunnels Framework* as an intermediate step before full validation. Its key advantage lies in their deterministic, repeatable nature, facilitating fine-tuning of cost functions and analyzing prediction effects on tunnel generation and decision-making.

For instance, Fig. 4.1 illustrates the outcome of two static tests performed using the *Single Selector*. In this scenario, a single opponent is positioned 23 m in front of the ego vehicle. Both vehicles are entering a corner, but the ego vehicle has a speed advantage of 15% higher than the opponent. The two tests were conducted with identical configurations, except for a variation in the right-of-way aggressiveness parameter. While this parameter does not affect how tunnels are initially computed, it influences tunnel selection. Referring to Eq. 3.17, in a static test, the cost of the previously selected tunnel is null, and the C_{side} has no influence as there are no mixed tunnels due to the presence of only one opponent. The right tunnel (green) has a

larger area, leading to a lower cost. However, its trajectory (dashed green line in Fig. 4.1) deviates significantly from the racing line (dashed blue line), increasing curvature and overall cost. In contrast, the left tunnel has a slightly higher cost due to its smaller area but is more favorable in this case, as the racing line remains inside it where the tunnel narrows, eliminating additional curvature costs. The *right-of-way aggressiveness* parameter affects how opponent predictions influence tunnel selection. A lower setting considers only the current positions, verifying whether the ego vehicle is already within right of way margins. A higher setting evaluates the full prediction horizon, determining if the tunnel will narrow at a point where the other agent should already yield. Intermediate values progressively incorporate parts of the prediction. In the tests shown in Fig. 4.1, the right-of-way distance is set to 10 m, smaller than the initial gap between vehicles at $T = 0$ s. With low aggressiveness, the planner selects the right tunnel (green) since the left tunnel is too narrow and thus *not allowed*. At $T = 1.6$ s, a predicted interaction suggests that vehicles will enter the right-of-way margin, so with high aggressiveness, the planner assumes the opponent will yield space, keeping the left tunnel allowed. At $T = 3.2$ s, if the opponent does not yield, the left tunnel would lead to a collision, illustrating how careful parameter tuning is required to balance rule compliance and trust in opponent behaviors.

Impact of Aggressiveness on Tunnel Selection The level of "trust" in predictions directly modulates the system's aggressiveness. This is primarily controlled via the *rowAggressiveness* parameter, which dictates how the selector interprets future Right-of-Way (ROW) status:

1. **Low Aggressiveness:** The planner relies on current spatial gaps and ignores long-term ROW predictions. Overtakes are initiated only when the available width is already sufficient. In this particular case planner selects the right tunnel (green) because the left tunnel is deemed "not allowed" due to current safety margins.

2. **High Aggressiveness:** The planner assumes opponents will yield if the ego-vehicle is predicted to gain ROW at any point in the horizon. This enables earlier and more decisive maneuvers but increases the risk of collision if the opponent does not comply with racing regulations. In this case the planner evaluates the full prediction horizon, thus keeping the left tunnel (blue) available, which is more favorable as it aligns with the racing line.

This demonstrates how the *rowAggressiveness* parameter functions as a proxy for trust in opponent behavior and rule compliance.

4.3 Trust in Predictions and Risk Management

The efficacy of the Model Predictive Control (MPC) and the Tunnels Framework is intrinsically linked to the reliability of both ego-vehicle and opponent forecasting.

4.3.1 Ego-Vehicle Prediction Mismatch

As discussed in Chapter 3, the framework utilizes a longitudinal planner that assumes an "always overtaking" profile for ego-prediction. While this is a conservative approximation intended to detect interactions early, it introduces inaccuracies

that can accumulate over the prediction horizon. Mismatches between the internal vehicle model and real-world dynamics can lead to tracking errors in the MPP and MPC.

A critical risk arises when a valid tunnel appears abruptly due to a late detection or a sudden change in predicted interactions. In such cases, the predictive advantage of the MPC is diminished, forcing the system into a reactive regime. This can lead to aggressive maneuvers, sub-optimal tracking of tunnel boundaries, or, in extreme cases, optimization infeasibility.

4.3.2 Robustness to Estimation and Tracking Errors

Geometric Representations: Tunnels' Shapes and Margins

The choice of geometric constraints significantly influences the MPC's behavior.

- **Ellipsoidal vs. Rectangular Bounds:** To reduce trajectory curvature during overtaking maneuvers, ellipsoidal bounds have been evaluated as an alternative to the rectangular bounds previously introduced. Ellipsoidal bounds enable smoother trajectories and a more efficient use of the available space, potentially improving performance. However, this comes at the cost of reduced robustness. Rectangular bounds, while more conservative, better account for uncertainty in opponent predictions and guarantee a safer separation. In particular, when comparing an ellipse and a rectangle centered on the same predicted interaction region, the main difference lies in the rectangle corners. These regions implicitly capture the combined effect of longitudinal and lateral prediction uncertainties, which are instead omitted by the ellipsoidal approximation. Simulation results confirm this trade-off: ellipsoidal bounds promote more aggressive behaviors and increase overtaking opportunities due to larger drivable areas, but they also lead to a higher collision rate in the presence of longitudinal prediction errors. For this reason, rectangular bounds have been selected as the default representation and are the only ones adopted to ensure safety during real-world on-track experiments.

- **Centering Strategies:** Centering the tunnel on predicted interaction points allows the corridor to naturally adapt to relative speed differences, effectively stretching or shrinking along the longitudinal direction. This results in a more anticipative and efficient behavior compared to purely reactive approaches. The main limitation of this strategy is that other agents may lie partially outside the constructed tunnel. In the presence of prediction errors, this could potentially lead to unsafe situations. However, in practice, prediction uncertainty is not uniform along the horizon: errors tend to grow in the far future, while the initial portion—especially the current detection—remains relatively accurate. Moreover, the tunnel is recomputed at high frequency (e.g., 20 Hz), enabling continuous adaptation to the evolving scenario. In practice, when interacting with vehicles of similar performance, relative speed variations between successive updates remain limited, reducing the likelihood of critical mismatches due to prediction errors. An alternative approach consists in centering the tunnel directly on the surrounding vehicles rather than on interaction points. While this guarantees that all agents are always enclosed, it leads to overly conservative behavior. In particular, it implicitly assumes that surrounding vehicles have zero velocity, resulting in artificially large relative speed estimates and a purely reactive strategy that anticipates interactions too

early. This prevents the ego vehicle from naturally aligning with opponents (e.g., along similar racing lines or during slipstreaming) and does not exploit right-of-way conditions. As a consequence, the vehicle tends to deviate prematurely, increasing lap time and potentially inducing more abrupt maneuvers when close interactions eventually occur. For these reasons, interaction-based centering is adopted as the default strategy in the proposed framework.

- **Dynamic Margins:** To ensure consistent behavior across the full speed envelope, safety margins are linearly scaled with the ego-vehicle velocity. At high speeds, larger buffers are required to compensate for reduced reaction times and system latency (e.g., a 40 ms delay at 250 km/h corresponds to a displacement of approximately 2.7 m). Conversely, at lower speeds, reduced margins allow the vehicle to operate closer to opponents, particularly during braking phases. This effectively enlarges the available drivable area in corner entry regions, which are typical overtaking opportunities. In contrast, static margins impose a suboptimal trade-off across different scenarios, either being overly conservative at low speeds or insufficiently safe at high speeds. This adaptive formulation enables a better balance between safety and performance without requiring scenario-dependent tuning.

Systemic Failures and Fallback Mechanisms

The framework must remain robust to unpredicted scenarios and edge cases arising from perception and prediction errors. These include false positives (where a non-existent interaction unnecessarily constrains the ego vehicle), false negatives (where a missed interaction may lead to a potential collision), as well as sudden appearances or disappearances of agents due to detection failures. In such conditions, the existence of a fully safe tunnel cannot always be guaranteed. As a result, the framework is designed to degrade gracefully by selecting the least hazardous available option rather than failing abruptly. Within the optimization, tunnel boundaries are treated as soft constraints, while longitudinal limits are enforced more strictly. If the *Single Selector* cannot identify a fully feasible tunnel, a "forced-choice" strategy is applied to select the safest alternative. In all cases, a longitudinal constraint is activated to enforce a safe following behavior, reducing the ego velocity and increasing the available reaction time. However, this fallback mechanism does not eliminate all risks. In particular, the sudden activation of longitudinal constraints can be challenging when the vehicle is already operating at the handling limits, where tire saturation limits the achievable deceleration. In these scenarios, the objective shifts from collision avoidance to impact mitigation, reducing the severity of the event rather than guaranteeing its prevention.

Chapter 5

Experimental Validation

This section presents the results obtained from the proposed framework, ranging from different simulation environments to experimental on-track tests.

Validation methodology The development and validation of the planner followed a rigorous, iterative cycle designed to isolate specific behaviors before full-scale deployment. This methodology ensures robust performance through three distinct stages:

1. **Static Testing:** The first stage focuses on the validation of decision-making and tunnel generation. This phase is primarily a spatial check, verifying that the generated tunnels satisfy geometric constraints and logical requirements without the complexities of temporal evolution.

2. **Simulation:** The second stage validates the behavior of the system over consequent iterations. This phase serves as a temporal check, ensuring that the planner maintains stability and consistency as the scenario evolves dynamically.

3. **Real-World Validation:** The final stage involves the deployment of the full system on the physical vehicle.

To maintain system reliability, a specific debugging workflow is employed: anomalies observed during real-world operations are recorded and replicated using real-world data within the static testing environment. Once the issue is resolved locally, the solution is verified in the simulator before being re-deployed to the physical vehicle.

Evaluation Criteria Establishing objective metrics for autonomous racing is non-trivial. A fundamental and objective criterion is the verification of safety: the planner must successfully avoid static obstacles or overtake opponents without collisions.

However, assessing whether a specific maneuver was the *optimal* choice is significantly more complex, as no universal metric exists to determine strategic optimality in racing. For instance, determining whether an overtaking maneuver executed on the right was superior to a potential attempt on the left is subjective. The optimal choice depends not only on the available track space but also on the cross-dependent behaviors of all surrounding agents. Furthermore, the inability to complete an overtake, or a decision to abort a maneuver, should not necessarily be classified as a failure. In many high-speed scenarios, maintaining position to ensure stability and safety is the superior strategic choice compared to attempting a risky maneuver.

Experimental Scenarios To demonstrate the effectiveness of the proposed approach, we present a set of representative cases illustrating the decision-making process across varying scenarios. These cases are selected from both simulation environments and real-world tests conducted on various racing tracks.

A special focus is placed on the data acquired during the A2RL race 5.2.2, held in November 2025. This event is highlighted for three primary reasons:

- **Complexity:** It represents the most dynamic and challenging scenario in which the framework has been tested to date on the real vehicle.
- **Comprehensiveness:** It provides a broad overview of the planner's performance across diverse racing situations.
- **Maturity:** As the most recent event, it showcases the most refined version of the *Tunnels Framework*. Where specific features discussed in this thesis were not yet implemented during older tests, this distinction will be explicitly noted in the respective results.

The following sections are organized as follows: first, we describe the simulation environments used for preliminary testing; second, we provide a detailed analysis of the real-world experiments; and finally, we discuss the computational performance of the framework.

5.1 Simulation Experiments

Before going on track the adaptability to oval circuits and road courses has been demonstrated through multiple simulation platforms, with several opponents sharing the track. To obtain a comprehensive evaluation, the proposed framework was tested across three distinct simulators, each with specific strengths and limitations. Overall, these simulation environments confirmed the flexibility and robustness of the proposed approach across various levels of fidelity. Despite the different limitations and strengths of each simulator, the *Tunnels Framework* consistently demonstrated its ability to handle complex interactions, laying a strong foundation before testing on track. The effectiveness of the proposed method in managing multi-agent interactions under varied conditions can be observed in the accompanying simulations at https://atoschi.github.io/tunnels-framework/. Reporting many scenarios would be cumbersome and limited compared to the video evidence; this why in this section we focus on describing the simulation environments used for testing and on the process, without going into specific scenarios.

5.1.1 Multibody simulator

The first simulation environment is UNIMORE Racing's proprietary multi-body simulator. It provides highly accurate vehicle dynamics, but does not include interactive behaviors: opponent vehicles follow predefined trajectories, do not react to the ego vehicle, and collisions are not modeled.

Beyond the high fidelity of the vehicle dynamics, a key advantage of this simulator is the complete control over the racing scenario. This allows specific situations to be easily replicated and initialized directly in the desired configuration, rather than waiting for them to arise naturally during a simulation run. The scenario generation pipeline and the automatic simulations are described in detail in Lambertini et al., 2025.

Furthermore, the environment includes a specialized fault injection module capable of introducing deterministic perturbationss - such as sensor delays, measurement offsets, and communication latencies — to assess the stack's robustness under non ideal conditions. To maximize testing efficiency, the simulation pipeline is integrated into a Continuous Integration (CI) workflow on GitHub, enabling automated execution at speeds up to three times faster than real-time. This setup automatically generates detailed validation reports that analyze safety metrics, vehicle dynamics, and collision data immediately after each simulation run.

Since the opponents' trajectories and performance profiles are known in advance, they can be exploited to generate ideal, error-free predictions. Testing the planner under this assumption represents the first step of the simulation-based validation, as it allows the assessment of the decision-making and planning logic in isolation, without the confounding effects of prediction uncertainty. Once this phase is completed, the ideal predictions are replaced by the forecasting module described in the previous chapter, enabling a more realistic evaluation while simultaneously validating the prediction framework.

In this simulator, opponents simply follow a predefined racing line. The primary objective of this environment is therefore to validate overtaking maneuvers using the most realistic vehicle dynamics model available, and to further tune both the Motion Primitive Planner (MPP) and the Model Predictive Controller (MPC).

The simulator relies on detailed multi-body models of the Indylight AV-24 and the Superformula EAV24, developed within the team in Dymola using the VeSyMA-Motorsports libraries Raji et al., 2024a. An initial set of parameters is derived from data provided by the race organizers and component manufacturers, while remaining unknown parameters are estimated from available information on similar vehicles. The model is then progressively refined using experimental data collected during on-track tests. As new experimental data become available, the simulator is continuously updated and adjusted to maintain consistency with real-world vehicle behavior.

In addition to single-agent simulations with non-reactive opponents, the simulator also supports multi-agent configurations in which multiple autonomous stacks are executed on different machines and connected through Ethernet or Wi-Fi. In this setup, each vehicle is controlled by an independent instance of the full software stack, enabling interactive behaviors in which other agents react to the ego vehicle while respecting autonomous racing rules. This configuration provides a significantly more realistic interaction model compared to predefined opponents that ignore the ego vehicle. At the same time, since all agents share a very similar decision-making architecture, the resulting scenarios may exhibit recurring behavioral patterns, potentially leading to a large number of diverse but partially repetitive interaction cases.

5.1.2 Assetto Corsa

Assetto Corsa is a racing simulator developed by Kunos Simulazioni that is widely used for motorsport simulation, virtual testing, and vehicle dynamics research. In this work, extensive tests were carried out on the Yas Marina Circuit using the Super Formula EAV24 vehicle model through the interfaces described in Remonda et al., 2024. The interfaces enabling the connection between Assetto Corsa and external planners and controllers are open-source and publicly available.[1]

[1] https://assetto-corsa-gym.github.io/

Unlike the proprietary multi-body simulator, Assetto Corsa includes active opponents that react both to the ego vehicle and to each other, and it supports simulated collisions. These features make it particularly suitable for evaluating wheel-to-wheel interactions and highly aggressive maneuvers with small overtaking margins. In this environment, localization and vehicle detection are idealized and therefore not simulated. However, the absence of ideal opponent forecasting results in a more realistic setup, as the future actions of other agents are inherently uncertain in real racing scenarios.

Moreover, the virtual opponents are not constrained by the rules of autonomous racing and therefore exhibit more aggressive behavior, closer to that of human drivers. For the experiments, opponents were configured to behave aggressively, while their engine torque was slightly reduced. This adjustment facilitated interaction testing by easing the tuning of relative speeds and accelerations between vehicles.

The primary trade-off in this environment is the reduced fidelity of the vehicle model compared to the Dymola multi-body implementation. However, this is offset by the superior behavioral realism of the agents. Furthermore, although not utilized for the results presented in this specific section, the platform supports Human-in-the-Loop (HiL) integration, potentially allowing human drivers to control opponent vehicles to create highly complex and challenging scenarios.

5.1.3 Autoverse (race)

The last simulation environment used in this work is Autoverse, developed by Autonoma.ai.[2] Although it was not originally required by the team for development purposes, it was adopted because it served as the official simulator for the A2RL competition.

Autoverse is periodically updated using experimental data; however, its vehicle dynamics model is still less accurate than the proprietary multi-body simulator described previously. The platform offers two main operating modes: simulations with virtual opponents controlled directly by the simulator, and multi-user simulations. The former does not provide significant advantages compared to Assetto Corsa or the team's proprietary simulator, as virtual opponents do not react to the ego vehicle and collisions are not modeled. The primary strength of Autoverse lies instead in its infrastructure support for multi-user simulations through a shared server. This feature enables multiple autonomous racing teams to connect their vehicles to a common simulation environment, allowing continuous interaction among agents controlled by different planning and control stacks. In addition, race control signals, such as flags and start procedures, are simulated and transmitted to the vehicles, enabling realistic reproduction of race conditions. In 2025, a championship composed of four races was organized using this simulator, in which UNIMORE Racing participated and achieved an overall third-place ranking out of eleven teams. The results obtained during each Sprint Race and Race are summarized in Table 5.1. Sprint races were conducted in three separate groups based on qualifying results, and only the first six (or seven, depending on the event) teams from each group were admitted to the final race, starting from their finishing positions in the sprint. It is worth noting that, due to non-ideal qualifying performance and unfavorable starting grid positions in several events, overtaking maneuvers were frequently required. In this context, UNIMORE Racing's vehicle completed the highest number of successful overtakes during the championship.

[2] https://www.autonoma.ai/autoverse

TABLE 5.1: Autoverse Championship Results Summary

Circuit	Sprint Start	Sprint Final	Race Final	Positions Gained
Yas Marina Full GP	7*	7	4	3
Autonodrome**	5*	5	3	2
Suzuka	DNS	7	3	8
Yas Marina North	4*	5	3	1

* First in our group, so no possibility to do overtakes unless first being overtaken.
** Fictional track created by Autonoma.

The continuous interaction with softwares developed by other teams, each controlled by different planning and control architectures, provided a unique opportunity to evaluate the robustness and adaptability of the proposed solutions in highly dynamic and unpredictable scenarios. The diversity of strategies and behaviors exhibited by competing vehicles created a rich testing environment, pushing the limits of the *Tunnels Framework* and highlighting both its strengths and areas for improvement. Furthermore, this experience proved valuable for building confidence prior to on-track multi-vehicle testing and for familiarization with official race procedures, including pit lane operations, flag handling, and start protocols.

5.2 Real World Experiments

The efficacy of the *Tunnels Framework* was evaluated across distinct circuit topologies, including two tri-ovals, a standard oval, and a road course. This section presents telemetry data and chronological sequences from official competitive events, highlighting the system's performance in challenging scenarios. Additional material, including video recordings of the experiments and further implementation details, is available on the project webpage: https://atoschi.github.io/tunnels-framework/.

5.2.1 Indy Autonomous Challenge

The experimental validation presented in this section covers the Indy Autonomous Challenge campaign, specifically the events held at Indianapolis and Las Vegas. These two case studies were conducted using different iterations of the *Tunnels Framework*, utilizing distinct MPP and controller architectures as detailed in Toschi, Prignoli, and Bertogna, 2025. In both instances, the *egoLoc* module utilized was the earlier version described in Section 3.2.1.

Despite architectural differences, the tuning approach for both events remained consistently conservative. The primary objective during this phase was to validate the decision-making logic and guarantee safety in high-speed scenarios without risking collisions. Consequently, large static safety margins were imposed. Table 5.2 summarizes the configuration parameters employed in these environments. Given the race formats, which limit the maximum speed to incentivate overtaking maneuvers, the conservative setup was deemed appropriate to ensure safety without excessively compromising performance.

With this safety-first configuration established, we analyze two representative case studies to illustrate the system's performance.

TABLE 5.2: IAC Tunnels Framework Configuration Parameters

Category	Parameter	Indianapolis	Las Vegas
Safety Margins	Longitudinal (Front/Back)	15.0 m	15.0 m
	Lateral (Left/Right)	2.5 m	2.5 m
Track Limits	Boundary Margin Left	0.5 m	0.5 m
	Boundary Margin Right	1.0 m	1.0 m
Tunnel Logic	Min / Allowed Width	1.0 m / 3.0 m	1.0 m / 3.0 m
	Row Aggressiveness	N/A	0*
	ACC Reference	35-40 m**	35.0 m**

* Feature developed after IMS race.
** IMS time gap scheduling and hard constraint; LVMS static soft constraint.

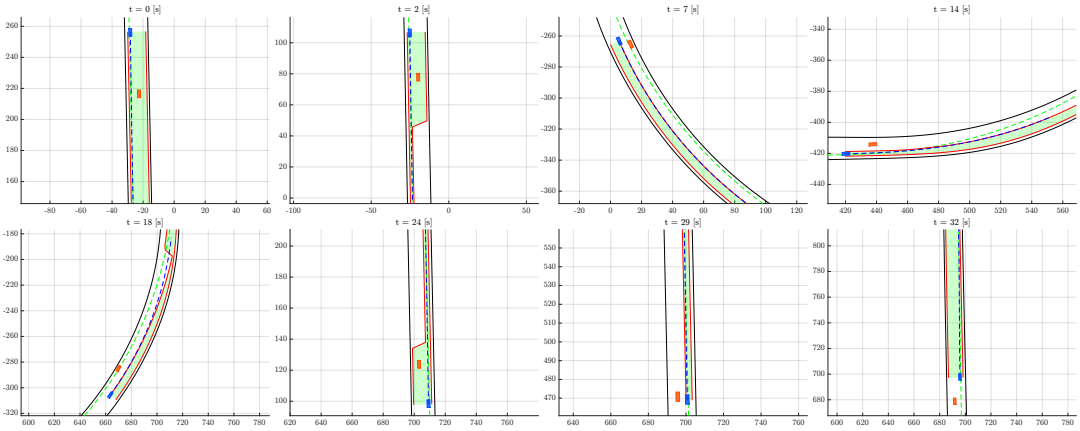


FIGURE 5.1: Time sequence of an high speed overtake at Indianapolis Motor Speedway during the Indy Autonomous Challenge. The ego vehicle approached the opponent at 267 km/h (t = 0 s), slowed down to 249 km/h due to the turn (t = 18 s), and finally completed the overtake at 276 km/h (t = 32 s).

Indianapolis

This first example highlights the fastest overtake performed during the 2024 Indy Autonomous Challenge final at the Indianapolis Motor Speedway. In this scenario, a high-speed vehicle is overtaken using the *Parallel Selector*. A total of six MPC instances were employed: three dedicated to longitudinal dynamics and three to lateral dynamics. Each pair operated on a different tunnel; two pairs relied on tunnels from the *Parallel Selector*, while one pair continuously performed ACC as explained in 3.5.1. After their parallel execution, a decision-making module selected which of the three generated trajectories to forward to an LQR controller. More details on this architecture and these results can be found in the journal Prignoli et al., 2026 and in the thesis Prignoli, 2026. Among the numerous overtakes performed during testing and competition, this case is particularly instructive from a decision-making standpoint. The maneuver is initiated but unintentionally aborted, as taking the outside line into the turn proved slower. It also highlights the planner's ability to respect track boundary constraints even near the vehicle's dynamic limits. The key phases of the overtake are illustrated in Fig. 5.1 and described as follows:

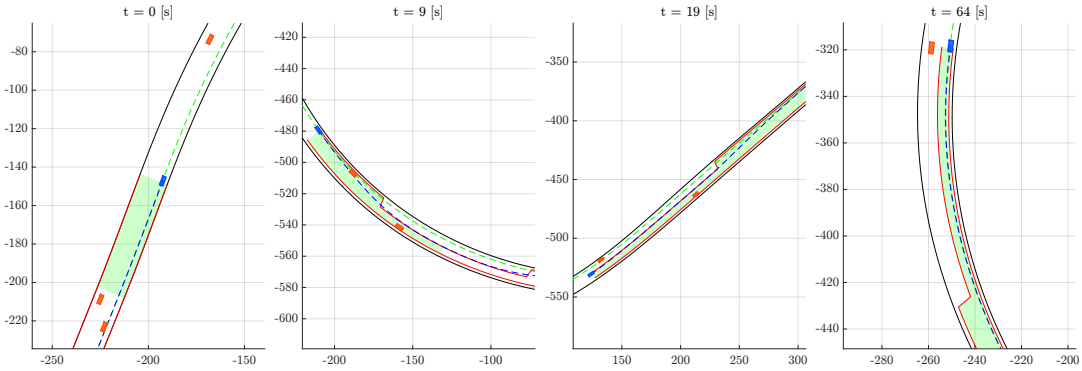


FIGURE 5.2: Time sequence of a double overtake at Las Vegas Motor Speedway during testing Indy Autonomous Challenge testing at CES 2025.

- $t = 0\ s$: The ego vehicle approaches on the main straight with no predicted interactions.
- $t = 2.5\ s$: The system predicts interactions and initiates an outside overtake instead of maintaining ACC, as it deems this approach feasible.
- $t = 7\ s$: The overtake continues into Turn 1 while closing in on the opponent.
- $t = 14\ s$: The opponent speed is higher through the turn, creating a gap; the vehicles remain within longitudinal safety margins, so the tunnel remains tightened.
- $t = 18\ s$: Both cars are still in Turn 2, with the gap slowly increasing. The ego vehicle is near the track boundary to respect constraints, staying away from the racing line but within the vehicle's dynamic limits.
- $t = 24\ s$: Entering the back straight, the car moved beyond the back safety margin, yet further interactions are predicted as the ego vehicle accelerates.
- $t = 29\ s$: The vehicles are side by side along the back straight, with the ego car proceeding to overtake.
- $t = 32\ s$: At the end of the back straight, the overtake is completed, and the entire track becomes available again.

Las Vegas

The second example utilizes the *Single Selector* architecture and, consequently, a single MPC instance (MPC-CURV, see Raji et al., 2023). While the absolute velocities are lower than in Indianapolis, this scenario presents higher complexity due to the interaction of four vehicles during a trial race for the Indy Autonomous Challenge at CES 2025.

A critical factor in this scenario is the *Push-to-Pass (P2P)* budget, which allows vehicles to trigger a temporary power boost followed by a cooldown period. Since both the ego vehicle and opponents may activate P2P at unpredictable intervals, longitudinal velocity predictions are subject to inherent stochastic errors. Figure 5.2 illustrates how the ego vehicle performs a double overtake in this uncertain context. The key phases of the overtake are:

- *t = 0 s*: Green flags were issued a few seconds earlier. The ego vehicle detects two opponents (almost side by side) ahead and one behind. No feasible tunnel is found, so it remains on the racing line and employs ACC to maintain safe longitudinal distances.
- *t = 9 s*: Approaching Turn 1 with P2P still active but ending soon, the ego vehicle identifies a slower opponent on the inner lane. The planner selects the right tunnel but imposes a longitudinal limit to follow the second opponent while overtaking the nearest one.
- *t = 19 s*: The previously predicted speeds prove inaccurate; by now, the ego vehicle was expected to have cleared the slower opponent. Exiting Turn 2 on the backstretch, the ego reactivates P2P, accelerating at $6\,\mathrm{m/s^2}$. Only one opponent remains relevant to the overtaking tunnel, as the other is moving faster and does not interfere.
- *t = 64 s*: After one full lap and several alternating P2P activations between attacker and defender, the ego vehicle completes the second overtake entering Turn 1.

5.2.2 Abu Dhabi Autonomous Racing League

This section presents the results obtained during the A2RL 2025 season. This event represents the most complex multi-agent scenario tested on track to date, serving as the real-world culmination of the virtual championship described in Section 3.5.2.

While a detailed chronicle of the competition is beyond the scope of this thesis, providing operational context is essential for interpreting the performance of the planner. The event progression was as follows:

- **Selection:** Participation was restricted to teams capable of passing rigorous quality gateways. Out of eleven initial participants, only six were admitted to the final multi-agent race.
- **Qualifying:** UNIMORE Racing demonstrated superior single-lap performance, ranking first in qualifying and securing pole position for the Sprint Race.
- **Sprint Race:** The field was split into groups, with two sprint races including three vehicles in each one. In the top-tier group, UNIMORE Racing finished second after being overtaken. Notably, the planning framework demonstrated strict rule compliance by yielding the right of way when required and robustness by successfully avoiding a collision via an off-track excursion. (Detailed analysis of this specific instance is omitted here for brevity).
- **Final Race:** Featuring six vehicles on track, UNIMORE Racing started from P2. The vehicle successfully executed an overtaking maneuver to lead the race for several laps before a collision occurred while attempting to lap for the second time a slower opponent exhibiting unexpected behavior. These critical scenarios are analyzed below.

The maneuvers executed during this event employed the most mature version of the *Tunnels Framework*, incorporating the latest *egoLoc* module described in Section 3.2.1. A key operational decision was the deactivation of the trajectory feasibility check introduced in Section 3.5.2, as it proved overly conservative for the tight

margins required in competitive racing. Instead, the system relied on the robustness of the low-level controller - validated through extensive simulation and preliminary track tests - to manage transient counter-steering actions and recover vehicle stability when aggressive trajectories were requested by the planner. The transition to the A2RL competition required a fundamental shift in planner tuning. Unlike previous oval events, where conservative configurations were preferred, the free-racing format demanded an aggressive setup to maximize overtaking opportunities. In this league, maximum speeds and accelerations are not explicitly limited; the primary regulatory constraint is that the leading vehicle (P1) has P2P disabled, while all other vehicles may activate it twice per lap. Under these conditions, overtaking is feasible only by exploiting very small margins.

Consequently, large static safety buffers were replaced by tight, dynamically adjusted constraints. As summarized in Table 5.3, overtaking margins were significantly reduced to shrink the effective safety bubble, and track boundary margins were set to 0.0 m, allowing the planner to fully exploit the available road width. In addition, the ACC logic was refined through a velocity dependent Look-Up Table (LUT), enabling substantially closer following distances. These aggressive parameters were selected based on extensive simulation campaigns 5.1.3 and prior experimental validation, and they represent the only viable configuration for enabling overtakes given the limited advantage provided by P2P. For the same reason, the use of ACC was deliberately minimized to avoid unnecessary loss of time: the follow mode is engaged only when strictly required, such as during formation laps or when an imminent collision is predicted. In all other situations, the combination of high *row aggressiveness* and reduced margins ensures that an *allowed* tunnel is almost always available, thereby avoiding reliance on ACC. The *Single Selector* and a MPP-MPC architecture similar to the one presented in Ayoub Raji, 2024 were employed for this event.

TABLE 5.3: A2RL Tunnels Framework Configuration Parameters

Category	Parameter	Value / Range
Safety Margins	Long. Standard (Min–Max)	4.0 – 5.0 m
	Lat. Standard (Min–Max)	0.8 – 1.2 m
	Long. Critical Limit (Back–Front)	0.5 / 2.0 m
	Lat. Critical Limit (Left–Right)	0.5 m
Track Limits	Boundary Margin (L/R)	0.0 m
Tunnel Logic	Min / Allowed Width	2.1 m / 2.2 m
	Row Aggressiveness	4
	ACC Distance[1]	6.0 – 11.0 m

[1] Dynamic Look-Up Table based on velocity (23 m/s to 55 m/s).

Race - Overtaking for P1

The first analyzed scenario focuses on the battle for the lead position. The race commenced with a rolling start, where initial performance was dictated by tire thermodynamics rather than raw power, as even the leader had Push-to-Pass (P2P) enabled. In this phase, lap times are significantly higher than nominal values and improve non-linearly as tire temperature increases. The rate of this progression depends on

the autonomous stack's strategy, which must trade off the ability to push on cold tires against the risk of exceeding adhesion limits.

Capitalizing on superior tire management and a strategic advantage regarding P2P usage, the ego vehicle managed to close the gap to the leader (TUM Autonomous Motorsport) during the second lap. The overtaking maneuver was preceded by a decision-making sequence in Turn 5, triggered by a false-positive interaction prediction. This sequence provides a valuable case study in planner robustness against forecasting uncertainty.

The temporal evolution of the event, from corner entry to exit, is illustrated in Figures 5.3–5.65.5—-5.4[3] and detailed below:

- $s = 1121.1\,m$ – Fig. 5.3, first snapshot: Approaching Turn 5, the ego vehicle starts predicting interactions and selects an inside overtake, as the *5 s* horizon indicates it will be ahead of the opponent at the apex.
- $s = 1186.2\,m$ – Fig. 5.3, second snapshot: As the vehicles approach the turn entry, the ego vehicle remains on the racing line and switches to an outside overtake tunnel. Updated forecasting shifts the predicted interaction forward in time, with the ego vehicle now predicted to be behind the opponent even at the end of the *6 s* horizon.
- $s = 1259.8\,m$ – Figs. 5.4 and 5.5: The ego vehicle is slightly off the racing line, but the *egoLoc* state remains *back*, as the vehicle is still within the blue cone (Fig. 5.4, first snapshot). Consequently, both overtaking tunnels are available. In the Frenet frame (Fig. 5.5), both tunnels are classified as *allowed*. The left tunnel is clamped to its minimum width but remains feasible because the predicted proximity to the opponent grants right of way. The right tunnel is selected due to the cost function: although the inner option requires less deviation from the racing line, the larger available area of the outer tunnel yields a lower overall cost.
- $s = 1276.5\,m$ – Fig. 5.6: The ego vehicle is now fully committed to the outside tunnel, with a clear lateral deviation and heading difference relative to the racing line. The predicted interactions shift forward once again, confirming that earlier interaction estimates were inaccurate.
- $s = 1303.3\,m$ – Fig. 5.4, second snapshot: Once committed to the outside line, switching back to the inside is prevented both by the accumulated cost and by the *egoLoc* state transitioning to *Right*, which renders the left tunnel *not allowed*. Although the hysteresis zone is not shown, its half-car-width extent is sufficient to place the ego vehicle's center of gravity fully outside it.
- $s = 1328.0\,m$ – Fig. 5.3, third snapshot: The vehicles are now closer while accelerating out of the corner. Predicted interactions move further ahead in time, and the tunnel is modified near corner exit.
- $s = 1380.7\,m$ – Fig. 5.4, third snapshot: As interactions shift forward, the ego vehicle returns to the racing line. With the vehicles aligned and the *egoLoc* state reverting to *back*, both tunnels become *allowed* again.

[3] In all plots shown in this section, the ego vehicle is represented in blue, the first interacting opponent in red, and the second interacting opponent in orange. When present, dot markers along predicted trajectories indicate 1 s intervals, facilitating interpretation of the longitudinal evolution. In global-frame plots, the color of predicted trajectories gradually fades with increasing prediction horizon to visually convey growing uncertainty.

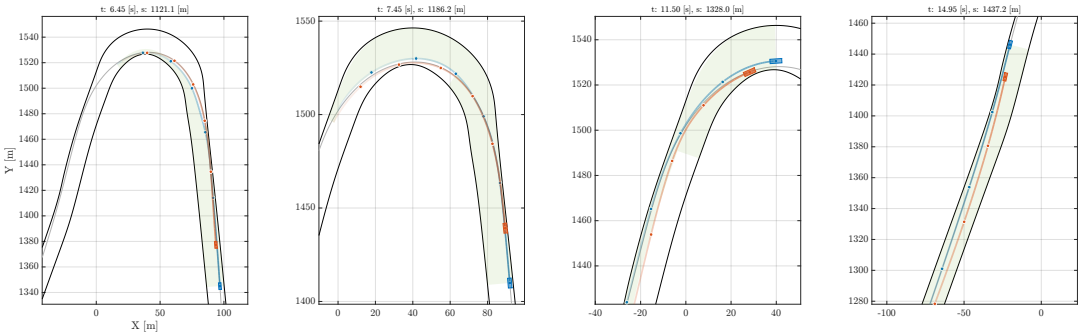


FIGURE 5.3: Time sequence of a predicted interaction at T5 of Yas Marina Circuit during the second lap of A2RL 2025 race.

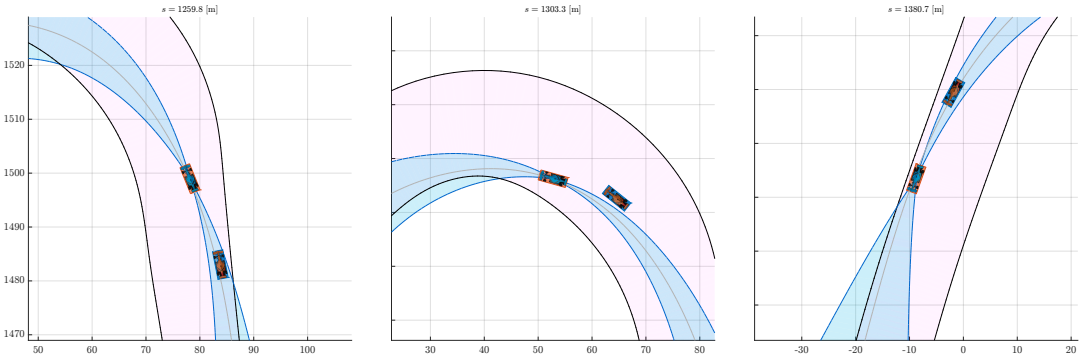


FIGURE 5.4: Time sequence of *egoLoc* evolution in T5 of Yas Marina Circuit during the second lap of A2RL 2025 race.

- $s = 1437.2\,m$ – Fig. 5.3, fourth snapshot: After exiting the corner, both vehicles accelerate along the straight. The ego vehicle activates P2P, and since no interactions are predicted within the horizon, the full track width becomes available.

This sequence illustrates how the *Planning Framework* effectively handles inaccurate interaction predictions while still executing a smooth and stable maneuver through continuous replanning. In addition to forecasting errors, the ego prediction does not account for the curvature induced speed loss of the outside trajectory, which is inherently slower than the racing line. Although remaining on the racing line without predicting interactions would have been optimal in hindsight, the planner's conservative behavior driven by overtaking margins acting as safety buffers—causes a temporary deviation to account for uncertainty.

Nevertheless, as shown in Fig. 5.6, the combination of small margins and smooth trajectory adaptation limits the time loss to a minimum. The resulting actuation signals remain smooth, and the ego vehicle ultimately benefits from an early apex line, braking and accelerating earlier than the opponent.

The second phase of the overtaking maneuver takes place at the end of the back straight and during the entry into Turn 6, where the actual pass for P1 is completed. The time sequence from corner exit to overtake completion is illustrated in Figures 5.7–5.9–5.8–5.10 and described below:

- $s = 2017.1\,m$ – Fig. 5.7, first snapshot: As in previous cases, interaction predictions are not perfectly accurate. The ego vehicle predicts that its center of gravity will be ahead of the opponent within $3\,s$ and selects a right-side overtake. Although the racing line is nearly straight, the track boundaries curve

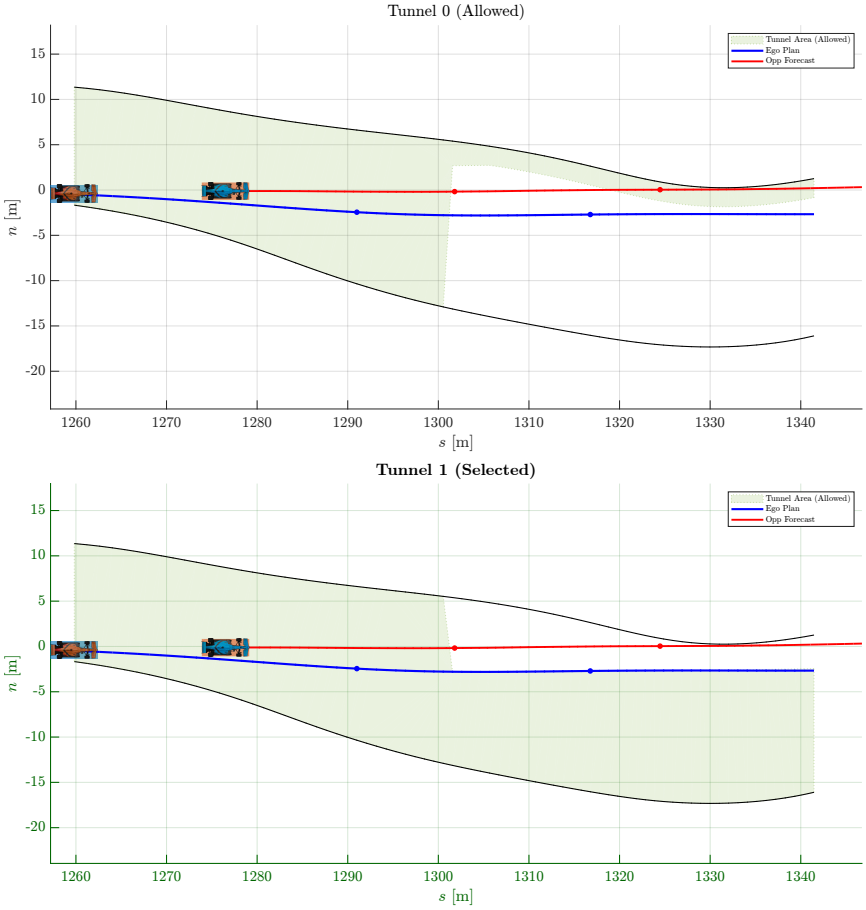


FIGURE 5.5: Frenet representation of all the calculated tunnels during the predicted interaction at T5 of Yas Marina Circuit during the second lap of A2RL 2025 race. The selected tunnel plot is highlighted in bold with green axes.

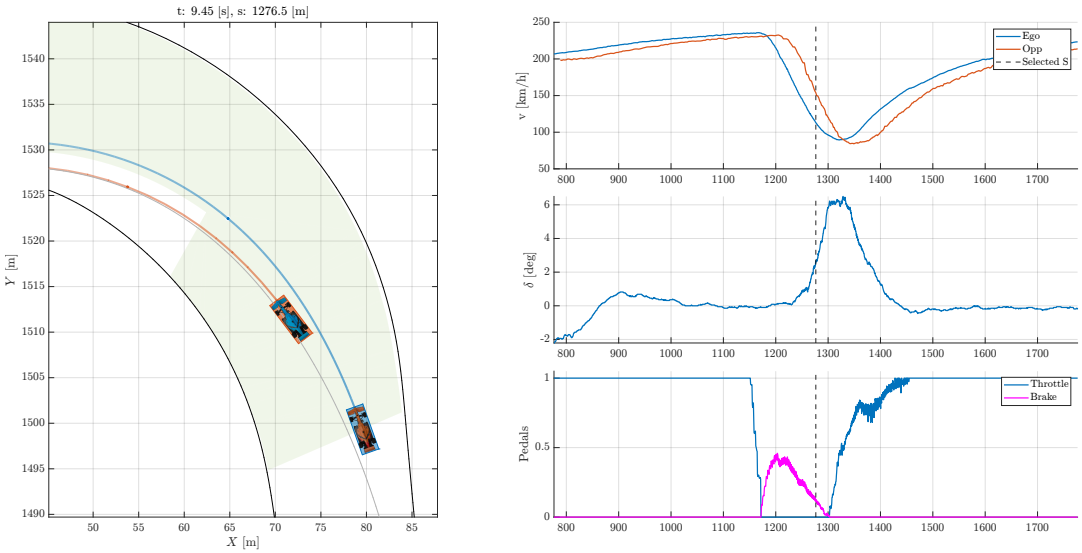


FIGURE 5.6: Vehicle speeds and ego vehicle actuation commands along a zoom in a specific point of the selected tunnel and predicted trajectories. Data refers to the predicted interaction at T5 of Yas Marina Circuit during the second lap of A2RL 2025 race.

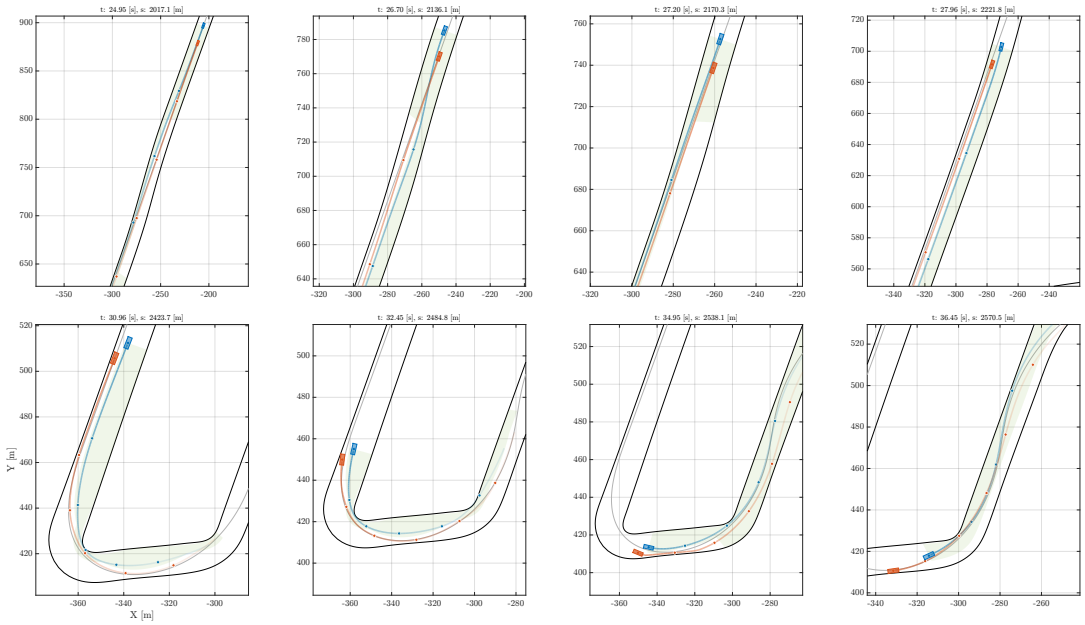


FIGURE 5.7: Time sequence of the overtake for the lead position (P1) at Yas Marina Circuit during the second lap of the A2RL 2025 race.

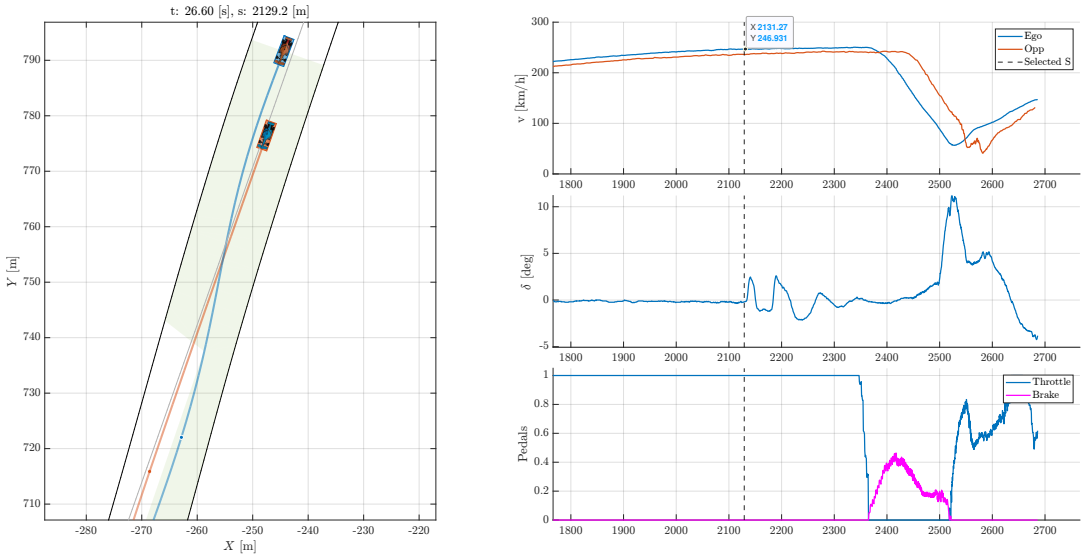


FIGURE 5.8: Vehicle speeds and ego-vehicle actuation commands, zoomed on a critical segment of the selected tunnel and the corresponding predicted trajectories, during the overtake for P1 at Yas Marina Circuit.

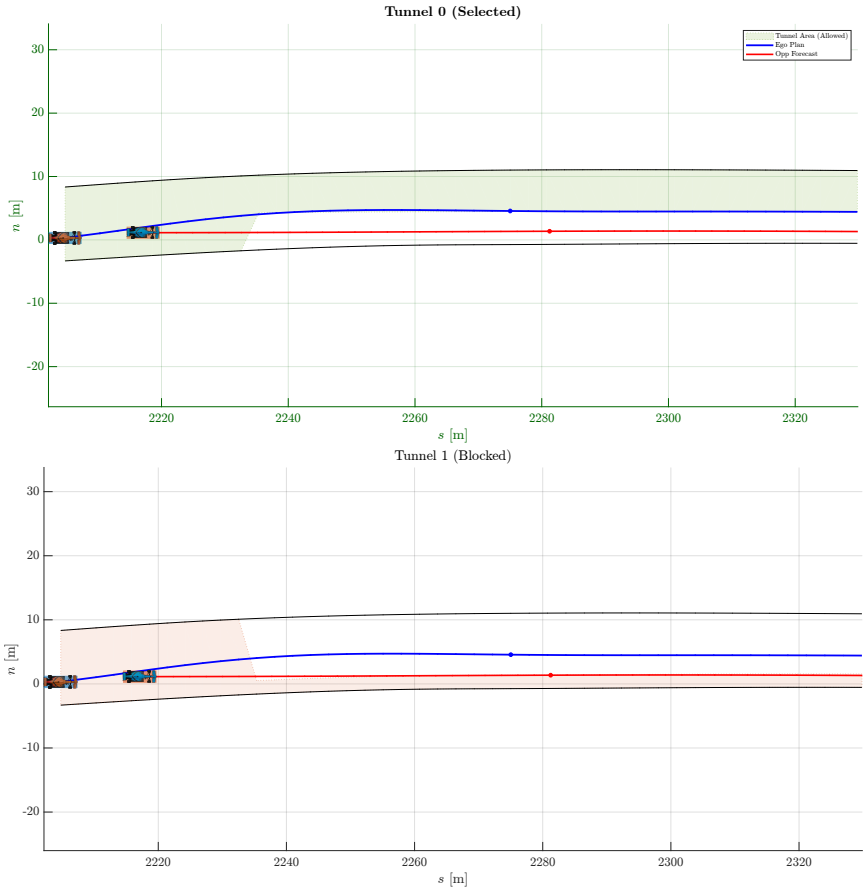


FIGURE 5.9: Frenet-frame representation of all calculated tunnels during the overtake for P1 at Yas Marina Circuit. The selected tunnel is highlighted in bold, with green axes.

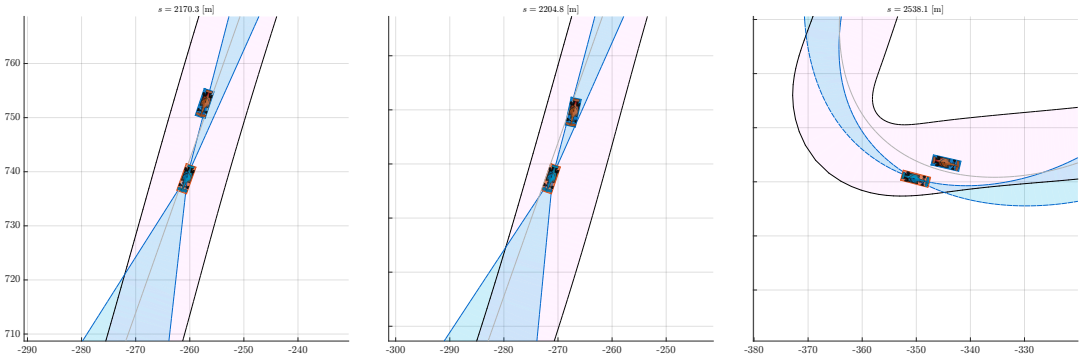


FIGURE 5.10: Time sequence of the *egoLoc* state evolution during the overtake for P1 at Yas Marina Circuit.

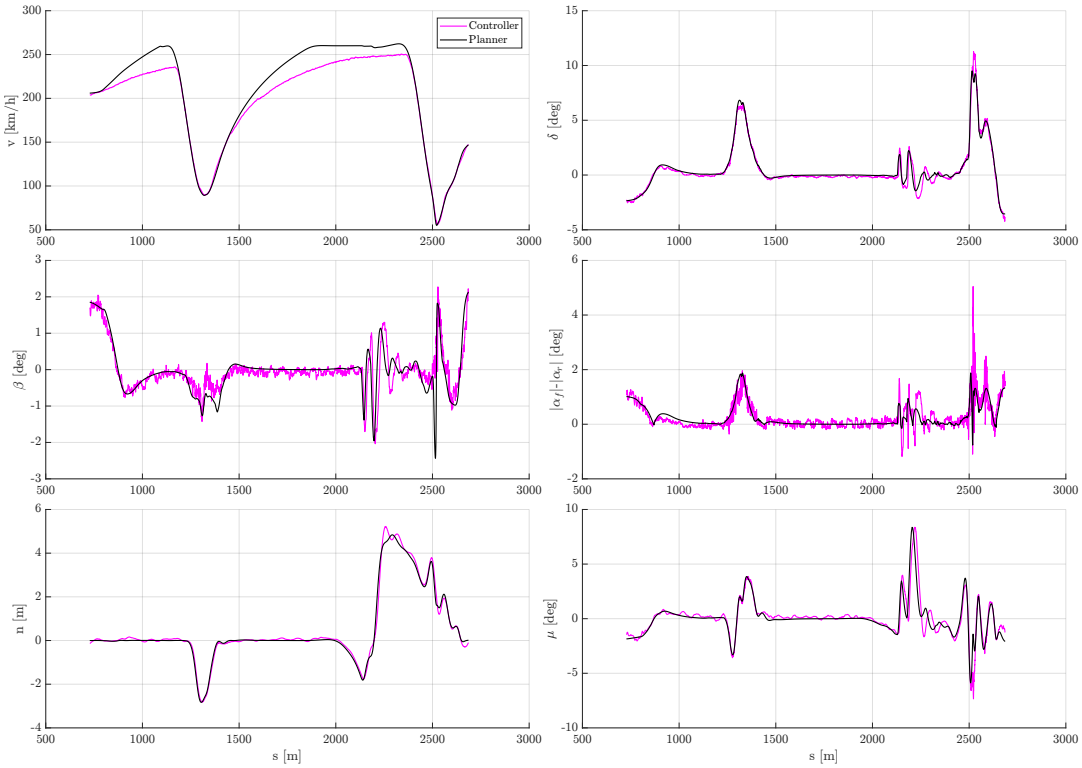


FIGURE 5.11: Telemetry data recorded during the overtake for P1, including vehicle speed, steering input, throttle, and braking commands.

slightly to the left, causing the racing line to shift toward the right near the braking point for Turn 6. Predicting an early interaction therefore implies being ahead before the track curvature becomes significant.

- $s = 2129.2\,m$ – Fig. 5.8: Approximately 100 m later, the ego vehicle has already deviated laterally to the right. Updated predictions now indicate that the ego vehicle will remain behind the opponent at the point where the racing line approaches the right track boundary. Consequently, the selected tunnel switches to the left one, which was already *allowed* but more expensive due to its greater deviation from the racing line. The *egoLoc* state remains *back*, but the right tunnel becomes *not allowed* because a collision is predicted, even though the right of way could have been obtained.

- $s = 2136.1\,m$ – Fig. 5.7, second snapshot: The ego vehicle now steers significantly to the left, fully committing to the left tunnel. It closes the gap to the opponent while remaining slightly offset laterally.

- $s = 2170.3\,m$ – Fig. 5.7, third snapshot, and Fig. 5.10, first snapshot: For a few planning iterations, the tunnel selection switches back to the right due to the *egoLoc* state. As shown, the vehicle is exactly at the boundary of the hysteresis zone, causing the *egoLoc* to briefly transition to *Right*. This renders the left tunnel *not allowed*. With both tunnels temporarily *not allowed*, the forcing logic is activated and prioritizes the *egoLoc* state, enforcing the right tunnel selection. Although for a short time, this results in a brief steering straightening.

- $s = 2204.8\,m$ – Fig. 5.10, second snapshot, and Fig. 5.9: Both vehicles are now closer to the right track boundary, becoming more laterally aligned. The *egoLoc*

state reverts to *back*, making the left tunnel *allowed* again. In the Frenet representation, the left tunnel is selected as the only feasible option; the right tunnel is clamped to its minimum width and remains *not allowed*, as a collision would occur within the corridor.

- $s = 2221.8\,m$ – Fig. 5.7, fourth snapshot: The ego vehicle fully recommits to the left tunnel, steering decisively left to finalize the overtake.
- $s = 2423.7\,m$ – Fig. 5.7, fifth snapshot: Predicted interactions move further ahead in time, with the vehicles expected to be side by side at the turn apex. The ego vehicle maintains the left line while staying as close as possible to the racing line.
- $s = 2484.8\,m$ – Fig. 5.7, sixth snapshot: Interactions are predicted even further ahead, in this case overestimating the opponent's speed in hindsight, contrary to earlier scenarios.
- $s = 2538.1\,m$ – Fig. 5.7, seventh snapshot, and Fig. 5.10, third snapshot: The ego vehicle is now ahead of the opponent. The *egoLoc* state was already *left* for many iterations, but this illustration shows how the cone-based logic naturally stabilizes side selection in tight cornering situations, without relying on cost-function comparisons.
- $s = 2570.5\,m$ – Fig. 5.7, eighth snapshot: The overtake is completed. The full track width is not yet available, as the opponent remains within the longitudinal right-of-way distance and the rules margin is still enforced on the right.

The initial change in tunnel selection (from right to left) is caused by a mismatch in longitudinal predictions. To improve safety, the forecasting module clamps the opponent's maximum speed to 230 km/h, which is slightly lower than the opponent's actual top speed (approximately 245 km/h). This conservative assumption ensures that interactions are always predicted at high speed, preventing alignment (slipstream) before braking. However, it also introduces a systematic error at top speed, which leads to the initial selection of the right tunnel. The subsequent rapid switching (left–right–left) is instead directly related to the *egoLoc* state and is a consequence of the tight margins and aggressive tuning, rather than a malfunction. Although the maneuver may appear abrupt, telemetry data in Fig. 5.11 clearly show that the controller tracks the planned trajectory without instability. The understeer degree remains limited, full throttle is maintained throughout the maneuver (Fig. 5.8), and the overall time loss is minimal. If a performance degradation is observed is attributable solely to the increased path length relative to the racing line, rather than to any speed reduction induced by braking.

Race - Overtaking Lapped Car

After leading the race for several laps, the ego vehicle approached a slower opponent to lap it for the first time. In this scenario, race control regulations require the lapped vehicle to facilitate the pass by disabling Push-to-Pass (P2P), limiting its speed to 200 km/h, and moving to an outer line by maintaining approximately 1 m from the right track boundary. Conversely, P2P is enabled for the faster vehicle. The following analysis focuses on the lapping maneuver occurring between the exit of Turn 7 and the exit of Turn 8.

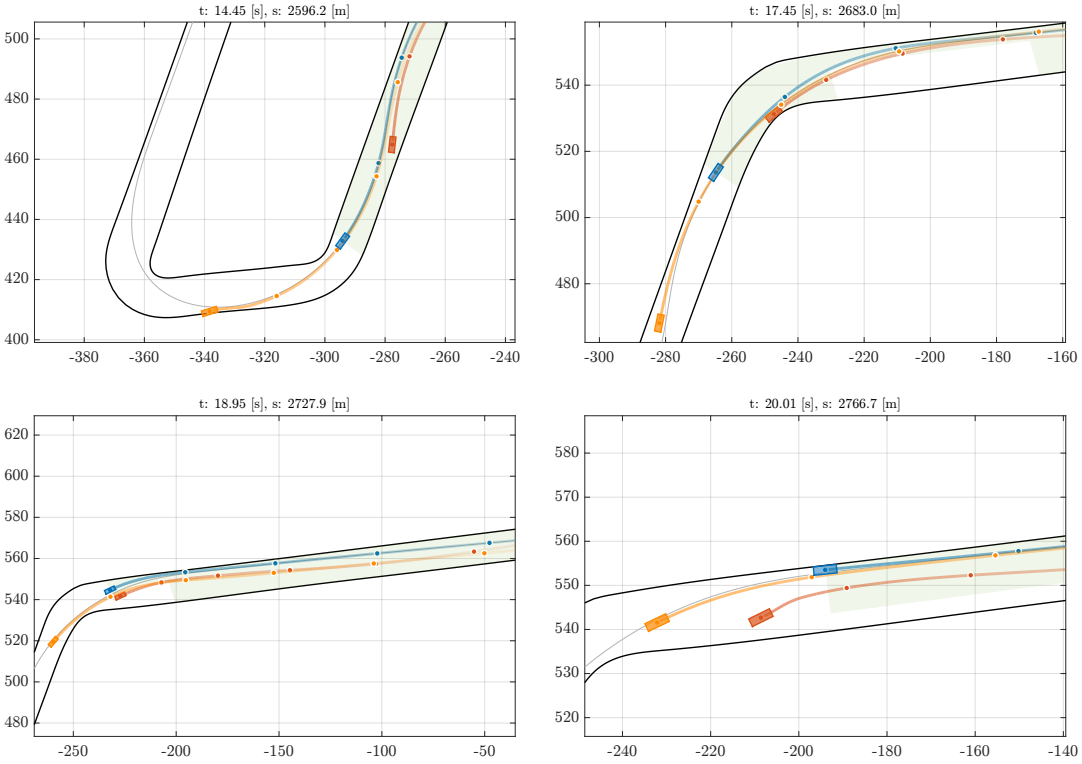


FIGURE 5.12: Time sequence of the lapping maneuver between Turns 7 and 8 at Yas Marina Circuit during the A2RL 2025 race.

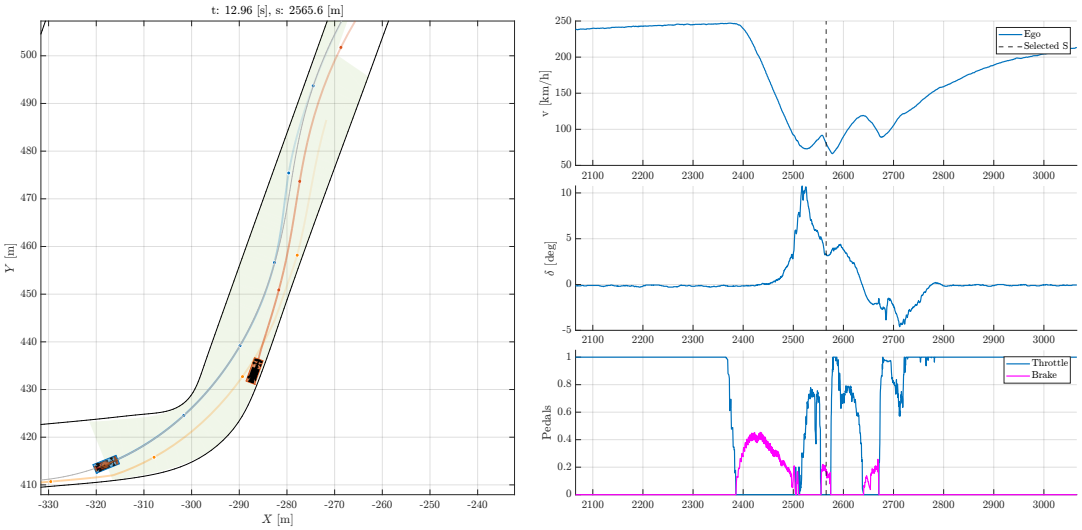


FIGURE 5.13: Zoomed view of vehicle speeds, actuation commands, and predicted trajectories during the lapping maneuver between Turns 7 and 8.

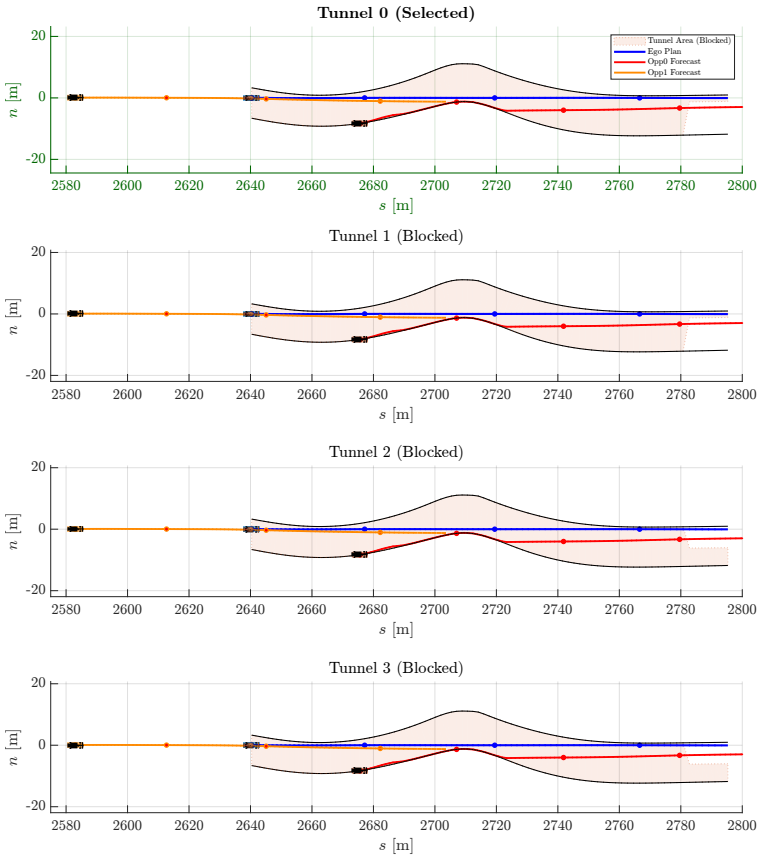


FIGURE 5.14: Frenet-frame representation of the tunnels generated during the lapping maneuver between Turns 7 and 8.

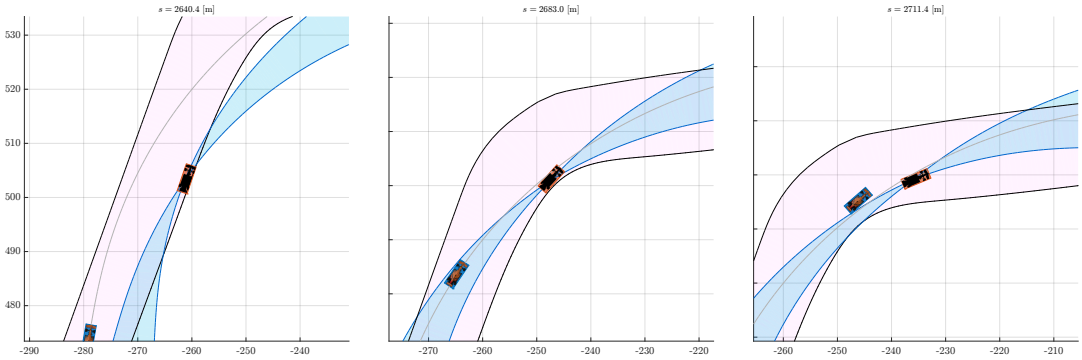


FIGURE 5.15: Evolution of the *egoLoc* state during the lapping maneuver between Turns 7 and 8.

- $s = 2565.6\,m$ – Fig. 5.13: The ego vehicle is positioned between two opponents: a slower vehicle ahead and a faster one behind. The trailing vehicle is not relevant for tunnel generation, as it is sufficiently far and has not gained right of way. The slower opponent ahead is predicted to merge toward the racing line and accelerate, since the forecasting module does not account for lapping rules. Based on this prediction, no *allowed* tunnel is available. Consequently, Adaptive Cruise Control (ACC) is activated to maintain a safe distance. A left tunnel is selected but forced, with a longitudinal constraint imposed due to its infeasibility.
- $s = 2596.2\,m$ – Fig. 5.12, first snapshot: After a short distance, the speed reduction induced by ACC causes predicted interactions to disappear. The ego vehicle therefore resumes acceleration.
- $s = 2640.4\,m$ – Fig. 5.14 and Fig. 5.15, first snapshot: As the ego vehicle accelerates, interactions are again predicted beyond Turn 8. The *egoLoc* state is *left*, but tunnels overtaking on the left are too narrow and thus *not allowed*, as the slower opponent is predicted to merge toward the racing line. Although four tunnels are generated, they effectively form two pairs, since the trailing vehicle does not influence the corridor generation. ACC remains active, and a forced left tunnel with a longitudinal limit is applied.
- $s = 2683.0\,m$ – Fig. 5.12, second snapshot, and Fig. 5.14, second snapshot: The slower opponent is now traveling significantly below the ego vehicle's speed. The left tunnel becomes *allowed*, as the ego vehicle is predicted to gain right of way before the interaction point. The *egoLoc* state is *back*, and thus does not constrain the side selection.
- $s = 2711.4\,m$ – Fig. 5.15, third snapshot: The ego vehicle deviates from the racing line, and the *egoLoc* state switches to *left*, forcing this side selection.
- $s = 2727.9\,m$ – Fig. 5.12, third snapshot: The lapped vehicle does not maintain the expected 1 m offset from the right track boundary. The vehicles are now side by side, and ACC can no longer be engaged. The lapped car is visibly slower.
- $s = 2766.7\,m$ – Fig. 5.12, fourth snapshot: The lapping maneuver is completed. The ego vehicle moves ahead of the slower opponent, and only the rules margin remains enforced on the right side.

As a result of the combined effects of conservative forecasting and two ACC activations, the ego vehicle lost 2.157 s compared to the previous lap without traffic. This allowed the second position vehicle to gain 1.463 s, reducing the gap to the UNIMORE Racing car from 2.408 s to less than one second. This time loss was partially mitigated by a favorable lapping configuration later in the race, when another slower opponent was encountered on a straight after already moving to the right side of the track.

Explicitly recognizing lapped vehicles and constraining their predicted behavior to the outer line would have further reduced the time loss. An additional improvement could be achieved by refining the ACC logic to reduce acceleration rather than applying hard braking when marginally exceeding the safety distance. However, implementing such behavior is non trivial, as it requires reliably distinguishing situations in which minor safety distance violations are acceptable from those in which aggressive braking is strictly necessary to avoid collisions.

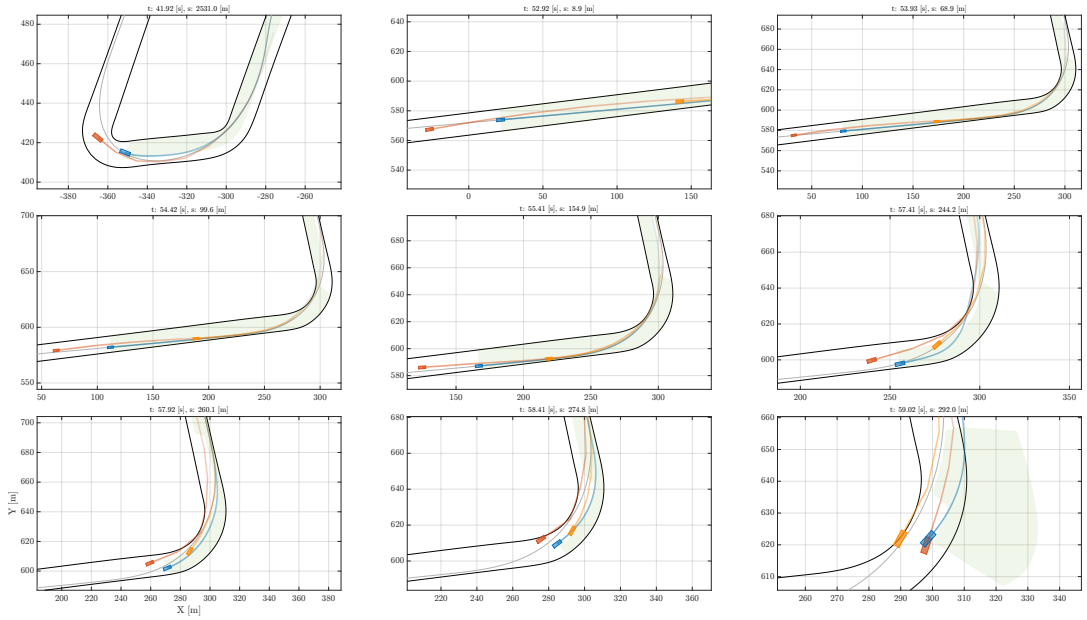


FIGURE 5.16: Time sequence of the crash scenario involving the ego vehicle and a lapped opponent at Yas Marina Circuit during the A2RL 2025 race.

Race - Crash

The ego vehicle and a following competitor were engaged in close racing, with the ego vehicle leaving the required rules margin on the outside of Turn 6, as the trailing car was sufficiently close to impose constraints but not close enough to initiate an overtaking maneuver. After leading the race for several additional laps, both vehicles again caught up with the same lapped opponent discussed in the previous section. In this instance, an abrupt change of direction followed by a sudden stop by the lapped vehicle in Turn 1 resulted in a collision with the ego vehicle.

The following analysis focuses on the crash sequence. It should be noted that complete log data were not available due to a system shutdown following the collision. The data presented here are therefore extracted from partial telemetry, recorded at a lower frequency. As a result, not all computed tunnels are available (only the selected one is), and some signals may appear less smooth. Nevertheless, the data remain sufficient to reconstruct and analyze the sequence of events.

- $s = 2531.0\,m$ – Fig. 5.16, first snapshot: The vehicles ahead are battling for the lead. The ego vehicle maintains the rules margin on the outside of Turn 6 due to the proximity of the trailing car, which has gained right of way. The lapped vehicle is not visible at this stage.

- $s = 8.9\,m$ – Fig. 5.16, second snapshot: The lapped vehicle becomes visible ahead, proceeding at a very low speed. The forecasting module predicts acceleration toward the nominal speed profile, and no interaction is detected yet.

- $s = 68.9\,m$ – Fig. 5.16, third snapshot: An interaction with the lapped vehicle is now predicted at the exit of Turn 1. The *Tunnels Framework* selects a left overtake, as the right side is too narrow.

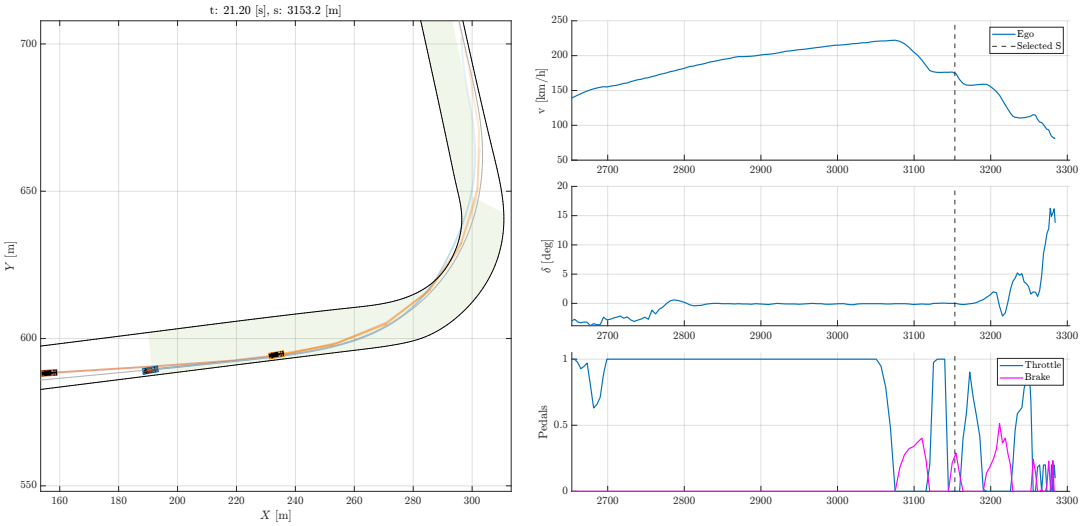


FIGURE 5.17: Zoomed view of the predicted trajectories and selected tunnel during the critical phase preceding the collision.

- $s = 99.6\,m$ – Fig. 5.16, fourth snapshot: As the predicted interaction shifts backward in time, the tunnel slightly widens. The left overtake remains selected but is classified as *not allowed*, and a longitudinal constraint is imposed.
- $s = 154.9\,m$ – Fig. 5.16, fifth snapshot: Predicted interactions shift forward again. The combination of ACC engagement by the ego vehicle and the extremely low speed of the lapped vehicle causes longitudinal predictions to oscillate. The left tunnel remains selected and becomes *allowed*.
- $s = 179.4\,m$ – Fig. 5.17: Both tunnels are again classified as *not allowed*, and the left option continues to be forced. Note that the figure reports $s = 3153.2\,m$, which corresponds to the same longitudinal position modulo the lap length (*one lap = 2973.8 m*).
- $s = 244.2\,m$ – Fig. 5.16, sixth snapshot: The lapped vehicle unexpectedly follows the nominal racing line instead of remaining on the outside, while the trailing car closes the gap and gains right of way. The selected tunnel still overtakes the lapped vehicle on the left, but a rules margin is now also enforced on that side, leaving only a very narrow corridor between the two constraints.
- $s = 260.1\,m$ – Fig. 5.16, seventh snapshot: All three vehicles are now in close proximity, with interactions predicted imminently. The available space collapses, and the planner switches to a right-side overtake of the lapped vehicle, while still maintaining a rules margin on the left due to the trailing car.
- $s = 274.8\,m$ – Fig. 5.16, eighth snapshot: The lapped vehicle suddenly steers to the right and brakes aggressively, coming to a near stop in the middle of the corner. No *allowed* tunnel remains available. The right-side option is retained but forced, with a longitudinal constraint applied.
- $s = 292.2\,m$ – Fig. 5.16, ninth snapshot: Hard braking induces tire locking and loss of directional control. The ego vehicle subsequently collides with the lapped vehicle.

This scenario demonstrates that, while the *Tunnels Framework* can handle highly critical situations, aggressive tuning combined with extreme and unpredictable behavior from other agents can still lead to collisions. Although this represents an edge case and the collision is not attributable to the ego vehicle's strategy, the planner must nonetheless manage such situations as robustly as possible.

One potential improvement would be to enhance the forecasting module's ability to detect severely degraded performance of other vehicles, thereby reducing prediction errors and providing additional reaction time. Another possibility is the introduction of an emergency mode in which tunnel generation is centered on an opponent CoG, temporarily disregarding forecasted trajectories when erratic behavior is detected. Finally, the current implementation does not explicitly consider the yaw angle of opponent vehicles, as this information is not provided by the perception module. While typically negligible, yaw information could be critical in extremely tight scenarios such as this one, particularly when a vehicle is significantly misaligned with the racing line. It should be noted that, in the plots of this analysis, opponent yaw angles are approximated from the initial segments of the predicted trajectories rather than from direct measurements.

5.3 Computational Performance

Computational Performance Analysis To validate the real-time feasibility of the proposed approach, we conducted extensive performance profiling of the stack on the specific onboard hardware used during the respective campaigns.

5.3.1 Hardware Platforms

Two different computing platforms were employed depending on the competition requirements:

- **Indy Autonomous Challenge (IAC):** The framework ran on a dSPACE AUTERA unit equipped with an Intel Xeon Processor D-2166NT (2.3 GHz).
- **Abu Dhabi Autonomous Racing League (A2RL):** The system was ran in a rugged HPC server, the Neousys RGS-8805GC. This unit is powered by an AMD EPYC 7003 "Milan" series processor (up to 64-core/128-thread).

The planner is designed to operate at a fixed frequency of 25 Hz (40 ms cycle time). It is critical to differentiate between the *Tunnels Framework* (Tunnel Generation + Decision Making) and the *Motion Predictive Planner* (MPP), which is responsible for the final trajectory optimization.

5.3.2 Tunnels Framework Profiling

We first analyze the computational cost of the Decision Making and Tunnel Generation modules.

IAC Configuration Table 5.4 reports the execution times recorded during the latest IAC event at LVMS on the dSPACE unit. The combined execution time remains consistently below 1 ms, occupying less than 3% of the available planning cycle.

TABLE 5.4: Tunnels Framework Profiling: IAC Configuration (dSPACE)

Agents	Samples	Median (ms)	Max (ms)	Min (ms)
0	15146	0.24	0.39	0.19
1	22402	0.36	0.61	0.19
2	8950	0.53	0.89	0.29
3	683	0.86	1.01	0.42

A2RL Configuration Table 5.5 presents results obtained during the A2RL Race using the Neousys RGS-8805GC. Despite the increased complexity and a higher density of agents (up to 5), the AMD EPYC processor managed the load efficiently. The median execution time with 5 agents was 1.26 ms, remaining negligible relative to the full cycle.

TABLE 5.5: Tunnels Framework Profiling: A2RL Configuration (Neousys RGS-8805GC)

Agents	Samples	Median (ms)	Max (ms)	Min (ms)
0	2893	0.567	0.868	0.335
1	12742	0.634	1.007	0.386
2	7910	0.696	1.110	0.444
3	2111	0.791	1.196	0.575
4	3018	0.988	1.354	0.732
5	50	1.260	1.693	0.786

5.3.3 Motion Predictive Planner (MPP) Profiling

Finally, we evaluate the performance of the MPP module, which performs the actual trajectory generation and optimization inside the selected tunnel. The data in Table 5.6 refers to the same A2RL Configuration on the Neousys hardware.

Unlike the tunnel generation, the MPP involves solving complex optimization problems, resulting in higher execution times. As shown, the median time fluctuates between 12.7 ms (0 agents) and 14.8 ms (5 agents). Even in the worst-case scenario, the sum of the Tunnels Framework ($\sim$1.3 ms) and MPP ($\sim$14.8 ms) totals approximately 16 ms. This demonstrates that the complete planning stack comfortably fits within the 40 ms real-time budget, leaving a safety buffer of over 50% that is more than enough for communication overheads and other auxiliary tasks; leaving room for future enhancements.

TABLE 5.6: MPP Profiling: A2RL Configuration (Neousys RGS-8805GC)

Agents	Samples	Median (ms)	Max (ms)	Min (ms)
0	2893	12.764	26.594	7.465
1	12742	13.670	25.008	7.414
2	7910	13.432	25.643	8.966
3	2111	13.129	22.608	9.376
4	3018	14.122	22.249	10.034
5	50	14.800	19.377	10.504

Chapter 6

Conclusion and Future Work

This dissertation addressed the problem of local motion planning for autonomous racing vehicles operating at the limits of handling in highly dynamic multi-agent scenarios. The transition from single-agent time-trial competitions to direct head-to-head racing has introduced a new layer of complexity, requiring algorithms that can simultaneously guarantee safety, respect strict racing regulations, and exploit the full dynamic potential of the vehicle.

The primary contribution of this work is the development and experimental validation of the *Tunnels Framework*. By decoupling the complex, non-convex problem of drivable area definition from the trajectory optimization, the proposed architecture enables real-time decision-making at frequencies exceeding 25 Hz. The same framework was even used to predict the behavior of the other agents. This approach was extensively validated not only in simulation but on two distinct full-scale racing platforms: the Dallara AV-21 in the Indy Autonomous Challenge (IAC) and the Dallara Super Formula SF23 in the Abu Dhabi Autonomous Racing League (A2RL). Extensive validation was carried out using a multi-stage methodology comprising static tests, high-fidelity simulation environments, and real-world on-track experiments. The framework was evaluated both in isolation and as part of a complete autonomous racing stack. Results demonstrated that the planner can safely and consistently generate feasible trajectories in complex racing scenarios, including high-speed overtakes, close wheel-to-wheel interactions, and congested race conditions. Experimental evidence from official competitions further confirmed the ability of the system to operate reliably under real racing constraints, achieving competitive performance while maintaining stability and safety.

6.1 Summary of Achievements

The research presented in this thesis has led to the following key achievements:

- **Modular Hierarchical Architecture:** We demonstrated that separating high-level strategic reasoning from low-level control (MPC) is a viable strategy for high-speed racing. This modularity allows for the integration of rule-based logic, essential for regulatory compliance, without compromising the mathematical rigorousness of the optimal control problem.
- **Dynamic Interaction Management:** The introduction of the *EgoLoc* logic, specifically the evolution from static regions to dynamic cones, provided a robust method for semantic spatial reasoning.

- **Tunable Aggressiveness:** Through the analysis of "Trust" parameters in Chapter 4, we showed how the planner's behavior can be continuously tuned between conservative (risk averse) and aggressive (performance oriented) profiles. This tunability proved crucial in adapting to different competition formats, from the structured passing battles of the IAC to the free-racing chaos of A2RL.
- **Real-World Robustness:** The proposed framework was designed with a system-level perspective, explicitly accounting for uncertainties arising from perception, localization, and prediction modules. Planning was formulated as the critical interface between noisy, delayed state estimates and aggressive low-level control, making robustness a central design objective. To this end, the framework integrates dynamic feasibility constraints derived from vehicle and tire models, adaptive grip limits, and regulatory rules governing right-of-way and interaction responsibility.
- **Coupling vehicle modeling and decision-making:** Beyond planning, this dissertation highlighted the tight coupling between vehicle modeling, state estimation, and decision-making in high-performance autonomous driving. Accurate estimation of lateral vehicle states and online adaptation of grip limits were shown to be essential enablers for aggressive yet controllable behavior. These aspects reinforce the importance of treating planning not as an isolated module, but as part of an integrated architecture where modeling, estimation, and control must co-evolve.

6.2 Limitations and Open Challenges

Despite the successful deployment in international competitions, the current framework exhibits limitations that open the door to further research:

6.2.1 Dependency on Explicit Forecasting

The current implementation relies heavily on the *Forecasting* module to generate opponent trajectories (via Racing Line Matching and MPPI). The planner treats these predictions as deterministic constraints (with safety margins). Consequently, if an opponent behaves erratically or performs a maneuver not captured by the forecasting module (e.g., a deliberate mistake or a loss of control), the *Tunnels* generator might produce infeasible or overly restrictive corridors.

6.2.2 Reactive vs. Interactive Planning

While the system is "interaction-aware" (it modifies its path based on others), it is not yet fully "interactive" in a Game Theoretic sense. The ego vehicle assumes the opponent will maintain its predicted trajectory and does not explicitly plan to influence the opponent's behavior (e.g., feinting an overtake to force a defensive line). The current approach is essentially a best-response strategy to a fixed prediction..

6.2.3 Model Mismatch at the Limit

As discussed in Chapter 4, the "Always Overtaking" assumption for the ego-prediction provides a safety buffer but introduces a kinematic mismatch. In scenarios of extreme tire degradation or reduced grip (e.g., rain or dirty track), the

disparity between the planner's optimistic geometric projection and the vehicle's actual capability can lead to tracking errors. While the online grip estimation helps, a tighter coupling between the strategic layer and the vehicle dynamics model is desirable.

6.3 Future Research Directions

The work presented in this thesis lays the foundation for several promising research avenues:

6.3.1 Learning-Based Intent Prediction

Future iterations of the stack will focus on replacing heuristic forecasting with data-driven approaches. By leveraging Deep Learning models trained on the growing dataset of autonomous racing interactions (from IAC and A2RL), the system could predict not just the trajectory, but the intent of the opponent (e.g., aggressive defense or yielding). Integrating these probabilistic intents into the *Tunnels* logic would allow for more nuanced risk management.

6.3.2 Transfer to Passenger Vehicles

While developed for racing, the concepts of "dynamic drivable areas" and "stability at the limit" have direct applications to passenger vehicles. The ability to plan safe evasive maneuvers at high speeds is critical for Level 4/5 autonomous systems facing emergency scenarios on highways. Adapting the *Tunnels Framework* to handle unstructured obstacles and varying road geometries could significantly enhance the active safety of commercial self-driving cars.

6.3.3 Expansion to New Platforms

The flexibility of the framework allows for its application beyond open-wheel racing. Preliminary tests on the *Roboracer* platform suggest that the logic can scale to different vehicle kinematics and sensor configurations. Future work will continue to validate this portability, aiming for a unified planning architecture capable of operating across the entire spectrum of autonomous mobility.

Bibliography

Ayoub Raji (2024). "Model Predictive Planning and Control for Autonomous Racing, from HPC to Embedded Platforms". PhD thesis. Parma, Italy: Università degli Studi di Parma.

Baumann, Nicolas et al. (Feb. 2025). "Predictive Spliner: Data-Driven Overtaking in Autonomous Racing Using Opponent Trajectory Prediction". In: *IEEE Robotics and Automation Letters* 10.2, pp. 1816–1823. ISSN: 2377-3766, 2377-3774. DOI: 10.1109/LRA.2024.3519878. URL: http://arxiv.org/abs/2410.04868 (visited on 02/19/2025).

Betz, Johannes et al. (2022). "Autonomous Vehicles on the Edge: A Survey on Autonomous Vehicle Racing". In: *IEEE Open Journal of Intelligent Transportation Systems* 3, pp. 458–488. ISSN: 2687-7813. DOI: 10.1109/ojits.2022.3181510. URL: http://arxiv.org/abs/2202.07008 (visited on 02/19/2025).

Bhargav, Jayanth et al. (July 2021). *Track Based Offline Policy Learning for Overtaking Maneuvers with Autonomous Racecars*. arXiv. DOI: 10.48550/arXiv.2107.09782. URL: http://arxiv.org/abs/2107.09782 (visited on 02/19/2025).

Buyval, Alexander et al. (Sept. 2017). "Deriving Overtaking Strategy from Nonlinear Model Predictive Control for a Race Car". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Vancouver, BC: IEEE, pp. 2623–2628. ISBN: 978-1-5386-2682-5. DOI: 10.1109/IROS.2017.8206086. URL: http://ieeexplore.ieee.org/document/8206086/ (visited on 02/19/2025).

Chung, Chanyoung, Hyunki Seong, and David Hyunchul Shim (Nov. 2024). *Learning from Demonstration with Hierarchical Policy Abstractions Toward High-Performance and Courteous Autonomous Racing*. arXiv. DOI: 10.48550/arXiv.2411.04735. URL: http://arxiv.org/abs/2411.04735 (visited on 02/19/2025).

Jank, Georg, Matthias Rowold, and Boris Lohmann (Sept. 2023). *Hierarchical Time-Optimal Planning for Multi-Vehicle Racing*. arXiv. DOI: 10.48550/arXiv.2309.06768. URL: http://arxiv.org/abs/2309.06768 (visited on 02/19/2025).

Jung, Chanyoung et al. (June 2021). *Game-Theoretic Model Predictive Control with Data-Driven Identification of Vehicle Model for Head-to-Head Autonomous Racing*. arXiv. DOI: 10.48550/arXiv.2106.04094. URL: http://arxiv.org/abs/2106.04094 (visited on 02/19/2025).

Jung, Chanyoung et al. (Mar. 2023). *An Autonomous System for Head-to-Head Race: Design, Implementation and Analysis; Team KAIST at the Indy Autonomous Challenge*. arXiv. DOI: 10.48550/arXiv.2303.09463. URL: http://arxiv.org/abs/2303.09463 (visited on 02/19/2025).

Lambertini, Giovanni et al. (2025). "Fast and Realistic Automated Scenario Simulations and Reporting for an Autonomous Racing Stack". In: *Proceedings of the 2025 IEEE Intelligent Vehicles Symposium (IV)*. Forthcoming.

Lee, Daegyu et al. (Sept. 2022). *A Resilient Navigation and Path Planning System for High-speed Autonomous Race Car*. arXiv. DOI: 10.48550/arXiv.2207.12232. URL: http://arxiv.org/abs/2207.12232 (visited on 02/19/2025).

Li, Nan et al. (Sept. 2022). "A Real-Time NMPC Controller for Autonomous Vehicle Racing". In: *2022 6th International Conference on Automation, Control and Robots*

(ICACR). Shanghai, China: IEEE, pp. 148–155. ISBN: 978-1-6654-9873-9. DOI: 10.1109/ICACR55854.2022.9935523. URL: https://ieeexplore.ieee.org/document/9935523/ (visited on 02/19/2025).

Liniger, Alexander (2018). "Path Planning and Control for Autonomous Racing". PhD thesis. ETH Zurich. DOI: 10.3929/ETHZ-B-000302942. URL: http://hdl.handle.net/20.500.11850/302942 (visited on 02/19/2025).

Liniger, Alexander, Alexander Domahidi, and Manfred Morari (Sept. 2015). "Optimization-Based Autonomous Racing of 1:43 Scale RC Cars". In: *Optimal Control Applications and Methods* 36.5, pp. 628–647. ISSN: 0143-2087, 1099-1514. DOI: 10.1002/oca.2123. URL: http://arxiv.org/abs/1711.07300 (visited on 02/19/2025).

Musiu, Nicola (2026). "Vehicle Dynamics and Friction-Adaptive Model Predictive Control in Autonomous Racing: A Comprehensive Modeling Approach". PhD thesis. Modena, Italy: Università degli Studi di Modena e Reggio Emilia.

Musiu, Nicola et al. (2026). "A comprehensive benchmark of vehicle dynamics models for autonomous racing: a deep dive into MPC". In: *Vehicle System Dynamics* 0.0, pp. 1–33. DOI: 10.1080/00423114.2026.2618732. eprint: https://doi.org/10.1080/00423114.2026.2618732. URL: https://doi.org/10.1080/00423114.2026.2618732.

Na, Yuseung et al. (June 2024). "AutoKU: An Autonomous Driving System Design for the World's First Mass-Produced Vehicle in Multi-Vehicle Racing Environment". In: *2024 IEEE Intelligent Vehicles Symposium (IV)*. Jeju Island, Korea, Republic of: IEEE, pp. 1373–1380. ISBN: 979-8-3503-4881-1. DOI: 10.1109/IV55156.2024.10588679. URL: https://ieeexplore.ieee.org/document/10588679/ (visited on 02/19/2025).

Prignoli, Francesco (2026). "Predictive Motion Planning and Control for Autonomous Racing Under Regulations: From Compliance to Strategic Awareness". PhD thesis. Modena, Italy: Università degli Studi di Modena e Reggio Emilia.

Prignoli, Francesco et al. (2026). "Bridging Predictive Optimization and Real-World Racing: Motion Planning and Control for Head-to-Head Autonomous Racing". Submitted to IEEE Transactions on Intelligent Vehicles.

Raji, Ayoub et al. (Oct. 2022). "Motion Planning and Control for Multi Vehicle Autonomous Racing at High Speeds". In: *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*. Macau, China: IEEE, pp. 2775–2782. ISBN: 978-1-6654-6880-0. DOI: 10.1109/ITSC55140.2022.9922239. URL: https://ieeexplore.ieee.org/document/9922239/ (visited on 02/19/2025).

Raji, Ayoub et al. (Dec. 2023). "A Tricycle Model to Accurately Control an Autonomous Racecar with Locked Differential". In: *2023 IEEE 11th International Conference on Systems and Control (ICSC)*, pp. 782–789. DOI: 10.1109/ICSC58660.2023.10449744. URL: http://arxiv.org/abs/2312.14808 (visited on 03/01/2025).

Raji, Ayoub et al. (Jan. 9, 2024a). "er.autopilot 1.0: The Full Autonomous Stack for Oval Racing at High Speeds". In: *Field Robotics* 4.1, pp. 99–137. DOI: 10.55417/fr.2024004. URL: https://doi.org/10.55417/fr.2024004.

Raji, Ayoub et al. (2024b). "Er.Autopilot 1.1: A Software Stack for Autonomous Racing on Oval and Road Course Tracks". In: *IEEE Transactions on Field Robotics* 1, pp. 332–359. ISSN: 2997-1101. DOI: 10.1109/TFR.2024.3501252. URL: https://ieeexplore.ieee.org/document/10756753/ (visited on 03/01/2025).

Remonda, Adrian et al. (July 2024). *A Simulation Benchmark for Autonomous Racing with Large-Scale Human Data*. arXiv. DOI: 10.48550/arXiv.2407.16680. URL: http://arxiv.org/abs/2407.16680 (visited on 03/01/2025).

Stahl, Tim et al. (Oct. 2019). "Multilayer Graph-Based Trajectory Planning for Race Vehicles in Dynamic Scenarios". In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. Auckland, New Zealand: IEEE, pp. 3149–3154. ISBN: 978-1-5386-7024-8. DOI: 10.1109/ITSC.2019.8917032. URL: https://ieeexplore.ieee.org/document/8917032/ (visited on 02/19/2025).

Suresh Babu, Varundev and Madhur Behl (Jan. 2022). "Threading the Needle—Overtaking Framework for Multi-agent Autonomous Racing". In: *SAE International Journal of Connected and Automated Vehicles* 5.1, pp. 12–05–01–0004. ISSN: 2574-075X. DOI: 10.4271/12-05-01-0004. URL: https://www.sae.org/content/12-05-01-0004/ (visited on 02/19/2025).

Thrun, Sebastian et al. (Sept. 2006). "Stanley: The Robot That Won the DARPA Grand Challenge". In: *Journal of Field Robotics* 23.9, pp. 661–692. ISSN: 1556-4959, 1556-4967. DOI: 10.1002/rob.20147. URL: https://onlinelibrary.wiley.com/doi/10.1002/rob.20147 (visited on 02/19/2025).

Ticozzi, Andrea et al. (2023). "Optimal Smooth Polynomial Lane Change Generation for Autonomous Racing". In: *IFAC-PapersOnLine* 56.2, pp. 11834–11840. ISSN: 24058963. DOI: 10.1016/j.ifacol.2023.10.584. URL: https://linkinghub.elsevier.com/retrieve/pii/S2405896323009515 (visited on 02/19/2025).

Toschi, Alessandro (2023). "Integration of Model Predictive Control for Autonomous Racing Cars". MA thesis. Modena, Italy: Università degli Studi di Modena e Reggio Emilia.

Toschi, Alessandro, Francesco Prignoli, and Marko Bertogna (Oct. 19, 2025). *Modular Decision-Making and Drivable Areas for Multi-Agent Autonomous Racing*, p. 12441. 12435 pp. DOI: 10.1109/IROS60139.2025.11246897.

Toschi, Alessandro et al. (June 2024). "Guess the Drift with LOP-UKF: LiDAR Odometry and Pacejka Model for Real-Time Racecar Sideslip Estimation". In: *2024 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. DOI: 10.1109/IV54300.2024.10588691. URL: https://arxiv.org/abs/2405.05668.

Trumpp, Raphael et al. (Oct. 2024). "RaceMOP: Mapless Online Path Planning for Multi-Agent Autonomous Racing Using Residual Policy Learning". In: *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Abu Dhabi, United Arab Emirates: IEEE, pp. 8449–8456. ISBN: 979-8-3503-7770-5. DOI: 10.1109/IROS58592.2024.10801657. URL: https://ieeexplore.ieee.org/document/10801657/ (visited on 02/19/2025).

Wurman, Peter R. et al. (Feb. 2022). "Outracing Champion Gran Turismo Drivers with Deep Reinforcement Learning". In: *Nature* 602.7896, pp. 223–228. ISSN: 0028-0836, 1476-4687. DOI: 10.1038/s41586-021-04357-7. URL: https://www.nature.com/articles/s41586-021-04357-7 (visited on 02/19/2025).

Ögretmen, Levent et al. (Nov. 2022). "Smooth Trajectory Planning at the Handling Limits for Oval Racing". In: *Actuators* 11.11, p. 318. ISSN: 2076-0825. DOI: 10.3390/act11110318. URL: https://www.mdpi.com/2076-0825/11/11/318 (visited on 02/19/2025).

Ögretmen*, Levent et al. (Sept. 2023). "A Hybrid Trajectory Planning Approach for Autonomous Rule-Compliant Multi-Vehicle Oval Racing". In: *SAE International Journal of Connected and Automated Vehicles* 7.1, pp. 12–07–01–0007. ISSN: 2574-0741, 2574-075X. DOI: 10.4271/12-07-01-0007. URL: https://www.sae.org/content/12-07-01-0007 (visited on 02/19/2025).

Ögretmen, Levent et al. (June 2024). "Sampling-Based Motion Planning with Online Racing Line Generation for Autonomous Driving on Three-Dimensional Race Tracks". In: *2024 IEEE Intelligent Vehicles Symposium (IV)*, pp. 811–818. DOI: 10.1109/IV55156.2024.10588726. URL: http://arxiv.org/abs/2403.18643 (visited on 02/19/2025).