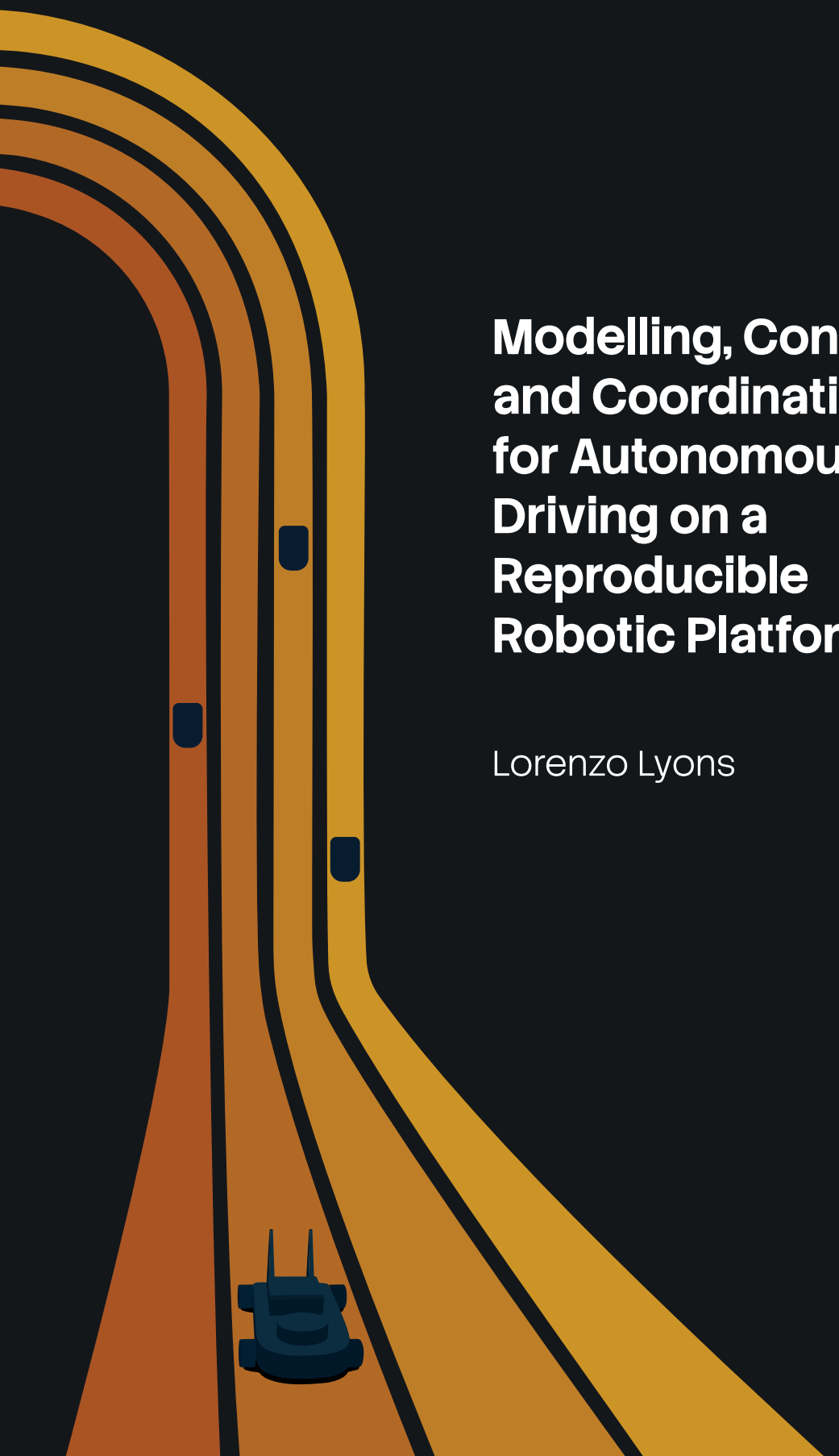


Modelling, Control, and Coordination for Autonomous Driving on a Reproducible Robotic Platform

Lorenzo Lyons

Modelling, Control, and Coordination for Autonomous Driving on a Reproducible Robotic Platform

Modelling, Control, and Coordination for Autonomous Driving on a Reproducible Robotic Platform

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology,
by the authority of the Rector Magnificus Prof. dr. ir. H. Bijl,
chair of the Board for Doctorates,
to be defended publicly on
Friday, June 12, 2026, at 10:00

by

Lorenzo LYONS

This dissertation has been approved by the promotors.

Composition of the doctoral committee:

Rector Magnificus,	Chairperson
Prof. dr. R. Babuška.,	Delft university of technology, *promotor*
Dr. L. Ferranti,	Delft university of technology, *promotor*
Independent members:	
Prof. dr. ir. R. Happee,	Delft university of technology
Prof. dr. ing. F. Braghin,	Polytechnic of Milan
Prof. dr. ing. J. Betz,	Technical University of Munich
Dr. ir. I.J.M. Besselink ,	Eindhoven University of Technology

Keywords: Vehicle dynamics, Model Predictive Control, Vehicle Platooning, Multi Agent Systems, Cyber-Physical Security
Printed by: Ridderprint
Cover: Cristina Ionne

ISBN 978-94-6518-349-7

An electronic version of this dissertation is available at

http://repository.tudelft.nl/.

"Felix, qui potuit rerum cognoscere causas."
Happy (is), who may know the cause of things.

Virgilius

Contents

Summary

Fully autonomous driving has long been promised, yet remains difficult to deploy at scale. Safe and efficient autonomy hinges on motion planning and control under tight constraints, realistic vehicle dynamics, and reliable cooperation between vehicles, often in the presence of delays, uncertainty, and adversarial communication. Progress in these areas depends on fast design–test cycles, but current validation workflows are polarized, high-fidelity simulation enables rapid iteration, yet struggles to reproduce road–tire interaction, actuator and sensor dynamics, and multi-agent network effects. Full-scale experiments capture these effects, but are costly, slow, and constrained by safety and regulatory requirements. This thesis addresses this gap by enabling physically grounded experimentation by designing a low cost and reproducible robotic platform, and by developing control and coordination methods for cooperative driving applications for more efficient traffic flow, as well as model based control strategies to handle safety critical maneuvers at the limits of handling.

Three limitations in the state of the art motivate the research objectives of this thesis. First, existing small-scale car-like platforms either remain relatively expensive for multi-agent experiments, depend on custom components that hinder reproducibility, or lack rigorous system-identification workflows needed for model-based control and sim-to-hardware transfer. Second, cooperative driving methods, for example platooning, often rely on timely attack detection or hard switching to degraded modes, leaving vulnerability windows and risking undesirable transients, moreover, adversarial robustness at the coordination and topology-management level, such as merge, split, rearrange, is less developed than low-level spacing control. Third, widely used MPCC formulations can become unreliable on highly curved paths because internal progress approximations drift under large deviations, which can degrade constraint handling, racing-oriented MPC variants frequently depend on offline priors, for example precomputed racelines, limiting transfer to emergency-like maneuvers where only local geometry is available. Accordingly, this thesis aims to develop: (I) a reproducible, low-cost experimental platform with a complete identification pipeline, (II) cooperative driving algorithms that preserve safety even under persistent or undetected communication attacks and remain feasible under actuator saturation, and (III) curvature-aware MPCC formulations that remain reliable on tight curvature and avoid dependence on offline priors in high-performance settings.

The thesis begins with the design and characterization of the Delft Autonomous-driving Robotic Testbed (DART). DART is a low-cost, reproducible, small-scale vehicle built on a commercial RC chassis and augmented with additional sensing and computational capabilities, including Lidar, an IMU, custom wheel encoders, and onboard computing. The platform preserves the essential features of full-scale vehicle dynamics, Ackermann steering, suspensions, electric motor, while remaining small enough to work with in laboratory settings. A central component is a comprehensive system identification procedure that yields reliable kinematic and vehicle dynamics models tailored to small-scale vehicles, along

with sub-models for motor force, friction, steering actuation, and tyre lateral forces. This modelling pipeline enables realistic testing and accurate model-based control on hardware.

The thesis then shifts focus to cooperative multi-vehicle systems by introducing a distributed, attack-resilient platooning framework. This work addresses two intertwined challenges: maintaining safety and formation integrity under malicious interference on the communication channels, and enabling coordinated platoon-level decisions such as merging, splitting, and rearranging. At the control level, the method combines sensor-based Adaptive Cruise Control with communication-based Cooperative Adaptive Cruise Control, while a safety filter ensures collision avoidance even when communicated acceleration data is corrupted. At the coordination level, a distributed topology-management strategy detects inconsistent information and isolates compromised vehicles by reorganizing the platoon. The approach is validated in simulation and experimentally on multiple DART vehicles, showing that safety and string stability are preserved despite communication attacks.

With this platform at hand, we then focus on designing a motion-planning strategy for urban driving. We investigate model predictive control and develop a Curvature-Aware Model Predictive Contouring Control (CA-MPCC) framework. Traditional MPCC formulations assume low curvature and rely on a lag-error term that couples progression along the reference path with lateral tracking. This complicates tuning and reduces reliability in tight curves. The CA-MPCC formulation resolves these limitations by explicitly accounting for curvature in the path geometry, removing the lag-error term entirely and simplifying both the cost structure and the tuning process. The method is validated in simulation and on the DART platform, demonstrating improved robustness, reduced parameter sensitivity, and reliable real-time performance even on highly curved trajectories.

Next, the thesis extends the curvature-aware MPCC methodology to high-performance domains with the rCA-MPCC framework for autonomous racing. Racing introduces additional challenges, such as operating at the limits of handling, rapid curvature changes, and strong coupling between dynamical states. The rCA-MPCC formulation augments CA-MPCC with a curvature-informed terminal cost and a compact reference-path representation that avoids dependence on precomputed race line information. This chapter also expands the physical vehicle model through actuator-dynamics identification and residual dynamic modeling using Gaussian Processes. Together, these advances improve prediction accuracy and enable robust high-speed control. We furthermore extend the car-racing formulation to aerial drone racing, bridging the gap between the two communities. Experiments on small-scale cars and simulations on quadrotor drones demonstrate faster, more consistent lap times and robustness across different dynamic model choices.

Finally, the thesis demonstrates how the developed platform and control tools can be applied to cooperative multi-robot tasks such as persistent monitoring and target detection. The chapter combines DART with the CA-MPCC controller to implement Lissajous-curve-based coordinated trajectories generated by time-inverted Kuramoto dynamics. The result is a distributed coordination strategy that guarantees collision avoidance and complete area coverage. Experimental validation confirms that the high-level coordination method and low-level CA-MPCC controller integrate smoothly on real hardware, even under disturbances and temporary agent failures.

Overall, this thesis contributes a unified framework for physically grounded experimen-

tation, accurate dynamics modeling, and robust control and coordination in autonomous driving. By developing a reproducible platform, constructing reliable models, and demonstrating advanced control strategies, from emergency maneuvers to cooperative platooning and multi-agent monitoring, it lowers the barrier to real-world validation and accelerates iteration cycles for autonomous driving research. The insights presented here support future work toward safer, more robust, and more efficient autonomous transportation systems.

Samenvatting

Volledig autonoom rijden wordt al lange tijd beloofd, maar blijkt nog steeds moeilijk op grote schaal te implementeren. Veilige en efficiënte autonomie berust op bewegingsplanning en regeltechniek onder strikte beperkingen, realistische voertuigdynamica en betrouwbare samenwerking tussen voertuigen, vaak in de aanwezigheid van vertragingen, onzekerheid en vijandige communicatie. Vooruitgang op deze gebieden vereist snelle ontwerp–testcycli, maar de huidige validatieworkflows zijn sterk gepolariseerd, hoogwaardig gesimuleerde omgevingen maken snelle iteratie mogelijk, maar slagen er niet goed in de interactie tussen wegdek en band, actuator- en sensordynamica en netwerkeffecten in multi-agent systemen realistisch te reproduceren. Experimenten op volledige schaal vangen deze effecten wel, maar zijn duur, traag en beperkt door veiligheids- en regelgevingseisen. Dit proefschrift overbrugt deze kloof door fysiek gefundeerde experimenten mogelijk te maken via het ontwerpen van een goedkoop en reproduceerbaar robotplatform, en door regel- en coördinatiemethoden te ontwikkelen voor coöperatieve rijtoepassingen voor een efficiëntere verkeersdoorstroming, evenals modelgebaseerde regelstrategieën om veiligheidskritische manoeuvres op de grenzen van het voertuiggedrag te beheersen.

Drie beperkingen in de huidige stand van de techniek vormen de motivatie voor de onderzoeksdoelstellingen van dit proefschrift. Ten eerste blijven bestaande kleinschalige, autoachtige platforms relatief duur voor multi-agent experimenten, zijn ze afhankelijk van maatwerkcomponenten die reproduceerbaarheid bemoeilijken, of missen ze rigoureuze workflows voor systeemidentificatie die nodig zijn voor modelgebaseerde regelingen en de overdracht van simulatie naar hardware. Ten tweede steunen coöperatieve rijmethoden, bijvoorbeeld platooning, vaak op tijdige detectie van aanvallen of op harde omschakeling naar gedegradeerde modi, waardoor kwetsbaarheidsvensters ontstaan en ongewenste transiënten kunnen optreden, bovendien is robuustheid tegen adversariële verstoringen op het niveau van coördinatie en topologiemanagement, zoals samenvoegen, opsplitsen en herschikken, minder ontwikkeld dan de laag-niveau afstandsregeling. Ten derde kunnen veelgebruikte MPCC-formuleringen onbetrouwbaar worden op sterk gekromde trajecten omdat interne voortgangsbenaderingen wegdrijven bij grote afwijkingen, wat de constraint-afhandeling kan verslechteren, en racegerichte MPC-varianten zijn vaak afhankelijk van offline voorkennis, bijvoorbeeld vooraf berekende racelijnen, waardoor overdracht naar noodsituatie-achtige manoeuvres, waarbij alleen lokale geometrie beschikbaar is, beperkt wordt. Daarom richt dit proefschrift zich op de ontwikkeling van: (I) een reproduceerbaar, goedkoop experimenteel platform met een complete identificatiepijplijn, (II) coöperatieve rij-algoritmen die veiligheid behouden, zelfs bij aanhoudende of niet-gedetecteerde communicatieaanvallen, en die haalbaar blijven onder actuatorsaturatie, en (III) krommingsbewuste MPCC-formuleringen die betrouwbaar blijven bij hoge kromming en niet afhankelijk zijn van offline voorkennis in high-performance toepassingen.

Het proefschrift begint met het ontwerp en de karakterisering van de Delft Autonomous-driving Robotic Testbed (DART). DART is een goedkoop, reproduceerbaar, kleinschalig

voertuig, gebouwd op een commercieel RC-chassis en uitgebreid met extra sensor- en rekenmogelijkheden, waaronder Lidar, een IMU, op maat gemaakte wielencoders en onboard computing. Het platform behoudt de essentiële kenmerken van voertuigdynamica op volledige schaal, Ackermann-besturing, ophanging en een elektrische aandrijving, terwijl het klein genoeg blijft om in laboratoriumomgevingen mee te werken. Een centraal onderdeel van dit hoofdstuk is een uitgebreide systeemidentificatieprocedure die betrouwbare kinematische en dynamische fietsmodellen oplevert, toegespitst op kleinschalige voertuigen, aangevuld met submodellen voor motorkracht, wrijving, stuuractuatie en bandkrachten. Deze modelleringspijplijn maakt realistische tests en nauwkeurige modelgebaseerde regeling op hardware mogelijk.

Vervolgens verschuift de focus naar coöperatieve multi-voertuigsystemen met de introductie van een gedistribueerd, aanvalbestendig platooning-raamwerk. Dit werk behandelt twee met elkaar verweven uitdagingen: het waarborgen van veiligheid en formatieintegriteit onder kwaadaardige verstoring van communicatiekanalen, en het mogelijk maken van gecoördineerde beslissingen op platoonniveau, zoals samenvoegen, opsplitsen en herschikken. Op regelniveau combineert de methode sensor-gebaseerde Adaptive Cruise Control met communicatie-gebaseerde Cooperative Adaptive Cruise Control, terwijl een veiligheidsfilter botsingsvermijding garandeert, zelfs wanneer doorgegeven acceleratiedata is gemanipuleerd. Op coördinatieniveau detecteert een gedistribueerde topologiemanagementstrategie inconsistente informatie en isoleert gecompromitteerde voertuigen door het platoon te herorganiseren. De aanpak is gevalideerd in simulatie en experimenteel met meerdere DART-voertuigen, en laat zien dat veiligheid en string stability behouden blijven ondanks communicatieaanvallen.

Met dit platform richten we ons vervolgens op het ontwerpen van een bewegingsplanningsstrategie voor stedelijk rijden. We onderzoeken model predictive control en ontwikkelen een Curvature-Aware Model Predictive Contouring Control (CA-MPCC) raamwerk. Traditionele MPCC-formuleringen veronderstellen lage kromming en maken gebruik van een lag-error term die de voortgang langs het referentietraject koppelt aan laterale tracking. Dit bemoeilijkt het afstellen en vermindert de betrouwbaarheid in scherpe bochten. De CA-MPCC-formulering lost deze beperkingen op door expliciet rekening te houden met kromming in de trajectgeometrie, de lag-error term volledig te verwijderen en zowel de kostenstructuur als het afstelproces te vereenvoudigen. De methode is gevalideerd in simulatie en op het DART-platform, en toont verbeterde robuustheid, lagere parametersensitiviteit en betrouwbare real-time prestaties, zelfs op sterk gekromde trajecten.

Daarna breidt het proefschrift de krommingsbewuste MPCC-methodologie uit naar high-performance domeinen met het rCA-MPCC-raamwerk voor autonoom racen. Racen introduceert extra uitdagingen, zoals opereren op de grenzen van het voertuiggedrag, snelle krommingsveranderingen en sterke koppeling tussen dynamische toestanden. De rCA-MPCC-formulering breidt CA-MPCC uit met een krommingsgeïnformeerde terminale kost en een compacte referentie-trajectrepresentatie die afhankelijkheid van vooraf berekende racelijninformatie vermijdt. Dit hoofdstuk breidt ook het fysieke voertuigm model uit via identificatie van actuatorsdynamica en residuele dynamische modellering met Gaussian Processes. Samen verbeteren deze stappen de voorspellingsnauwkeurigheid en maken ze robuuste regeling bij hoge snelheid mogelijk. Bovendien breiden we de autorace-formulering uit naar luchtdroneracen, waarmee een brug wordt geslagen tussen beide onderzoeksge-

meenschappen. Experimenten met kleinschalige auto's en simulaties met quadrotor-drones tonen snellere, consistenter rondetijden en robuustheid over verschillende dynamische modelkeuzes.

Ten slotte laat het proefschrift zien hoe het ontwikkelde platform en de regeltools kunnen worden toegepast op coöperatieve multi-robot taken zoals persistent monitoring en doeldetectie. Het hoofdstuk combineert DART met de CA-MPCC-regelaar om gecoördineerde trajecten op basis van Lissajous-curven te implementeren, gegenereerd door tijdgeïnverteerde Kuramoto-dynamica. Het resultaat is een gedistribueerde coördinatiestrategie die botsingsvermijding en volledige gebiedsdekking garandeert. Experimentele validatie bevestigt dat de hoog-niveau coördinatiemethode en de laag-niveau CA-MPCC-regelaar soepel integreren op echte hardware, zelfs onder verstoringen en tijdelijke uitval van agenten.

In het geheel draagt dit proefschrift bij aan een geïntegreerd raamwerk voor fysiek gefundeerde experimenten, nauwkeurige dynamische modellering en robuuste regeling en coördinatie in autonoom rijden. Door een reproduceerbaar platform te ontwikkelen, betrouwbare modellen op te bouwen en geavanceerde regelstrategieën te demonstreren, van noodmanoeuvres tot coöperatief platooning en multi-agent monitoring, verlaagt het de drempel voor validatie in de echte wereld en versnelt het de iteratiecycli voor onderzoek naar autonoom rijden. De inzichten in dit proefschrift ondersteunen toekomstig werk richting veiligere, robuustere en efficiëntere autonome transportsystemen.

1

Introduction

For more than a decade, full autonomy has been "just around the corner," yet it remains elusive. In 2012, Google co-founder Sergey Brin suggested that self-driving cars would be available to "everyone within five years" [2]; in 2016, Elon Musk said a Tesla would drive coast-to-coast without human intervention by the end of 2017 [3]; and by 2018 Waymo's CEO had tempered expectations, arguing that ubiquitous autonomy was still decades away [4]. Meanwhile, examples of deployments do exist: in the United States and parts of Asia, commercial robotaxi services operate under constrained conditions [5, 6]. These systems work by narrowing the operating design domain, effectively restricting deployments to predefined areas of the city, and by investing heavily in high-definition maps and other prior knowledge that make localization and planning tractable, an approach that is powerful but costly to build and maintain [7]. In other words, while modern perception stacks can robustly detect and track many relevant objects in structured, well-mapped areas, the broader problem of navigation, i.e., deciding how to move safely and efficiently amid uncertainty, rare events, and multi-agent interactions, still demands faster iteration on control and decision-making strategies. This thesis aims to accelerate that iteration (rapid design–test cycles) in two complementary ways: first, by introducing a low-cost, reproducible 1:20 scale open-source car-like robot to shorten the design–test loop; and second, by advancing the state of the art in control strategies for autonomous driving at both the single-vehicle level (emergency maneuvers with real-time Model Predictive Control) and the multi-vehicle level (cooperative driving and platooning) on this physical platform.

Figure 1.1: Image taken from [1]

1.1 Traffic in our cities

Picture the morning rush hour on a ring road, Figure 1.1. Sirens howl in the distance; brake lights stack into a constellation of red dots. A truck and a car have collided in the left lane. For the people in the vehicles, the consequences are immediate and personal: injury

risk, fear, and financial loss, all pivoting on the fragile luck of seconds and centimeters. For everyone else, the crash radiates outward as delay: thousands of commuters idling, missed meetings, delayed deliveries, late school drop-offs, and frayed tempers. Multiply that scene by every weekday and every city, and a simple truth emerges: our urban roads are congested, and human error behind the wheel imposes daily, society-wide costs.

The burden is not evenly distributed. As cities grow, so does the time their residents spend in traffic. Larger urban areas concentrate travel demand on finite infrastructure, making queues longer and incidents more disruptive. Demographic and economic trends point to continued urbanization and metropolitan expansion, which means the status quo of unsafe roads and mounting delays will only get worse.

Given decades of awareness and investment, why does congestion persist and crash risk remain stubbornly high? Why have we not already engineered the problem away?

Traditional remedies scale poorly. Adding lanes or interchanges is constrained by space, cost, permitting, and environmental impact, and even when built, new road capacity often induces additional demand, eroding the intended relief.

Autonomous vehicles promise a different solution: removing the human component would reduce crashes and increase effective road throughput via tighter car-following, efficient merging, and cooperative behaviors. Yet fully autonomous traffic remains out of reach. Safety concerns, the complexity of mixed human–robot environments, and the need for rigorous validation of new approaches continue to slow deployment.

1.2 Related works and research objectives

This thesis advances the state of the art towards safer and more efficient road traffic control strategies, addressing three main challenges. First we focus on the need for real-world testing at a faster pace by developing a reproducible and open-source scaled-down car-like robotic platform, providing a middle ground between pure simulation and full-scale deployment. Secondly, we focus on multi-agent cooperative driving, both on vehicle platooning, a type of formation driving where vehicles reduce the distance between them to reduce air drag and increase road throughput, as well as persistent monitoring tasks that require coordinated path tracking and speed control. Lastly, we address safety concerns, approaching the challenge at the single vehicle level, by developing control strategies for autonomous driving at the limits of handling, typical of emergency maneuvers, contributing both to the vehicle dynamics modeling and to the controllers that rely on them.

We will now provide an overview of the state of the art and highlight the gaps in the literature that we address in this thesis.

Experimental validation on small-scale platforms. Autonomous-driving research is commonly validated at two ends of a spectrum. On one end, simulation enables rapid iteration and large-scale testing, supported by increasingly capable open simulators such as CARLA and other open-source driving simulation stacks [8, 9]. However, even with these tools, simulation necessarily abstracts or idealizes parts of the physical stack; in particular, it remains difficult to faithfully capture the coupled effects of road-tire interaction, actuator and sensor dynamics, and the electronic/software sources of noise and delay that shape closed-loop behavior, especially when the goal is to validate control and multi-agent coordination.

On the other end, full-scale vehicle experiments provide realism but impose high cost, long iteration cycles, and strict safety/regulatory constraints [10], making them poorly suited for fast design-test loops when developing new planning, control, and coordination methods.

Small-scale car-like platforms are a natural compromise, providing those valuable real-world effects that are difficult to emulate in simulation, while mitigating the costs and timescales needed for full-scale autonomous driving testing.

Concerning the available small-scale platforms in the literature, we find a range of car-like robots such as AutoRally [11], MIT RACECAR [12], BARC [13], MuSHR [14], and DonkeyCar [15]. Typical prices range from $300 to about $5,000. On the high end, AutoRally targets aggressive outdoor driving but is relatively costly and involves a high number of custom parts [11]. MIT RACECAR provides a strong sensor/computation stack but relies on components that can be difficult to replicate (e.g., a self-made ESC) and has practical barriers like limited shipping availability [12]. Medium-cost platforms ($\approx$ $1,000) such as BARC and MuSHR use Jetson-class compute [13, 14] while still providing an overall versatile and reliable platform. Despite the reduced cost however, especially when performing multi-agent experiments, the price range may still be a limiting factor. On the low-price range ($300), DIY approaches like DonkeyCar are accessible but typically constrained by low-power compute and minimal sensing (e.g., single RGB camera), which reduces suitability for model-based control and multi-agent studies [15].

The key factors limiting broader adoption of existing platforms can thus be identified as the cost, the reliance on custom components, that require lab-specific technical support to deploy, and the lack of system identification procedures for small-scale platforms.

Research objective I

We aim to develop a platform that achieves comparable capability to the medium price range platforms [13, 14], while still being cheaper and easier to reproduce by maximizing the use of off-the-shelf hardware. Additionally, to support model-based control experimentation and enable reliable simulation-to-hardware transfer, we aim to provide detailed build documentation and a comprehensive system-identification workflow.

Cooperative driving. Many of the anticipated benefits of autonomous driving, e.g., higher road throughput, fuel savings, and smoother traffic flow, depend on the vehicles' ability to explicitly communicate their intentions, something that human drivers can only do implicitly and within visual range by relying on traffic rules and social norms.

Vehicle to vehicle communication (V2V) unlocks great potential to improve safety, since it can transfer information to other vehicles much faster compared to human-driven traffic. As a quantitative measure, information transfer over wireless networks is in the order of milliseconds, while the average human's reaction time is approximately 1.2s [16]. As a practical example, during an emergency situation, V2V communication can be used to quickly initiate braking in downstream vehicles.

Additionally, cooperative driving strategies such as Cooperative Adaptive Cruise Control (CACC) and vehicle platooning, rely on V2V to maintain small inter-vehicle distances at high speeds while maintaining string-stable behavior, reducing fuel consumption and increasing traffic throughput.

On the other hand however, communicating to other vehicles over wireless networks also exposes the system to cyber-physical attacks. In particular, false data injection (FDI) on

communicated acceleration/intent signals can propagate through the formation, creating oscillations, inflating gaps, or, in the worst case, resulting in a car crash [17].

A large body of related work addresses such threats via attack detection/estimation (e.g., observer-based schemes) [18], robust distributed control synthesis [19], or adaptive/learning-based mitigation [20]. Yet important limitations remain when these methods are confronted with the realities of safety-critical closed-loop operation. First, detection-based strategies inevitably introduce a *vulnerability window*, that is, they rely on the assumption that an attack will be detected quickly enough, yet safety must be guaranteed even when an attack is not detected promptly (or at all). Second, switching-based mitigation (e.g., switching from CACC to purely sensor-bases operation upon attack detection) can introduce discontinuities that degrade string stability and comfort, especially under tight spacing and heterogeneous actuation limits. Third, feasibility under input saturation remains a persistent gap: robust controllers such as [21, 22] may demand large control actions that are unachievable in practice, while MPC-based platooning such as [23] must maintain recursive feasibility and collision avoidance even when the communicated reference trajectory of the upstream vehicle is compromised.

Moreover, cooperative driving extends beyond low-level spacing control. Real deployments require *platoon-level coordination*, formation management, merge/split decisions, and rearranging, to realize system-level efficiency [24, 25]. Compared to spacing control, these coordination layers are less studied under adversarial conditions: malicious interference can target not only the control signals but also the *coordination/topology layer* (e.g., falsified topology information), and distributed treatments that explicitly address such attacks remain limited. This gap becomes even clearer when considering broader cooperative robotics tasks such as persistent monitoring, where distributed coordination is typically designed at a high level (e.g., neighbor-based synchronization in ring topologies), but lack low-level controllers that respect actuation limits and remain robust to disturbances and temporary agent failures [26].

Research objective II

With the previously mentioned limitations in mind, this thesis aims at developing cooperative driving methods that provide safety guarantees even under persistent or undetected communication attacks, avoid unsafe transients caused by hard switching, remain feasible under actuator saturation, and treat coordination/topology decisions in a distributed and attack-aware way, supported by experimental validation on real multi-vehicle platforms.

Safety focused model based predictive control. Apart from higher road throughput, fuel savings, and smoother traffic flow, the main promise of autonomous driving is to increase safety by removing one of the main sources of car crashes: the human driver. Achieving this promise ultimately requires motion-planning and control algorithms that can make safe decisions in real time while respecting vehicle and environment constraints.

Model Predictive Control (MPC) has become a widely adopted technique for autonomous driving and mobile robotics because it provides a reliable optimization-based decision-making framework: it lets the designer encode intuitive objectives (e.g., lane tracking and driving smoothness) directly in the cost function while explicitly enforcing state/input constraints, including nonlinear dynamics and nonlinear safety and feasibility constraints [27–29].

Model Predictive Contouring Control (MPCC) is a variant of MPC tailored to lane-

following applications that has seen widespread uptake in the autonomous driving community [30]. Rather than tracking a time-parameterized trajectory, as standard MPC, it tracks a geometric reference path, e.g., the lane centerline, lending itself particularly well to lane-following applications. Classical MPCC formulations introduce a progress variable s along the reference and avoid solving the closest-point projection inside the optimization by relying on simplified progress dynamics and by penalizing both contour and *lag* errors in a Frenet-like frame [31].

In autonomous driving, however, tight curvature, lane-boundary constraints, obstacles, and actuator saturation routinely lead to larger deviations from the reference path, precisely where the simpler path progress approximation becomes inaccurate. As a result, the internal progress estimate can drift, the path projection becomes unreliable, and the controller may mis-evaluate geometry-dependent costs and constraints, resulting in overly conservative speed choices, or even in violating lane boundary constraints. These issues become even more pronounced in high-performance settings that serve as proxies for emergency maneuvers, where rapid curvature changes and strong state coupling amplify modeling and timing imperfections. Many racing-oriented MPC/MPCC variants address this by exploiting offline priors such as precomputed racelines [32] or globally optimized reference solutions (often embedded through terminal constraints/terminal sets [33]), which can work well on fixed tracks but does not transfer cleanly to settings where only local geometry is available.

Research objective III

To address the limitations desribe above, in this thesis we aim to develop MPCC formulations that rely on curvature-aware path progress estimates, enabling safe deployment of MPCC-based autonomous driving even on highly curved trajectories. We furthermore aim to avoid the dependence on offline priors in autonomous racing scenarios, in order to facilitate the transfer from the racetrack to real-world emergency situations.

1.3 Contributions

Guided by the objectives outlined in the previous section, this thesis contributes to the field of autonomous driving in the following ways:

1. **The DART platform: a compact and reproducible experimental framework for autonomous driving research.**
   Chapter 2 introduces the "Delft Autonomous Robotic Testbed" (DART), a 1:20 scale car-like robotic platform designed to support rapid, low-cost, and reproducible experimentation in autonomous driving and mobile robotics. The platform strikes a balance between affordability and realism by maximizing the use of off-the-shelf components while retaining key vehicle dynamics features and difficult-to-model hardware effects such as actuation and communication delays. To favor the uptake of the platform we provide building instructions, low-level controllers for basic lane-following functionality, and a simulation environment. A systematic system identification procedure is proposed to derive accurate kinematic and dynamic bicycle models, explicitly modeling friction, motor force and steering. These models enable precise control and high-fidelity simulation, as demonstrated through use cases such

as distributed model predictive control, curvature-aware path tracking, cooperative platooning, and persistent monitoring.

2. **Distributed and attack-resilient platooning framework.**
Chapter 3 presents a novel platooning framework that embraces a robustness approach towards collision avoidance and cyber-physical security against attacks on the communication channels. Our framework considers platooning both at the control level, where the aim is to keep a fixed distance between vehicles, and at the coordination level, where the objective is to manage the topology of the whole platoon by adding, removing and rearranging vehicles in the platoon. At the control level, we combine a sensor-based Adaptive Cruise Control (ACC) policy with a communication-based Cooperative Adaptive Cruise Control (CACC) term, where a safety filter limits the authority of the communication-based control to preserve safety under corrupted data. At the coordination level, we design a distributed strategy that can reorganize the platoon formation to isolate compromised vehicles by moving them to the tail of the platoon, while also handling merging and splitting requests and remaining robust to false topology information. The effectiveness of the proposed framework is demonstrated through extensive simulations and experimental validation on a team of four scaled-down autonomous vehicles.

3. **Curvature-Aware Model Predictive Control (MPC) for urban driving.**
Chapter 4 presents a novel curvature-aware Model Predictive Contouring Control (CA-MPCC) framework for local motion planning of autonomous vehicles. The proposed method generalizes the traditional MPCC formulation, originally developed for machining applications, to handle sharp curvatures and lane boundary constraints typical of autonomous driving scenarios. Our curvature-aware reformulation accurately accounts for path curvature, removing the need for the so called "lag error" term, typical of traditional MPCC formulations. This simplifies the parameter tuning process while preserving real-time feasibility. The method is validated both in simulation and on the 1:20 scaled-down car-like robot presented in Chapter 2, demonstrating our claims of higher reliability and ease of tuning compared to the baseline MPCC formulation.

4. **Curvature-Aware MPCC for autonomous racing.**
Chapter 5 presents rCA-MPCC, a Curvature-Aware Model Predictive Contouring Control framework tailored for autonomous racing. Building upon the CA-MPCC formulation developed for urban driving in Chapter 4, the proposed approach extends its applicability to high-speed racing by introducing a curvature-informed terminal cost and a lightweight reference-path representation that do not rely on prior knowledge of the racetrack. This design alleviates the dependence on precomputed track information and enables consistent closed-loop performance with faster lap times compared to state-of-the-art racing MPCC formulations. The framework further enhances the physics-based dynamic model introduced in Chapter 2 by incorporating actuator dynamics identification, which allows for the collection of meaningful data during high-speed operation and the construction of a Gaussian Process residual dynamics model that complements the physics-based one. Extensive experimental evaluations on real robotic platforms demonstrate that rCA-MPCC achieves superior

consistency and reliability across a wide range of controller architectures and dynamic models. In addition, the framework is extended to autonomous drone racing, further showcasing its versatility across different robotic domains.

5. **Cooperative driving applications**
   Chapter 6 demonstrates the practical deployment of the DART platform presented in Chapter 2 and the Model Predictive Control framework introduced in Chapter 4 in multi-agent robotic applications. In this work, we present a distributed strategy for persistent monitoring and target detection in obstacle-free environments, where each robot follows a smooth Lissajous trajectory coordinated through time-inverted Kuramoto dynamics. While the theoretical framework establishes formal guarantees for collision avoidance and coverage, my main contribution lies in bridging this high-level coordination strategy with the real robotic implementation. Specifically, I designed and implemented the low-level Curvature-Aware Model Predictive Contouring Control (CA-MPCC) formulation tailored to this problem and deployed it on the real hardware for the experimental validation, enabling accurate path tracking under actuator limitations and dynamic constraints. The experimental results confirm the robustness and reliability of the proposed approach under disturbances and temporary agent failures, closely matching the theoretical predictions and demonstrating the feasibility of the framework on real robotic systems.

1.4 Foundations of this thesis

In this paragraph we discuss the technical principles that have guided the development of this thesis, clarifying the framework and the mindset adopted throughout the work. The philosophy behind our technical choices can be expressed with a simple principle: "align the testing conditions with the final application." For mobile robotics, this means that studying robotics requires direct interaction with a real robot, not just theoretical or simulated models, since the final objective is to deploy these systems in the real world. Simulation is a powerful tool, but it inevitably involves choices about what to model and what to omit. Anticipating which aspects will prove relevant is often difficult without validation on a real platform. For this reason, our approach emphasized experimentation with physical robots, while relying on simulation as a way to accelerate the testing in the lab. We will now discuss in more detail how this principle translated into specific technical choices in each main aspect of this thesis.

Robotic Platform Design. Our aim was to investigate evasive maneuvers and cooperative driving under realistic conditions. To maximize reproducibility, as later described in Chapter 2, we began with a commercially available robotic platform [34] and followed an iterative process: extending hardware capabilities, refining software, and returning to hardware upgrades once the software was no longer the limiting factor. Over the course of this thesis I have had the pleasure (and occasionally the headache) of building the platform from the ground up, from gluing magnets onto a speed encoder to programming the distributed logic that enables a platoon of vehicles to autonomously rearrange. For this we needed a reliable robotic testing platform. DART should capture the essential characteristics of a full-scale vehicle while remaining accessible and easy to replicate. To this end, the platform was equipped with components that mirror those of a real car: Ackermann steering,

oil-filled shock absorbers, an electric motor, a servomotor for steering, speed encoders, a Lidar, cameras, and wireless communication capabilities. While these subsystems are simpler than their full-scale counterparts they fulfill the same functionality, and, more importantly, exhibit analogous hard-to-simulate behaviors such as communication lag or actuation delays in steering and braking.

In addition, care was taken to make the platform easy to adopt by other researchers. Documentation and demonstration materials [35], were produced to lower the barrier to entry. This emphasis on reproducibility and accessibility is intended to facilitate the uptake of the platform by the broader community.

Modeling the vehicle Dynamics. A central part of this work has been the modeling of vehicle dynamics. A precise dynamic model is the backbone of any reliable control strategy, and was essential to this thesis, both in the context of model-based trajectory optimization (Chapters 4 and 5), and in cooperative scenarios, such as platooning (Chapter 3) and persistent monitoring (Chapter 6). Unlike for full-scale vehicles, good dynamic models for small-scale robotic cars are not readily available, which called for significant effort toward system identification.

Particular attention was devoted to identifying and characterizing delays in actuation and communication. Accurately quantifying and compensating for these delays not only improves runtime performance but also enables the construction of more reliable dynamic models. If unaccounted for, these delays can break the input-output correlation assumption, i.e., the assumption that the control input u is correlated with the system output x, undermining any subsequent modeling approach.

Concerning the specific choice of dynamic models, both physics-based and data-driven approaches, such as Gaussian processes, were explored. The strategy adopted in this work was to build on the extensive body of knowledge embedded in physics-based models, using them as a principled prior. On top of this structured foundation, more flexible data-driven components were introduced to capture residual dynamics that are difficult to model explicitly. This combination leverages the strengths of both approaches: the interpretability and reliability of physics-based modeling together with the adaptability of data-driven methods.

Model Predictive Control. In Chapters 4, 5 and 6, we relied on Model Predictive Control for path tracking since it has been widely adopted in the autonomous driving community due to its intuitiveness and ability to explicitly account for constraints. While alternative approaches based on machine learning, notably reinforcement learning, have shown promise in applications such as drone racing, recent comparisons suggest that MPC can still outperform, or at least match, RL-based strategies in terms of measured performance.

Indeed, as we have observed in Chapter 5, above a certain performance threshold the bottleneck shifts from the control algorithm itself to the physical limitations of the platform. In such cases, further improvements depend less on the choice of control strategy and more on the inherent capabilities of the vehicle, motivating the MPC-focused contributions detailed later in the thesis.

Robust approach to Cyber-Physical security. Concerning the platooning framework developed in Chapter 3, we start from the assumption that, given enough resources, any purely cyber-security measure may be breached. Cryptographic protocols and authen-

tication schemes provide important protection, but they cannot be regarded as absolute guarantees, particularly in the case of insider attacks or compromised vehicles with valid certificates. For this reason, we adopted a robust approach, that is, we design the control and coordination layers of the system to remain safe even when the communication channels are manipulated by a malicious entity.

In accordance to this view, we depart from many existing approaches in the literature by not assuming that cyber-attacks will be detected within a short time window. Instead, we explicitly consider the worst-case scenario in which an attack remains undetected indefinitely. This design philosophy shifts the emphasis from detection and signal reconstruction toward robustness.

1.5 Thesis outline

The remainder of this thesis is organized as illustrated in Figure 1.2. Following this introduction, Chapter 2 introduces the DART platform, a 1:20 scale car-like robot designed to enable reproducible experimentation in autonomous driving. Chapter 3 then develops a distributed and attack-resilient vehicle platooning framework that leverages this platform. Chapter 4 presents a curvature-aware Model Predictive Contouring Control (CA-MPCC) framework for motion planning in urban driving scenarios, and Chapter 5 extends it to high-speed autonomous racing. Finally, Chapter 6 demonstrates the deployment of these methods in cooperative multi-agent tasks such as persistent monitoring and target detection. The thesis concludes with Chapter 7, which summarizes the main findings and outlines possible directions for future research.

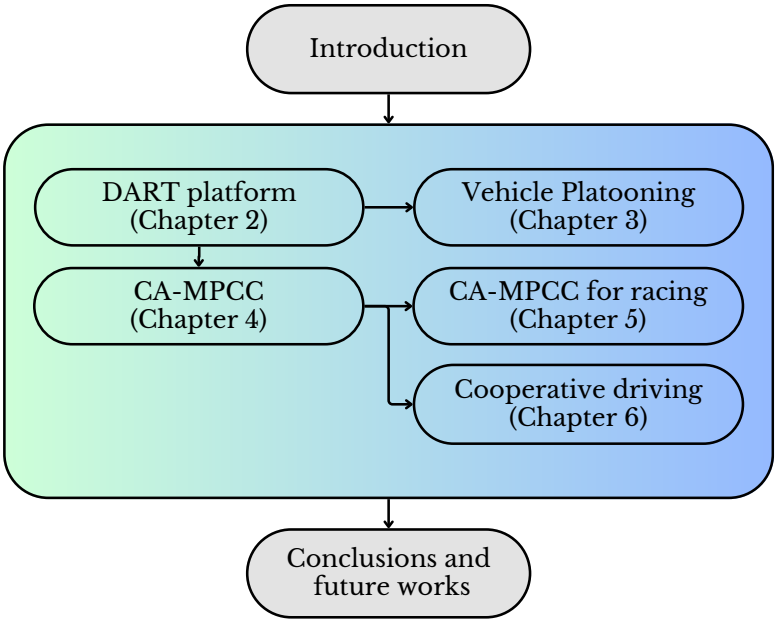


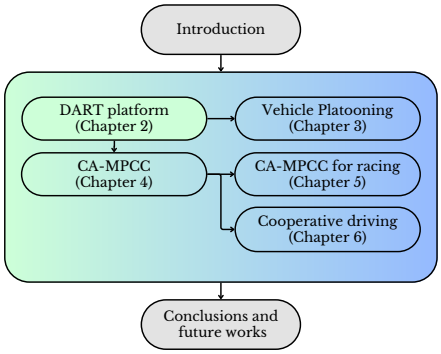
Figure 1.2: Overview of the thesis structure, illustrating the progression from foundational developments to advanced applications in cooperative and high-speed autonomous driving.

2

Small-scale autonomous vehicle platform design

Overview

This chapter presents Delft's Autonomous-driving Robotic Testbed (DART)[1] The work introduces a low-cost, small-scale, car-like robotic platform designed to bridge the gap between simulation and full-scale testing in autonomous driving research.



The chapter outlines the motivation for DART: while simulation provides flexibility, it cannot fully capture real-world sensing and actuation effects, and full-scale testing is costly and complex. Small-scale vehicles therefore provide a valuable intermediate step. However, many existing platforms suffer from high costs, limited reproducibility, or heavy reliance on custom components. DART addresses these shortcomings by building on the commercially available JetracerPro AI kit and augmenting it with a Lidar, wheel encoder, IMU, and an improved steering setup. The design emphasizes accessibility, reproducibility, and versatility, with a total cost of around €670 and minimal technical requirements to assemble.

A core contribution of the work is the detailed process for system identification of small-scale vehicles, an often overlooked but essential step for enabling model-based control and realistic simulation. The chapter presents both kinematic and dynamic bicycle models tailored to the platform, alongside identification procedures for sub-models such as motor and friction forces, steering actuation, and tire forces.

[1]This chapter is based on *Dart: A compact platform for autonomous driving research* by Lorenzo Lyons, Thijs Niesten, and Laura Ferranti [36]. L. Lyons' contribution to the original paper includes leading the hardware design, system identification, experimental validation, and writing the manuscript. T. Niesten's contribution is designing and producing the custom hardware components. L. Ferranti supervised the research.

Finally, the paper demonstrates DART's versatility through several use cases, including distributed model predictive control, persistent monitoring with multi-robot teams, vehicle platooning, and contouring MPC. Together, these examples highlight how DART can serve as a reliable and reproducible testbed for motion planning and control research, while lowering the barrier of entry for labs and educational institutions.

In addition to the results presented in the published paper, this chapter also includes further system identification work carried out after its publication. In particular, we investigate the use of machine learning techniques to model the steering actuation dynamics. By collecting an additional dataset with more detailed timing information between the control inputs and the measured states, we were able to gain a deeper understanding of the steering behaviour. This refinement allowed us to improve the accuracy of the dynamic bicycle model originally presented in the paper.

2.1 Introduction

Autonomous driving technologies have seen significant advancements in recent years leading to many companies around the world investing in, testing, and, in some cases, commercializing autonomous driving services, such as robotaxis [37] and truck platooning [38]. Despite these encouraging signals, the day when fully autonomous cars will be a large percentage of traffic participants is decades ahead of us [39], and much research is sill needed to unlock the full benefits of autonomous driving.

To deploy these technologies in our cities, extensive testing and validation of tailored navigation algorithms are necessary steps to understand potential limitations, but also gain users' trust. Assessing these methods in simulation is a cost effective solution, yet despite the increasing number of available driving simulators in recent years[8, 28], issues concerning fidelity of sensor data and vehicle dynamics still remain a significant drawback [9]. Indeed it is quite challenging, if not impossible, to accurately capture and model all the interactions among the different components of an autonomous vehicle such as road-tire interactions, actuator and sensor dynamics, electronic and software components that give rise to measurement noise and delays. On the other hand, testing autonomous driving algorithms directly on full-scale platforms entails for significant costs and long time scales, not to mention safety and regulatory restrictions. Small-scale testing platforms provide an intermediate step (or a compromise) between a purely simulated system and a full-scale autonomous car. Compared to the latter, small-scale testbeds feature simpler hardware components with simpler dynamics and low-level embedded controllers, like braking systems, powertrain and suspensions, and they also typically feature cheaper sensors such as cameras, Lidars and Radars. Yet despite these differences they still need to fulfill all the functional requirements as their full-scale counterparts, and thus offer a holistic and more realistic testing environment compared to a simulator, at a fraction of the cost of a full-scale system.

Despite the advantages of using small-scale platforms in autonomous driving research, very few models are available on the market, most of which focus on machine-learning perception algorithms for racing, making them challenging to use for control and multi-agent applications. As a result many research groups have developed their own customized platform, leading to reproducibility and accessiblity issues.

With the previously mentioned issues in mind this paper presents the Delft's Autonomous-

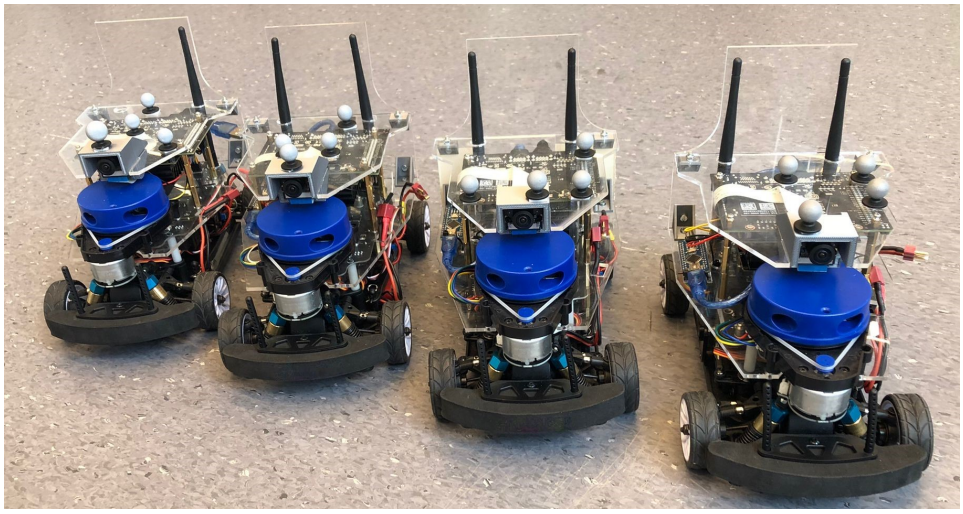


Figure 2.1: DART, a small-scale robotic platform for autonomous driving research.

driving Robotic Testbed (DART), shown in Figure 2.1. This affordable and easy to build small-scale car-like robot is designed for both single and multi-vehicle autonomous driving experiments.

2.2 RELATED WORKS

This section reviews small-scale car-like robots, roughly between 1:5 and 1:20 of a full vehicle. Full-scale research platforms and differential wheeled robots like TurtleBot [40] and Duckiebot [41] are considered out of scope for the present paper and will thus not be covered. We will describe the most well-known platforms starting from largest (1:5 scale) to smallest (1:16 scale) highlighting their main features, summarised in table 2.1 for convenience. Notice that the listed prices are indicative, as costs may vary over time and across different countries.

The AutoRally [11] is 1:5 scale vehicle-like robot platform developed for aggressive autonomous outdoor driving at the Georgia Tech Autonomous Racing Facility. The AutoRally platform is based on a 1/5 scale RC trophy truck. Due to its size, it can include a powerful mini desktop computer for state-of-the-art computationally heavy algorithms. The mini desktop can communicate through 2.4GHz/5GHz Dual-Band High-Speed WiFi and through an extra added 900 MHz XBee Pro providing a low latency, low-bandwidth wireless communication channel. Featuring high-end computation units, sensors, and actuators the total cost of this platform is around 5000$. To add these components to the main body and make them ready for outdoor conditions, several parts are customized.

The MIT RACECAR [12] (Rapid Autonomous Complex-Environment Competing Ackermann steering Robot) is a 1:10 scale vehicle-like robot platform developed by MIT. The RACECAR uses a Traxxis Rally Car platform with a powerful Nvidia Jetson TX1 computer to process all the data from the attached sensors. Communication is provided by

Platform	Cost [$]	Scale	Computing	Sensors	Drawbacks
AutoRally [11]	5.000,00	1:5	Powerful custom build computer	3DM-GX4-25 IMU Emisphere P307 GPS 2 RGB cameras Odometry sensor	High number of custom parts
MIT RACECAR [12]	3.663,00	1:10	Nvidia Jetson TX1	Hokuyo UST-10LX Stereolabs ZED Structure.io depth camera	Self-made ESC, no shipping outside USA
BARC [13]	950,00	1:10	Jetson Nano	Intel Realsense D435i RGB camera	
MuSHR [14]	1050,00	1:10	Jetson Nano	Intel Realsense D435i YDlidar X4 VEX bumper switch	
Donkey Car [15]	300,00	1:16	Raspberry Pi	RGB camera	Low power CPU, one camera

Table 2.1: Main features of the small-scale car-like platforms available in the literature.

2.4GHz/5GHz Dual-Band High-Speed WiFi supported by the Nvidia Jetson TX1. Featuring high-end computation units, sensors, and actuators the total cost of this platform, including the lidar and Jetson TX1 computer, is ≈ 3700$. The system is also equipped with a self-made Vedder ESC that can prove difficult to replicate.

The BARC [13] (Berkeley Autonomous Race Car) is a 1:10 scale car-like robotic platform developed by UC Berkeley. The latest version, V4.0, consists of a Tamiya chassis and a Nvidia Jetson Nano, that supports the same Communication features as the more powerful TX1 computer. The platform, featuring an front facing Intel Depth camera and rear facing RGB camera, costs around 950$.

The MuSHR [14] (Multi-agent System for non-Holonomic Racing) is a 1:10 scale platform developed at the University of Washington's Paul G. Allen School of Computer Science Engineering. MuSHR can be built using the Redcat Racing Blackout SC 1:10 chassis and has two additional features compared to the BARC, which are a 2D Lidar and a bumper switch for collision detections. This platform comes at a cost of around 1050$.

The Donkey Car [15] is a 3D printed platform that can easily be attached to a 1:16 scale vehicle-like robot platform, with a few mentioned in the build instructions [15] but they are no longer available. The computing module on the Donkey Car is a Raspberry Pi that uses 2.4Ghz WiFi for communication. This platform features only one RGB camera and costs around 300$.

Contributions. The main factor that hinders the widespread adoption of the small-scale platforms available in literature is the need for custom parts. Indeed many prestigious research institutions have a team of skilled technicians specifically dedicated to design and maintain the robotic platforms, yet this may not be the case for other research labs. Some DIY robots, like the Donky Car project, solve this problem by relying on easily printable 3D parts, yet they feature a limited computation power and sensor-actuator suite, diminishing their appeal as research platforms. DART aims to fill this gap in the literature by presenting itself as a low-cost yet versatile platform. The contributions of this paper are threefold:

- DART. We present the result of the design process behind the platform, as well as a list of parts and indications on the level of skill required to build it.

- System identification. A reliable dynamic model is essential to accurately control the vehicle, yet in contrast to full-scale cars where a vast amount of literature is available, much fewer works focus on model-fitting practices for small-scale cars. In this paper we present a model identification process specifically designed for 1:10 scale car-like robots.
- Hardware and software setup. To ease reproducibility and speed up the setup of the platform the building instructions and the code relative to low level control, system identification and some examples of high level controllers, as well as a simulation environment can be found in the GitHub repository [35].

2.3 DART

This section presents the main features of the proposed robotic platform. For a quick overview of the functionalities and the relative required parts see Table 2.2, while for a complete hardware and setup guide please refer to the GitHub repository [35].

DART's design adhers to the following criteria:

Accessibility
the total cost of the platform should be as low as possible, making it affordable for most research institutions, even for educational purposes.

Reproducibility
the platform should be based on a commercially available hardware and the custom parts should be as few as possible.

Versatility
the platform should lend itself to a broad variety of research fields, thus maximizing the number of researchers using the same platform, ultimately facilitating knowledge transfer.

With these criteria in mind, the robot is based on the commercially available JetracerPro AI kit [42]. This platform is originally designed for machine learning tasks and features a good computation unit (the Nvidia Jetson Nano), high reachable speeds due to the powerful brushed electric motor and 4-wheel drive.

Our goal is to use the platform to test navigation and control algorithms for autonomous and cooperative driving. Hence, to be able to use such a vehicle for these applications, we augmented the base kit as follows. We introduced a custom-made magnetic and infrared wheel encoder and an IMU necessary to produce odometry measurements, as well as an Arduino to process the raw sensor data. We also added a Lidar needed for localization and mapping (note that a camera with fish-eye lenses is already available in the base kit). Furthermore, after some testing, we upgraded the servomotor used for steering and included an external LiPo battery to power both the latter and the longitudinal driving electric motor. These two upgrades have proven to significantly improve the consistency of the measured longitudinal acceleration and steering angle, largely increasing control performances. The total cost of the platform is around €700 and requires little technical knowledge to assemble.

One challenge we faced was how to optimize the available space to accommodate the extra hardware given the size of the platform without compromising its driving performance.

Component	Functionality	Notes	Cost [€]
JetracerPro AI kit	Base robotic platform	18650 batteries are not included	420,00
YLidar X4 Lidar	Localization and mapping	–	100,00
Encoder	Wheel velocity measurement	Requires medium -low technical skills to build	10,00
IMU BNO055	Acceleration and yaw rate measurement	–	40,00
Arduino Nano	Collect data from encoder and IMU	Price includes jumper-wires and adaptors	30,00
LiPo battery	Power servo and driving motor	Price is for 2 batteries and a charger	30,00
Servomotor DS3225	Improved steering consistency	–	20,00
PVC mounting plates	Accommodate augmented sensor suite	Needs to be custom made	20,00
		Total cost	670,00

Table 2.2: Overview of the parts needed to build DART.

For this, we developed custom PVC plates to accommodate the augmented sensor suite and the encoder to measure wheel velocity. The PVC plates can be produced from the .stl files available in the GitHub repository [35] and many companies or university workshops offer online laser cutting services. The encoder requires fitting small magnets on the gear of the main shaft, yet no soldering is required and this operation requires low technical skills. Compared to platforms in the same price range, i.e. [14] and [13], DART features comparable hardware components and thus offers similar performance in terms of control accuracy and overall platform capabilities, yet relying more on off-the shelf components such as the JetracerPro AI base kit, it is much easier to replicate.

2.4 System identification

Building reliable vehicle models is not only required to run model-based controllers such as MPC, but is also necessary to develop realistic simulators to test any kind of controller before deploying it on the real hardware.

A dynamic system, such as a robot, can be represented by a system of ordinary dif-

ferential equations $\dot{\hat{\boldsymbol{x}}} = \hat{\boldsymbol{f}}(\hat{\boldsymbol{x}}, \boldsymbol{u})$, where $\hat{\boldsymbol{x}} \in \mathbb{R}^{n_x}, \boldsymbol{u} \in \mathbb{R}^{n_u}$ are the real state and control input, respectively. System identification refers to the problem of building a function $\boldsymbol{f}$, i.e., the model of the system, that is able to adequately approximate the real dynamics $\hat{\boldsymbol{f}}$ [43]. This can be achieved by recording measurements of the state and control action, collected in a matrix $\boldsymbol{X}$ of size $N \times M_x$, where N is the number of data points and M_x is the dimension of the state data and $\boldsymbol{U}$ of size $N \times M_u$, where the latter is the size of the input data. Note that the field of control the term "input" usually refers to the control action, while in system identification and machine learning the same term indicates the inputs to the model, i.e. both $\boldsymbol{U}$ and $\boldsymbol{X}$. For each row in $\boldsymbol{X}, \boldsymbol{U}$, the corresponding measured system output is collected in a matrix Y of size $N \times n_y$, where n_y is the dimension of the output of the model. In this paper we will make use of parametric models, that is, we assume that the function $\boldsymbol{f}$ has a fixed structure where a set of parameters $\boldsymbol{p} \in \mathbb{R}^{n_p}$ can be chosen in order to better approximate the real dynamics, i.e., $\boldsymbol{f} = \boldsymbol{f}(\boldsymbol{X}, \boldsymbol{U}, \boldsymbol{p})$. The optimal set of parameters $\boldsymbol{p}$ is chosen as the solution the following minimization problem:

$$\min_{\boldsymbol{p}} \sum_{k=0}^{N} \mathcal{D}\left(\boldsymbol{f}\left(\boldsymbol{X}_k, \boldsymbol{U}_k, \boldsymbol{p}\right), Y_k\right), \tag{2.1}$$

where $\boldsymbol{X}_k, \boldsymbol{U}_k$ and Y_k indicate the k-th row in the repsective matrices, and $\mathcal{D}$ is a function that measures the distance between the observed data point Y_k and the model output $\boldsymbol{f}\left(\boldsymbol{X}_k, \boldsymbol{U}_k, \boldsymbol{p}\right)$. A typical choice for $\mathcal{D}$ is the squared error, i.e. $\mathcal{D}(\boldsymbol{f}\left(\boldsymbol{X}_k, \boldsymbol{U}_k, \boldsymbol{p}\right), Y_k) = (\boldsymbol{f}\left(\boldsymbol{X}_k, \boldsymbol{U}_k, \boldsymbol{p}\right) - Y_k)^\top (\boldsymbol{f}\left(\boldsymbol{X}_k \boldsymbol{U}_k, , \boldsymbol{p}\right) - Y_k)$.

The most critical aspect of parametric model identification is the design of the function $\boldsymbol{f}(\boldsymbol{X}, \boldsymbol{U}_k, \boldsymbol{p})$ and the definition of reasonable parameter bounds, since this will heavily influence the ability of the model to correctly represent the data. While for full-scale vehicles there is a rich literature featuring various models [27], empirical formulas like the Pacejka tire model [44] for tire forces estimation, best practices for data collection and reference values for the fitting parameters [45], much less is available for small-scale cars. Furthermore, the dynamic models used for full-scale cars do not easily transfer to their small-scale counterparts, since for example the latter do not feature pressurised tires and the weight scale is around 2-3 orders of magnitude different. As a result, the range of typical values for some important parameters (e.g., the cornering stiffness) can be significantly different between full-scale and small-scale cars, making model identification particularly challenging for the latter.

In the remaining part of this section we will describe how to obtain reliable kinematic and dynamic bicycle models [46] for these small-scale vehicles. We will describe how to obtain $\{\boldsymbol{X}, \boldsymbol{U}_k, Y\}$ in problem (2.1) from the raw sensor data and how to define reasonable parameter bounds. The values of the parameters are obtained by numerically solving problem (2.1) where $\mathcal{D}$ has been chosen as the square error, using the ADAM gradient descent based algorithm implemented in PyTorch. Table 2.3 shows the obtained parameter values. Notice that all the raw data and the code for data processing and model fitting is available in the GitHub repository [35], as well as a simulation environment that uses the obtained vehicle models.

2.4.1 Kinematic bicycle model

The kinematic bicycle model consists of the following system of ordinary differential equations:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\eta} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} v\cos\eta \\ v\sin\eta \\ v\tan(\delta)/l \\ (F_m + F_f)/m \end{bmatrix} \tag{2.2}$$

Where the states x, y, η and v are the position of the rear axle, orientation, and longitudinal velocity of the vehicle, respectively. The mass and distance between the front and rear axles of the robot are indicated by m and l, respectively. F_m and F_f are the motor and friction forces, respectively. For full-scale cars, the motor characteristic curve that indicates the longitudinal force F_m transmitted to the wheels, and the steering angle δ as a function of the steering input are usually provided by the manufacturer. For small-scale cars, on the other hand, throttle τ and steering input s are provided to an ESC module and a servomotor, respectively, both of which accept normalized non-dimensional values between $[-1, 1]$. Identifying how these inputs are related to motor torque and steering angles represents an additional complication in the model fitting process.

The kinematic bicycle model (2.2) for small-scale cars requires the identification of the following sub-models: $F_f(v)$, $F_m(\tau, v)$, $\delta(s)$, as well as the actuation delays. Simply collecting driving data and fitting the model poses significant challenges since the fitting function for each sub-model needs to be user designed, and the relative parameters need to be initialized to some reasonable value in order to numerically solve Problem (2.1). For this reason our approach was to isolate each component and progressively build the full model. Attempting to identify one sub-model at a time allows us to tailor the type of test (e.g., step input or sinusoidal input test) to better highlight its contribution, while clearly visualizing the measured data aids in the design of the fitting function.

Longitudinal dynamics

We performed a series of step response tests for different throttle values on a smooth and level surface, while the steering is kept equals to zero. The collected raw data is a time series of throttle-velocity pairs. The acceleration is then obtained by numerically differentiating the velocity, while the resulting longitudinal force $F = F_m + F_f$ acting on the vehicle is estimated by multiplying the acceleration by the mass of the vehicle $m = 1.67$ Kg that can be easily measured. The training inputs $\boldsymbol{U}, \boldsymbol{X}$ are thus respectively the throttle and velocity, while the training labels $\boldsymbol{Y}$ are the resulting measured longitudinal force. Figure 2.2 shows an example of the step response data.

We start by modeling the friction force F_f. To isolate its contribution we selected the data with $\tau = 0$, that is, when the motor is switched off and vehicle is slowing down due to friction alone. By using the data showed in Figure 2.3, we designed the friction curve as:

$$F_f(v) = -(a\tanh(bv) + vc)$$

Where a, b, c are the fitting parameters. Notice that reasonable parameter initialization can be found by plotting the resulting friction curve over the data.

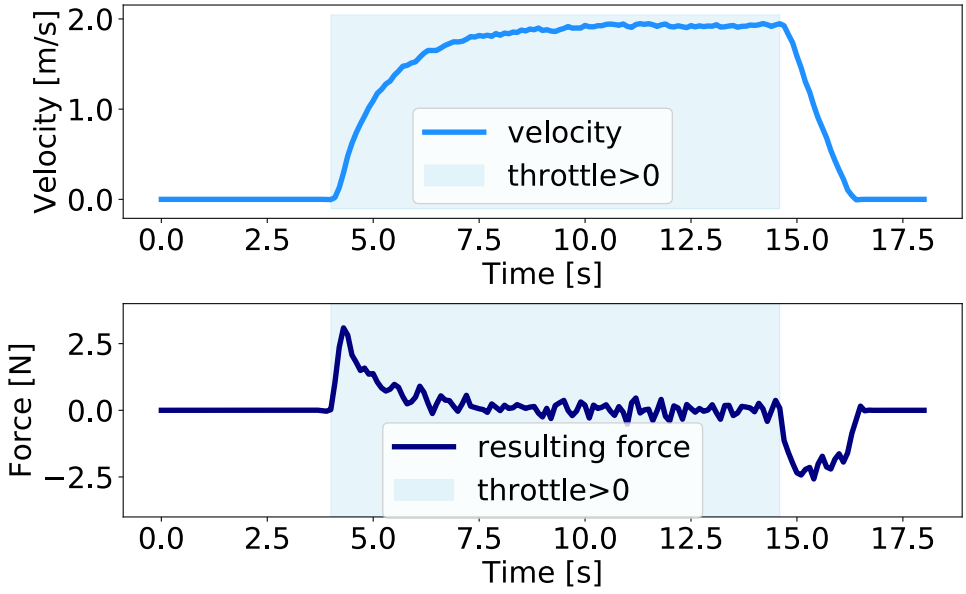


Figure 2.2: The velocity profile (top) and the estimated resulting force acting on the vehicle (bottom) measured in response to a step throttle input (shaded area).

We now procede to model the motor force F_m. Since the friction term F_f has been successfully identified, we can define new training labels $\boldsymbol{Y}$ as the estimated motor force F_m, obtained as $\boldsymbol{Y} = F - F_f(\boldsymbol{X}, \boldsymbol{U})$. We designed F_m in accordance to the characteristic curve of a brushed electric motor [47]:

$$F_m = (d - ve)\tilde{\tau}$$
$$\tilde{\tau} = (\tau + g)0.5(\tanh(100(\tau + g)) + 1)$$

Where d, e, g are the fitting parameters. Note that the term $\tilde{\tau}$ is an approximation of the function $\tilde{\tau} = \max(0, \tau + g)$ but is continuously differentiable. This is highly desirable if the model will be used for model-based control such as MPC. Reasonable first guess parameter values for d and e can be obtained by plotting the measured step response data for a certain throttle value on the $\{v,F_m\}$, plane. The initial value for g was estimated by increasing the throttle from $\tau = 0$ until the vehicle begins moving. Figure 2.4 shows the step response data for $\tau = 0.4$ and $1 \leq v \leq 3$ m/sec used to initialize d and e, and the overall characteristic motor curve, note that the latter was obtained using the full dataset featuring τ in the range $\tau \in [0.15, 0.4]$.

Steering input to steering angle mapping

The servomotor used for steering only accepts steering inputs in the range $s \in [-1, 1]$, we therefore need to identify how they relate to actual steering angles δ. To do so, we preformed a series of constant steering input tests while keeping the throttle at a constant value. The raw data consists of the resulting yaw rate $\dot{\eta}$ and the vehicle longitudinal speed

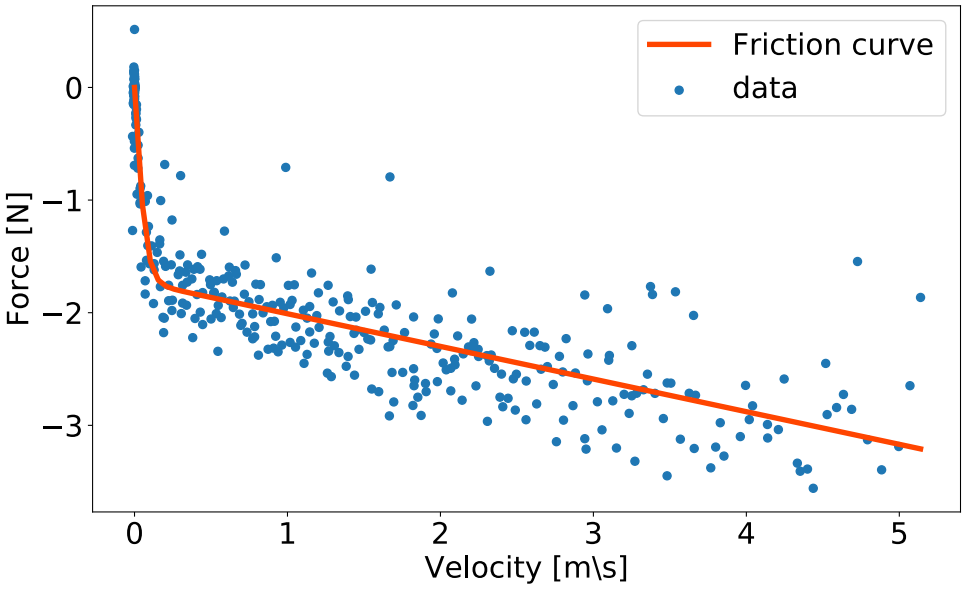


Figure 2.3: Friction curve fitting results.

v. By inverting the η dynamics (i.e., the third equation in (2.2)) we are able to estimate the steering angle as:

$$\delta = \arctan\left(\frac{l\dot{\eta}}{v}\right) \tag{2.3}$$

We can thus create a dataset using the steering input s as the training input $\boldsymbol{X}$, $\boldsymbol{U}$ and the measured steering angle δ as the training labels $\boldsymbol{Y}$. Using the data shown in Figure 2.5 we define the steering input to steering angle mapping as the weighted sum of two sigmoidal curves. This is needed to capture the asymmetry between steering to the left and steering to the right. The curve $\delta(s)$ is defined as:

$$\delta(s) = \tilde{w}\tilde{a}\tanh(\tilde{b}(s+\tilde{c})) + (1-\tilde{w})\tilde{d}\tanh(\tilde{e}(s+\tilde{c})) \tag{2.4}$$

$$\tilde{w} = 0.5(\tanh(30(s+\tilde{c})) + 1) \tag{2.5}$$

Reasonable first guess values were identified empirically from the shape of the measured data.

Actuation delays

The longitudinal actuation delay was measured by lifting the robot off the ground (so to reduce the system's inertia) and measuring the time delay between a throttle step input and a change in the wheel speed v. From these tests the longitudinal actuation delay was found to be negligible (in the order of 0.01 sec). The steering delay was instead obtained by providing the vehicle with a low frequency sinusoidal steering input. This allowed us to

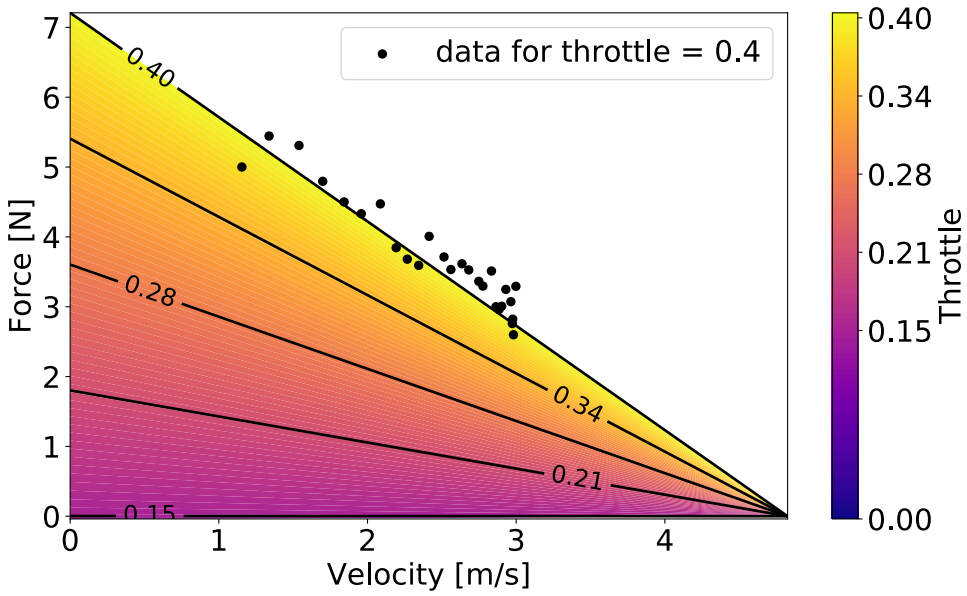


Figure 2.4: Motor curve fitting results.

capture the reaction time of the steering, that is, the time interval before seeing a reaction after sending a steering command. The delay was then estimated by performing the cross correlation [48] between the steering angle input $\delta(s)$ and the measured steering angle obtained from equation (2.3). The steering delay was found to be around 0.15 sec and is thus not negligible.

2.4.2 Dynamic bicycle model

DART allows one to perform highly dynamic maneuvers (e.g., racing). In such a context, a kinematic model is no longer a good representation of the vehicle dynamics, since it neglects the lateral dynamics of the vehicle. Indeed it relies on the assumption that no lateral slipping occurs, that is, there is no lateral motion in the vehicle body frame. This assumption holds until the tires are able to provide enough lateral friction force to counter the centrifugal force during curves. The centrifugal force can be evaluated as $F_y = m\,\dot{\eta}\,v$. For high speed and high yaw rate, the assumption that lateral slip is negligible is no longer valid and the kinematic bicycle model fails. The dynamic bicycle model, on the other hand, does account for the lateral dynamics and relies on a tire model to predict tire force saturation, providing better model accuracy at high lateral accelerations.

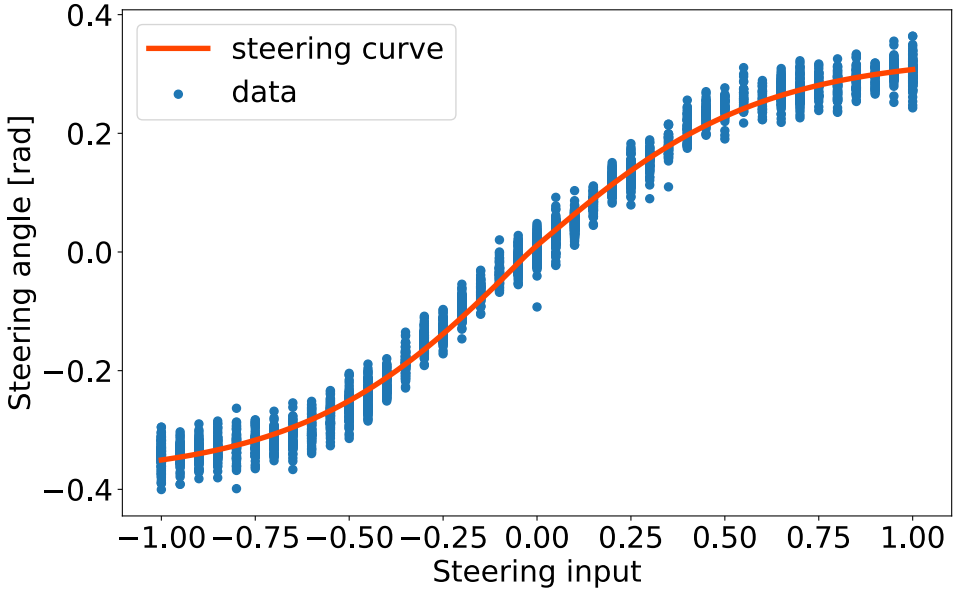


Figure 2.5: Steering input to steering angle static mapping.

The dynamic bicycle model is defined as:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\eta} \\ \dot{v}_x \\ \dot{v}_y \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} v_x \cos\eta - v_y \sin\eta \\ v_x \sin\eta + v_y \cos\eta \\ \omega \\ (\hat{F}_{x,r} + \hat{F}_{x,f} \cos\delta - \hat{F}_{y,f} \sin\delta)/m + \omega v_y \\ (\hat{F}_{y,r} + \hat{F}_{y,f} \cos\delta + \hat{F}_{x,f} \sin\delta)/m - \omega vx \\ (l_f(\hat{F}_{y,f} \cos\delta + \hat{F}_{x,f} \sin\delta) - l_r \hat{F}_{y,r})/I_z \end{bmatrix}, \tag{2.6}$$

where v_x and v_y are the velocity components of the centre of mass measured in the vehicle body frame. l_f and l_r are the distances between the centre of mass and the front and rear axle, respectively. ω is the yaw rate and I_z is the moment of inertia around the vertical axes. The front and rear tire forces components $\hat{F}_{f,x}$, $\hat{F}_{f,y}$, $\hat{F}_{r,x}$ and $\hat{F}_{r,y}$ are measured in the respective tire's body frame. The evaluation of the tire forces is the most critical component of the dynamic bicycle model and a vast amount of literature is available for full-scale cars. One of the most famous tire models is the Pacejka magic formula [44], defined as:

$$\hat{F}_y = D \sin(C \arctan(B\alpha - E(B\alpha - \arctan(B\alpha)))) \tag{2.7}$$

where α is the slip angle, defined as the angle from the direction of motion of the tire and the tire axis. Front and rear slip angles are defined as:

$$\alpha_f = \delta - \arctan\left(\frac{v_y + l_f\omega}{v_x}\right) \tag{2.8}$$

$$\alpha_r = -\arctan\left(\frac{v_y - l_r\omega}{v_x}\right) \tag{2.9}$$

To identify the parameters D, C, B, E we collected driving data keeping a constant steering angle and gradually increasing the longitudinal velocity. To ensure numerical stability at very low velocities in Eq. (2.8) and (2.9), we correct the value of v_x to a small positive number. Since we need to measure both longitudinal and lateral velocities in the vehicle body frame we used an external motion capture system, as the on-board sensors are not able to measure such quantities. The raw data thus consists of the vehicle's centre of mass position and orientation in the global reference frame. We then computed the time derivatives to measure the absolute velocity $\tilde{v}_x$, $\tilde{v}_y$ and acceleration $\dot{\tilde{v}}_x$, $\dot{\tilde{v}}_y$, as well as the yaw rate ω and it's time derivative $\dot{\omega}$. The training inputs $\boldsymbol{X}$ are thus the front and rear slip angles evaluated as in equations (2.8) and (2.9), where the velocities in the vehicle body frame are evaluated as:

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} \cos(-\eta) & -\sin(-\eta) \\ \sin(-\eta) & \cos(-\eta) \end{bmatrix} \begin{bmatrix} \tilde{v}_x \\ \tilde{v}_y \end{bmatrix} \tag{2.10}$$

The training labels $\boldsymbol{Y}$ are the lateral forces in the tire body frame and have been evaluated by solving the equations of motion of the body in the absolute frame of reference:

$$\begin{bmatrix} F_x \\ F_{y,f} \\ F_{y,r} \end{bmatrix} = \begin{bmatrix} \cos\eta & -\sin\eta & -\sin\eta \\ \sin\eta & \cos\eta & \cos\eta \\ 0 & l_f & -l_r \end{bmatrix}^{-1} \begin{bmatrix} \dot{\tilde{v}}_x/m \\ \dot{\tilde{v}}_y/m \\ \dot{\omega}/I_z \end{bmatrix} \tag{2.11}$$

Where F_x is the total longitudinal force in the vehicle frame, $F_{y,f}$ and $F_{y,r}$ are the front and lateral forces in vehicle frame, and $I_z \approx 0.006513\text{Kgm}^2$ was estimated considering the robot as a rectangle with uniformly distributed mass of size $l \times w$, where $w = 0.1\text{m}$ is the width of the vehicle. The lateral forces in the tire frame of reference are finally evaluated as:

$$\hat{F}_{y,f} = \sin(-\delta)F_{x,f} + \cos(-\delta)F_{y,f} \tag{2.12}$$

$$\hat{F}_{y,r} = F_{y,r} \tag{2.13}$$

Where $F_{x,f} = \frac{1}{2}F_x$ since the vehicle features a 4 wheel drive and we assume that the motor force is equally shared by front and rear axle. Figure 2.6 shows the obtained data and fitting results. Notice that since the rear tire does not reach saturation we chose to model it with a simple linear relation instead of equation (2.7), i.e., as:

$$F_{y,r} = C_r\alpha_r. \tag{2.14}$$

Additionally, we notice in table 2.3 that the value for C is less than 1, while typical ranges for full scale vehicles are between 1 and 2. In our case $C < 1$ indicates that the curve does not have a peak value, but rather asymptotically reaches a maximum value. This may be due to the fact that the tyres of our small scale vehicle are not pressurized, showing a simpler friction model.

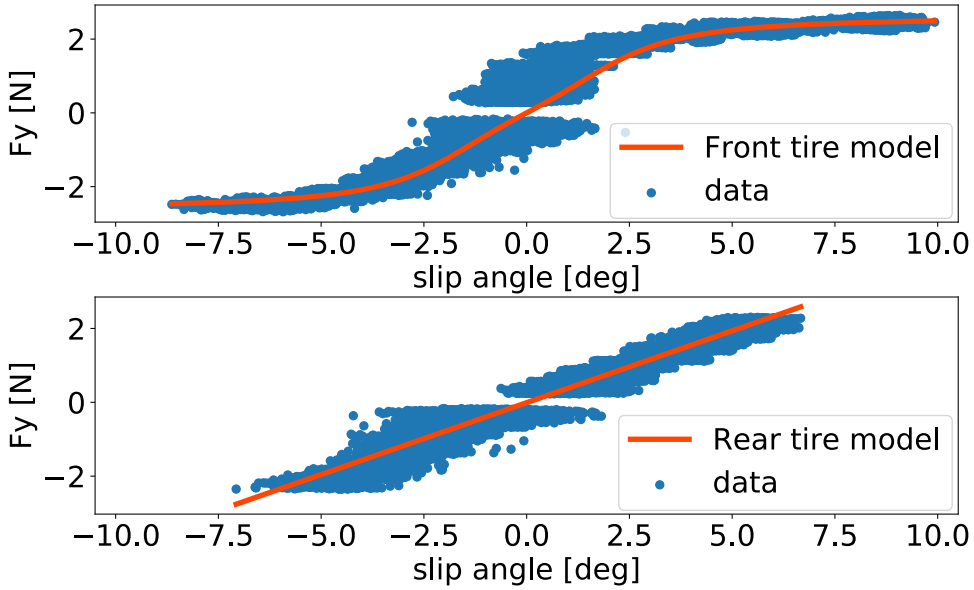


Figure 2.6: Lateral tire force model for the front tire (top) and rear tire (bottom).

a	1.72	d	28.88	$\tilde{a}$	1.64	$\tilde{d}$	1.66
b	13.32	e	5.99	$\tilde{b}$	0.33	$\tilde{e}$	0.38
c	0.29	g	−0.15	$\tilde{c}$	0.02	C_r	0.39
D	2.98	C	0.69	B	0.29	E	−3.07

Table 2.3: Identified parameter values.

2.5 Use cases

This section presents an overview of work that featured DART as a test bed[2]. From a functional point of view all these applications can be seen as a tracking problem, where a vehicle needs to follow a path at a certain reference speed. This reference speed may be fixed, or may be evaluated at runtime based on the robot's global position or on the position and/or velocity of other robots. To perform this planning task the robot needs a good dynamic model in order to track the reference velocity, access to its own pose (position and orientation) in the global reference frame, and access to the other robots' poses if needed. This can be achieved by means of an external motion capture system if it's available, or by means of on-board localization. The latter is however a viable option only for low to medium speeds, since the two key components are a sensor to perceive the environment, such as a camera or a lidar, and odometry data provided by the kinematic bicycle model (see section 2.4). In the remainder of the section we will describe the main

[2]The aim of this section is to showcase the platform's capabilities thus we will not provide an in-depth discussion on the scientific merits of the described experiments.

features of each kind of experiment, focusing on the functional requirements the platform needed to meet.

Distributed MPCC [29]
We presented a distributed Model Predictive Contouring Control algorithm (D-MPCC) for a team of robots. Each robot aims at following a certain path at a given reference speed, while avoiding collisions with other agents. This is achieved through a distributed computation scheme that also accounts for possible packet loss over the communication network. The algorithm was tested in an intersection crossing, shown in Figure 2.7, and a lane merging scenario. In both cases the robots need to be aware of each other's position and intended future trajectory in order to avoid collisions. To achieve this a good vehicle model is required, to ensure a consistent behaviour.

Persistent monitoring [49]
A team of robots is tasked with monitoring a certain area. Each robot is equipped with omnidirectional sensors that are able to detect a target within a certain range. By relying on Lissajous curves and time-inverted Kuramoto dynamics, all vehicles follow the same smooth path within the designated area and adjust their speed based on the current position of the preceding and following vehicles, as shown in Figure 2.8. The emerging behaviour of the mobile sensor network is guaranteed to detect a moving target within bounded time and avoid collisions among agents. To successfully carry out the experiments the vehicles need to follow a highly curved path while accurately control their speed, since the latter needs to be adjusted according to the local path curvature and to the position of the other robots.

Vehicle platooning
Platooning refers to the problem of vehicles driving along a relatively straight path while maintaining a certain distance among each other. The typical application is heavy duty trucks driving on a highway, where for small inter-vehicle distances significant fuel efficiency can be gained due to air drag reduction. To achieve this behaviour the vehicles need to share information on their current speed and position. From a practical point of view, the main challenge is that experiments require a long straight path, thus an external motion capture system will typically not be large enough, requiring the robots to rely on on board sensors for localization. Thanks to the lidar and on board odometry data this can be achieved using standard ROS libraries for Simultaneous Localization And Mapping (SLAM) and Adaptive Monte Carlo Localization (AMCL). Another significant challenge is to carefully adjust the steering in order to limit lateral deviations from the path. This is why we upgraded the servomotor used to steer the robot. Figure 2.9 shows the robot using on-board localization and steering and velocity controllers to follow a straight path.

Contouring MPC [50]
We present a Model Predictive Contouring Control algorithm (MPCC) that also includes the information on the local path curvature, called Curvature-Aware MPCC. The new formulation features an improved estimation of the progress along the path and consequently more reliable lane boundary constraint satisfaction. Furthermore it features less cost function terms and is thus easier to tune. As far as experiments are concerned the

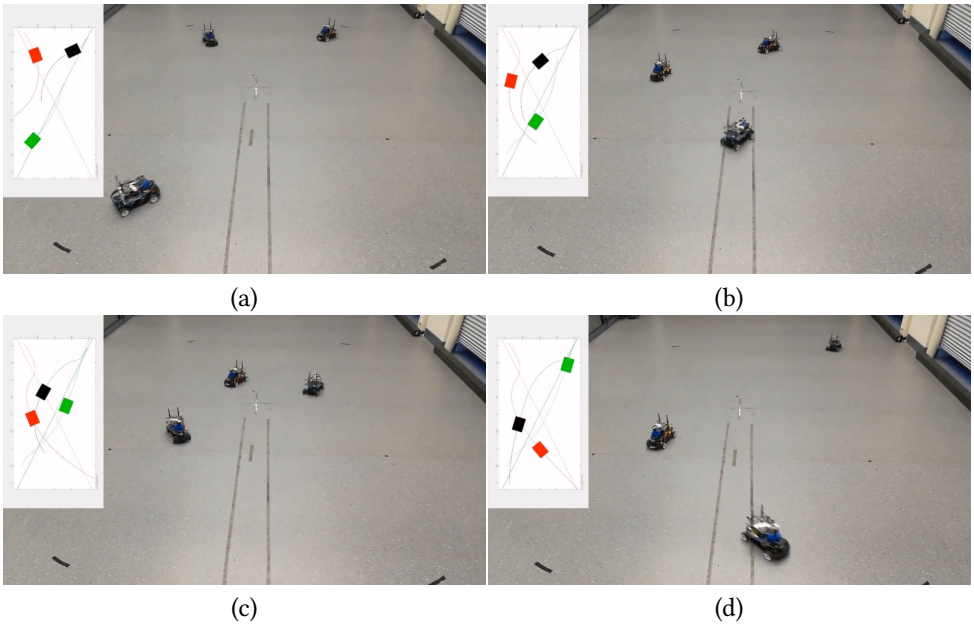


Figure 2.7: A team of robots navigate through an unsupervised intersection crossing using a distributed MPC scheme. This figure has been taken from [29].

main requirement for the platform is to exhibit consistent behaviour in order to highlight the differences due to the specific algorithm's formulation. This requires a good vehicle model. Figure 2.10 shows the robot following a highly curved path while avoiding collision with a virtual dynamic obstacle.

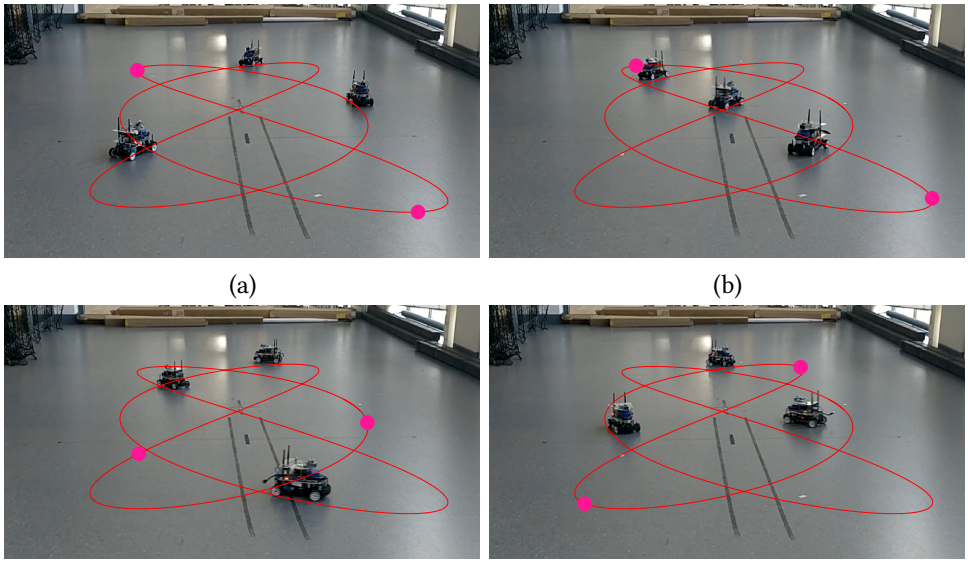


Figure 2.8: A team of robots following a Lissajous curve under a time-inverted Kuramoto dynamics feedback controller. This figure has been taken from [49] .

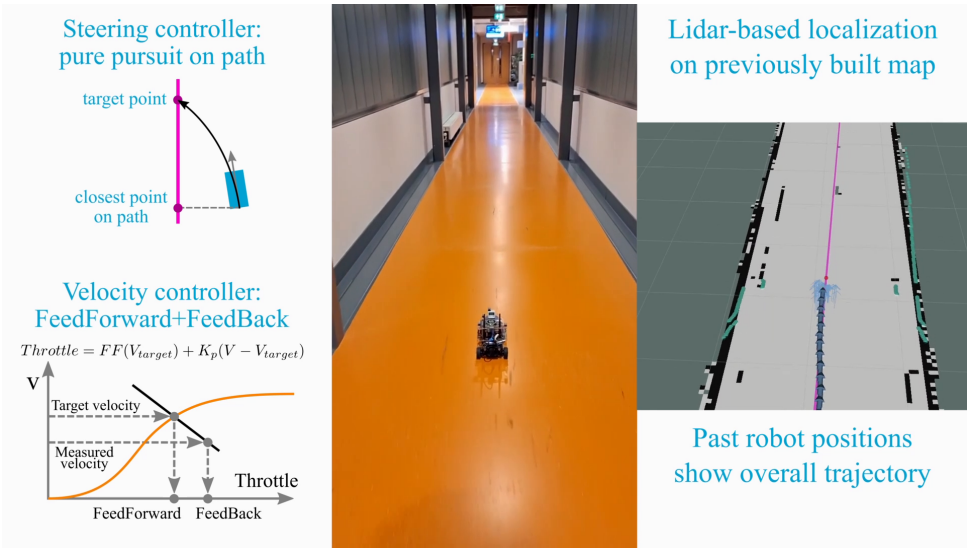
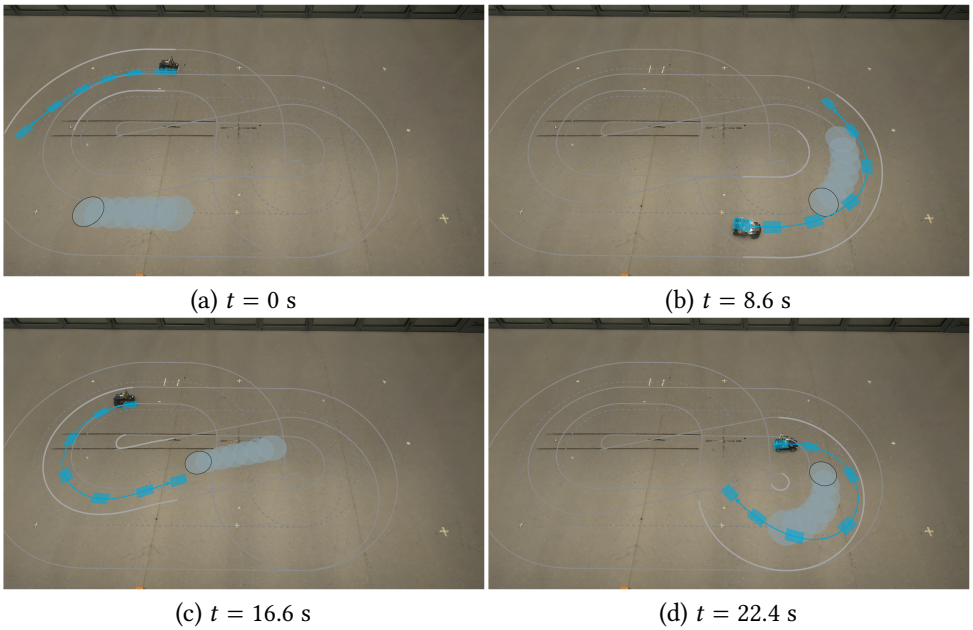


Figure 2.9: A robot following a straight line using on-board sensors for both localization and state feedback controllers. This image was taken from [51].



(a) $t = 0$ s (b) $t = 8.6$ s

(c) $t = 16.6$ s (d) $t = 22.4$ s

Figure 2.10: A robot following a highly curved path with the Curvature-Aware MPCC controller. This figure has been taken from [50].

2.6 Conclusions

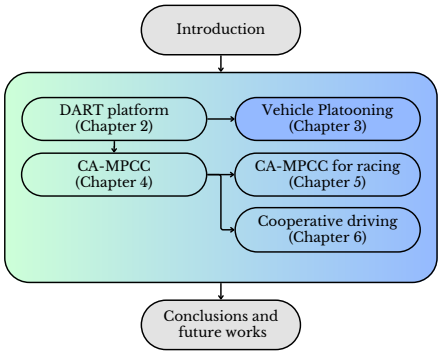
This paper introduced the Delft's Autonomous-driving Robotic Test bed (DART), a small-scale car-like robot suitable for (multi-robot) motion planning and control. Compared to other available small-scale platforms DART maximizes the use of available off-the-shelf hardware and features a lower number of custom parts, making it cost-effective and easier to reproduce. We provide a system identification procedure to obtain kinematic and dynamic bicycle models that allow the platform to be used for a wide range of applications, as demonstrated by the number of published works that featured DART as a test bed. Finally, we provide a GitHub repository containing building instructions, the data and code used for the identification, as well as a simulation environment and some readily-available low-level controllers.

3

Attack resilient vehicle platooning

Overview

This chapter work addresses the problem of securing cooperative vehicle platooning against cyber-physical attacks on the communication channels, with a particular focus on false data injection (FDI)[1]. While platooning has the potential to significantly improve traffic flow, fuel efficiency, and road safety, its reliance on vehicle-to-vehicle (V2V) communication exposes it to malicious interference that may jeopardize both safety and stability of the formation.



The paper introduces a novel distributed platooning strategy that guarantees safety even in the presence of undetected attacks. The method combines two layers: a low-level platoon controller and a distributed high-level coordinator. The low-level controller fuses a sensor-based Adaptive Cruise Control (ACC) policy, which ensures safety and robustness, explicitly accounting for actuation limits, with a Cooperative-ACC (CACC) policy that leverages communication for improved performance. To prevent unsafe behavior under compromised data, the communication term is subject to a safety filter that limits its control authority, thus guaranteeing collision avoidance even if the attack is remains undetected indefinitely. A tailored gain-tuning procedure allows to select inter-vehicle distances as small as 5-10

[1]This chapter is based on the paper *Distributed Attack-Resilient Platooning Against False Data Injection* by Lyons, Boldrer, and Ferranti. L. Lyons' contribution consists in developing the methodology, except the platoon coordinator (Section 3.3.4), performing the method validation in simulation, designing the platoon rearranging logic, building the robot localization and object detection pipelines, performing the experiments and writing the manuscript. M. Boldrer developed the platoon coordinator and wrote part of the introduction. L. Ferranti supervised the research.

meters, which are essential for maximizing aerodynamic and fuel-saving benefits, while preserving string stability and collision avoidance.

Although safety is formally guaranteed by design, an adversary can still impact the platoon's performance. In particular, malicious data injection at the communication level, may increase the inter-vehicle spacing or introduce oscillations in the formation. To mitigate performance degradation under attack, the framework incorporates a Kalman filter-based attack detection algorithm that monitors the consistency between on-board sensing and received data. Upon detection, the system switches from CACC to ACC mode to preserve safety.

As an additional attack-mitigation strategy, the high-level platoon coordinator can reorganize the formation to isolate a compromised vehicle. The coordinator also manages merging and splitting requests, making the framework suitable for real-world deployment.

The proposed method is validated extensively in simulation against state-of-the-art Model Predictive Control (MPC) and robust control approaches, showing superior safety guarantees under a wide range of attack scenarios, including undetected persistent attacks and emergency braking conditions. Furthermore, the algorithm is implemented on a fleet of four scale-model autonomous vehicles, where real experiments demonstrate its ability to maintain string stability, prevent collisions under FDI attacks, and successfully detect and isolate compromised vehicles through topology reconfiguration. To support reproducibility and further research, the authors provide open-source code and experimental data.

3.1 Introduction

Autonomous vehicles are becoming a reality and promise to radically change the world of transportation. In the last decade, many studies have been focused on platooning of autonomous vehicles. Platooning driving involves a group of vehicles traveling in a formation, with each vehicle maintaining a constant distance behind the preceding vehicle. Platoon-based driving can decrease traffic congestion, reduce emissions, improve road safety and driving comfort [24], as well as alleviate the need for human drivers, that is becoming an even more pressing issue [52].

To enhance coordination among vehicles in a platoon, it is essential to combine the use of on-board sensors and a communication network. Relying solely on on-board sensors can limit the vehicles' understanding of their neighbors' intended motion, affecting the overall platoon performance (e.g., more conservative inter-vehicle margins). While a communication network allows vehicles to exchange information about their intentions and improve platoon performance, communication among the vehicles can also introduce vulnerabilities into the system. In fact, the information transmitted through a communication network is more susceptible to malicious attacks, such as false data injection (FDI) or denial of service (DoS) [53].

This paper focuses on the design of a distributed algorithm to achieve attack-resilient longitudinal platooning. The algorithm combines on-board sensing and vehicle-to-vehicle (V2V) communication. It provides collision avoidance guarantees and is also able to detect and isolate malicious attacks to continue the operation of the platoon by conveniently reconfiguring the platoon formation. It can furthermore manage merging and splitting requests. This work considers FDI attacks on the acceleration signal that is sent from the predecessor to follower vehicle, (see Figure 3.1). We consider the case where the attacker may have full

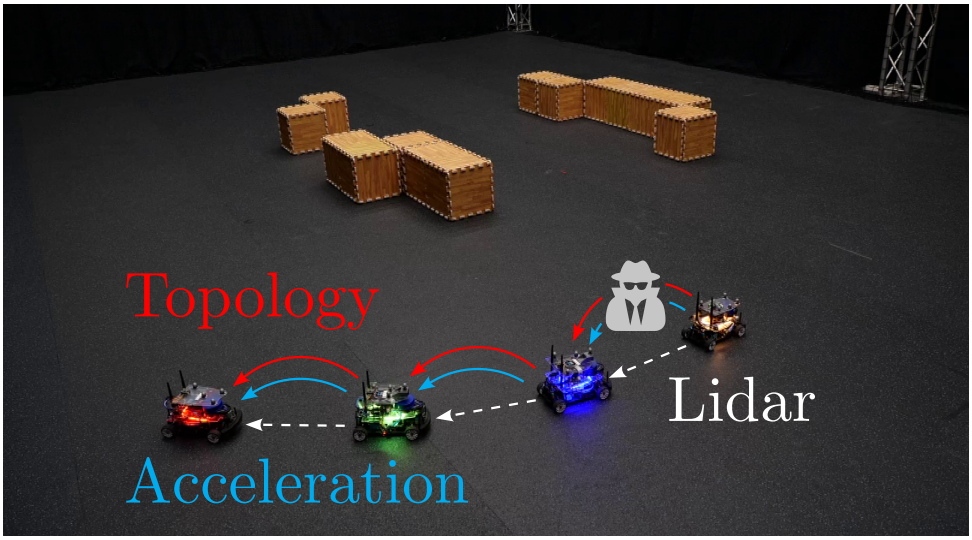


Figure 3.1: Experimental platform consisting of 4 scaled-down car-like robots [54]. Each robot measures its state relative to its predecessor using a Lidar and receives acceleration and platoon topology data through the communication network. Data flowing through the network is however susceptible to man-in-the-middle attacks (e.g. between the leading vehicle and its follower as shown in the figure) that we mitigate with our strategy.

knowledge of the system and acts in a strategic manner to cause a crash in the platoon.

3.1.1 Related works

Vehicle platooning is largely addressed in the literature. As [55] describes, platooning can be addressed on two different levels, *Platooning control*, which relates to the design of the low-level controller that allows to maintain the desired distance among the vehicles and is responsible for the safety and stability of the formation, and *Platooning coordination*, which takes care of the platoon management, that is, the organization and composition of the platoon formation itself.

Platooning control

There is a plethora of methods in the literature. Traditional approaches rely on linear feedback controllers such as [56], while in recent years model predictive controller (MPC) based solutions, such as [57–60], have become increasingly popular. Strategies based on techniques other than feedback controllers or MPC include artificial potential fields [61], consensus-based platooning [62], and Deep Reinforcement Learning [63].

Irrespectively of the specific platooning strategy, a necessary requirement for real-world applicability is providing safety by avoiding rear-end collisions. In the literature, this requirement has almost always been translated mathematically into the problem of proving string stability. Although definitions may vary slightly, the core idea is that disturbances in acceleration, velocity, or position introduced at the head of the platoon should not be amplified as they propagate downstream. In this work, we formalize string stability using a frequency-domain transfer function approach (see Section 3.3.2), where we require that

the magnitude of the transfer function from one vehicle to the next remains below one across all frequencies. This ensures that oscillations decay along the platoon. However, as we show, string stability alone is not sufficient to guarantee collision avoidance in the presence of actuator saturation, which we address through additional safety constraints.

In Adaptive Cruise Control (ACC) based on traditional linear feedback controllers, the typical way to prove string stability is to rely on frequency domain analysis and show that the magnitude of input and state signals is decreasing along the platoon [56]. However, since these methods can not include actuator limitations, collisions in the head of the platoon may still occur during emergency braking scenarios as shown in [56], where the issue of safety is addressed by performing an *a posteriori* analysis that identifies the set of safe controller parameters, yet no *a priori* safety guarantees or design criteria are provided.

When a platooning strategy uses communication among vehicles it is referred to as Cooperative-ACC (CACC). This brings significant advantages, since vehicles can now act in a predictive way by knowing their predecessor's intentions. The most popular class of CACC strategies is MPC since it also allows to consider actuation limits. The connection between actuator limits and safety concerns is further detailed in [21], where collision avoidance is provided in a bidirectional cooperative setting by ensuring that saturation will not be reached in nominal platooning conditions, while in [23] the authors rely on an a worst-case preceding vehicle prediction to evaluate the risk of collision.

However, the benefits of communication-based platooning strategies can be completely compromised in the presence of malicious attacks. As argued in [53], the risk of an adversarial attack on wireless communication is far greater than an attack on the on-board sensors, which makes cyber-physical security a major concern for CACC. The consequences of a malicious attack can be catastrophic, since, once injected, the false information may flow in the network, affecting the whole system and threatening efficiency, stability, and safety of the formation. The authors of [17] reviewed the impact of several attacks on existing platoon controllers. In the literature, there are different approaches to deal with malicious attacks, as is discussed in [22], but all of them suffer from specific issues.

Observer-based control strategies (e.g., [18]) may experience latency problems due to the detection and mitigation mechanism. Robust control strategies (e.g., [19]) can generate high control inputs, which can be unfeasible in a real scenario due to input saturation. Adaptive control strategies (e.g., [20]) can experience unwanted transient response issues. MPC strategies do not easily provide guarantees on the existence of feasible control solutions.

Another specific issue with MPC that often eludes the literature's attention is that after a vehicle detects a compromised communication channel it can not simply shut it down, since the MPC requires the preceding vehicle's predicted trajectory as a reference. To circumvent the issue attack-resilient strategies either rely on identifying the attack vector and reconstructing the true signal [64], or generating a worst case reference trajectory for the preceding vehicle [65]. These solutions however either suffer from latency issues because they rely on a bank of past data to build a prediction of the preceding vehicle's behavior or may result in overly conservative inter-vehicle margins.

A promising approach is presented in [66] and [67], where the authors leverage the assumption that on-board sensors are more reliable than the communication channels and define an MPC-based CACC policy that will switch to a sensor-based ACC strategy in the presence of either significant communication time delays as in [66], or when a malicious

attack is detected as in [67].

However we identify 3 issues with these approaches. Firstly, although these strategies can disable a compromised communication link and revert to an ACC policy, the downstream vehicles are now left without an open-loop reference for the MPC. Secondly, in the case of a malicious attack, [67] relies on a quick attack detection module to avoid safety issues during the time when a malicious attack is present but not yet detected. Lastly, when switching from CACC to ACC mode there is a discontinuity in the control strategy and guaranteeing safety after the ACC takes over is not trivial in the presence of actuator saturation.

Platooning coordination

Platooning coordination is less addressed in the literature [55]. A fuel-optimal centralized solution is proposed in [25]. In [68] the authors propose a non-cooperative game that models the multi-fleet platoon matching problem. In [69] a decentralized bio-inspired strategy for platoon management is proposed. A taxonomy on the objectives that may be considered for platooning is provided in [70]. In [71] the authors present infrastructure-aided platoon management strategies for highways and urban areas, while in [72] the authors propose both a centralized and decentralized high level platoon coordinator, which updates the parameters of the low-level platoon controller based on traffic conditions. While most platoon coordination methods focus on traffic flow management and managing platoon merging and splitting requests, our proposed coordinator is specifically designed to mitigate the effects of a malicious attack on the communication channels. In particular, leveraging the predecessor-follower topology to isolate a compromised vehicle. Notice that in practice, an attacker may be a vehicle affected by malicious software or by a man-in-the-middle attack at the communication network level [53]. We furthermore observed a lack of literature related to malicious attacks at the coordination level, this is mainly due to the fact that the proposed strategies are mostly centralized or decentralized. In contrast, we propose a distributed solution that can also deal with the presence of an attacker in the network. We differentiate between attacks at the platoon coordinator level from attacks on the platoon controller according to the type of data being compromised. Attacks at the coordinator level concern topology information, i.e. in what sequence the vehicles should be arranged. Attacks on the platoon controller concern data about the vehicle's state such as acceleration and braking. Typically the latter is more safety critical since it's the controller's function to maintain the specified distance between vehicles, while the former deals with higher level decision making about the platoon formation.

3.1.2 Our contribution and paper organization

This paper presents a novel distributed and attack-resilient algorithm for platooning. It features three main contributions. First, we provide a low-level platoon controller that combines a linear sensor-based ACC controller, which is unaffected by network corruption, with a communication-based CACC controller, which improves performance but may be vulnerable to FDI in the communication channel. The proposed strategy falls in the category of ACC-CACC switching controllers. Differently from [67], we do not rely on MPC, since when a vehicle switches to ACC mode, it will not have an open loop prediction of its intended motion to pass to its following vehicle. The proposed method addresses the

limitations we identified in the state of the art:

- In [67] the authors rely on the implicit assumption that a malicious attack will be detected quickly enough to prevent any safety-related concerns, while our method is able to guarantee safety even if a severe attack remains undetected indefinitely. To do so, similarly to [23], we define a set of constraints that limit the authority of the communication-based controller, i.e. a *safety filter*, and perform a worst-case scenario reachability analysis. In contrast to [23], we additionally consider communication and design the safety filter to keep the vehicle state within a safe set even in case of a malicious attack.

- Concerning the fallback ACC algorithm, we directly address the issue of guaranteeing safety when switching to ACC mode of operation. We design a gain tuning procedure that provides both string stability and collision avoidance at the head of the platoon by considering actuation limits. Furthermore, the gain tuning procedure allows one to select an arbitrarily small inter-vehicle distance. This is of particular relevance since the highest gain in terms of aerodynamic drag and fuel efficiency are achieved for very small inter-vehicle distances of around $5 - 10$m [73], and typical constant time-headway policies (such as [56]) do not allow to select a fixed spacing among vehicles, or lack formal *a priori* safety guarantees.

Second, we propose a novel distributed platoon coordination strategy that allows to reorganize the platoon in order to isolate compromised vehicles, further limiting the effect of a malicious attack. The coordinator module, that runs on each vehicle, is also able to handle platoon merging and splitting requests. In the literature, only centralized or decentralized strategies are proposed for this purpose (not distributed). Moreover, our approach directly considers the reliability of a communication channel as a parameter to reorganize the platoon. Furthermore, our platoon coordinator can detect and isolate a malicious vehicle that communicates false data at the coordinator level.

Third, we implemented the proposed platooning algorithm on real robotic platforms using a team of 4 scaled-down car-like robots [54], see Figure 3.1, and provide a working code base to deploy our method in practice [74]. The videos of the experiments can be found in [75].

To complete our framework we also present a tailored Kalman filter-based attack-detection policy specifically designed for platooning applications, that signals both to the low-level platoon controller to switch from CACC to ACC mode, and to the platoon coordinator to trigger a platoon reconfiguration maneuver.

The paper is organized as follows. Sec. II introduces the problem. Sec. III provides the proposed algorithm for resilient platooning. Sec. IV shows the simulation comparison between our low-level platoon controller, the MPC strategy [57] and the robust control approach [76]. Sec. V shows the experimental results. Sec. VI concludes the paper.

3.2 Problem description

This paper proposes a distributed solution for attack-resilient platooning. This problem is formulated as follows:

Problem 1. Given N vehicles, we want to achieve platoon-based driving, where each vehicle maintains a constant distance from the preceding vehicle. The platoon-driving condition has to:

1. ensure *safety*, that is, avoid inter-vehicle collisions;
2. ensure (predecessor-follower) *longitudinal string stability*. A system is *string stable* if any error in position, velocity or acceleration does not amplify along the platoon [56];
3. mitigate *malicious attacks on the V2V communication*, that is, if a vehicle in the platoon communicates false acceleration data to its follower, the system must preserve safety and string stability;
4. enable *dynamic platoon configuration*, that is, the system must manage changes in the platoon configuration, such as merging, splitting and reorganization.
5. *isolate* the effects of a compromised vehicle once an attack has been detected.

In addition, in the remainder of the paper, we make the following assumptions:

Assumption 1 (*On-board sensors*)**.** The sensors provide to the i-th vehicle its own velocity, and the position and velocity of the preceding vehicle. As motivated in section 3.2.1, we consider the sensors to provide reliable state measurements. As a point of discussion, we identify the sensors that provide the preceding vehicle's position as the most critical, since the main goal of the controller is to keep a certain inter vehicle distance. An attack on the Lidar sensor, such as the one described in [77], could very well lead to a collision. A viable countermeasure is to ensure significant sensor redundancy to safeguard the reliability of this critical quantity, yet we consider this discussion outside fo the scope of the present work.

Assumption 2 (*Graph topologies*)**.** For the design of the platoon controller, we rely on the predecessor-follower (PF) topology, while for the platoon coordinator, we assume a bidirectional PF topology (see Figure 3.2). We adopt a PF topology in the platoon controller in order to provide stronger safety guarantees, as detailed in section 3.3.2, at the cost of slightly reducing the platooning performance, compared to a bidirectional topology, while under the effect of a malicious attack, as shown in the simulation results in table 3.1 in section 3.4. Extending the method to bidirectional topologies would require a redefinition of the control law and safety filter to account for the additional information flow from following vehicles. This is left as a direction for future work.

Assumption 3 (*Motion constraints*)**.** We consider double integrator dynamics and same actuator limits for each vehicle, i.e. a homogeneous platoon that can be controlled in acceleration. The homogeneity assumption is not a strict requirement, yet we keep it for the sake of clarity, since it simplifies the analysis presented in section 3.3.2. We also neglect external forces such as aerodynamic drag and rolling friction, since we assume an additional control input can always be added on top of our controller's output to compensate for them, as in [57]. Furthermore, even simpler models that use velocity control, have been shown to be sufficient for experimental validation on robotic platforms similar to ours [78]. While considering the vehicle dynamics as a second order system simplifies

the analysis and is common in platooning literature, we acknowledge that real actuators cannot instantaneously change acceleration, i.e. we neglect actuator dynamics. In our experimental implementation, we address this by using a low-level acceleration tracking controller (see Appendix 3.7), which ensures smooth and realistic actuation.

Assumption 4 (*Attacker model*)**.** The attacker can act both at the platooning control level and at the platooning coordinator level. In the former case it can manipulate the information regarding the vehicle's intended motion that is sent to its follower ,i.e. its acceleration. In the latter, the attacker can communicate a fictitious platooning configuration. A compromised vehicle is one whose outbound communication is manipulated by a malicious agent, while its inbound communication remains functional. This assumption is motivated by the possibility that an attacker has breached authentication protocols for a given vehicle, see Section 3.2.1. Notice that multiple vehicles may be attacked at the same time.

Assumption 5 (*Platoon velocity and spacing*)**.** We assume that each vehicle in the platoon knows the desired platoon velocity v^D and the desired inter-vehicle distance d. Moreover, we constrain the vehicles' velocity to positive values and not to exceed a given maximum value.

3.2.1 Attack feasibility and motivations

Concerning the vulnerability of the on-board sensors, global positioning information is susceptible to spoofing attacks on GNSS technologies, this is especially critical for aerospace applications [79]. However, for automotive applications we consider GNSS spoofing not to be safety critical, since collision avoidance will depend on local positioning information provided by sensors such as RADAR, cameras and LIDAR. Concerning the latter, recent works have shown that it is possible to compromise the LIDAR by removing obstacles or pedestrians [77], however this requires a physical device in close proximity to the target vehicle, making such attacks difficult in practice. Therefore, throughout the remainder of this paper, we assume that a potential malicious attacker does not have enough resources to tamper with the vehicle's state measurements.

On the other hand, data transfer within a platoon is usually managed through a Dedicated Short Range Communications (DSRC) in a Vehicular Ad-hoc Network (VANET). According to [80], various types of attacks can be critical to the safety and string stability of platoon formations. Our solution specifically focuses on mitigating attacks at the application layer, in particular, the message manipulation attack (or false data injection). To address these concerns, state-of-the-art security architectures leverage robust cryptographic systems to effectively counter application layer attacks, such as digital signatures and one-time pads in messages. Despite these security methods, practical challenges persist in deploying, implementing, and standardizing such security architectures in VANETs. Moreover, when the threat comes from a trusted insider, like a compromised vehicle with a valid certificate, the problem becomes considerably more challenging. In the following, we propose an attack-resilient platooning, where we assume that the attacker can modify the transmitted information, by exploiting a lack in the security architecture or a valid certificate that allows to override messages.

3.3 Attack-Resilient Platooning

This section introduces the design of the control and coordination layers of our attack-resilient platooning strategy as well as the tailored attack detection policy.

3.3.1 Method Overview

Figure 3.2 depicts the overall algorithm scheme for the $(i+1)$-th Vehicle. Each vehicle has its own local platoon coordinator (detailed in Sec. 3.3.4) and platoon on-board controller (detailed in Sec. 3.3.2). The coordinator decides in what order to place the vehicles, as well as managing adding or removing vehicles from the platoon. The controller is responsible for keeping the desired distance from the preceding vehicle. More in detail, the coordinator provides a valid local platooning topology to the controller, that is, it indicates which vehicle to follow. The $(i+1)$-th Vehicle's controller synthesizes its acceleration, relying on its on-board sensors (ACC mode) and on the acceleration data received from its preceding vehicle (CACC mode). Figure 3.2 also depicts the attack detection module, which provides the controller and the coordinator with σ_{i+1}, a parameter that measures the communication reliability associated with the preceding vehicle. In the platoon controller σ_{i+1} is used to switch off CACC mode and revert back to ACC mode if an attack is detected. In the coordinator σ_{i+1} is used to trigger a topology change in order to isolate a single compromised vehicle. We motivate this design choice with the objective of maximizing platooning performance according to the available information concerning the presence of attacks, while still providing safety guarantees. Indeed during nominal conditions CACC ensures best performance in terms of keeping a fixed inter-vehicle distance, as shown in Table 3.2. When an attack is present but still undetected we ensure collision avoidance guarantees thanks to the safety filter that limits the authority of the feed-forward term in the CACC. Once an attack has been detected we switch off the communication between the compromised vehicle and it's follower. We do this because even if the attack is unable to cause a collision, it may still increase the inter-vehicle distance and consequently fuel consumption. This is shown in Figure 3.10. This strategy applies to each vehicle and therefore is a valid mitigation approach even if all communication channels between vehicles are compromised, as shown in section 3.4 table 3.1. If, however, only one vehicle is affected by an attack, since operating in ACC mode is suboptimal, we then rely on the coordinator to position the compromised vehicle at the end of the platoon. In this way it has no followers, and thus all vehicles resume nominal CACC operation. Notice that we assume only the communication to be affected by an attack, which means that the last vehicle in the platoon, despite having corrupted outbound communication, is still able to receive inbound data from its predecessor and operate normally.

3.3.2 Local platooning controller

We build our platooning controller starting from the ACC module using a modified version of the string-stable longitudinal linear controller proposed in [56]. We design a gain tuning procedure that explicitly takes actuator limits into account and is able to provide both string stability and collision avoidance guarantees. We then add a feed-forward predictive term that is essential to improve platooning performance during nominal conditions. We then consider potential FDI attacks and add a safety filter consisting of control authority constraints to the feed-forward policy. We finally add an ACC-CACC switch to disable the

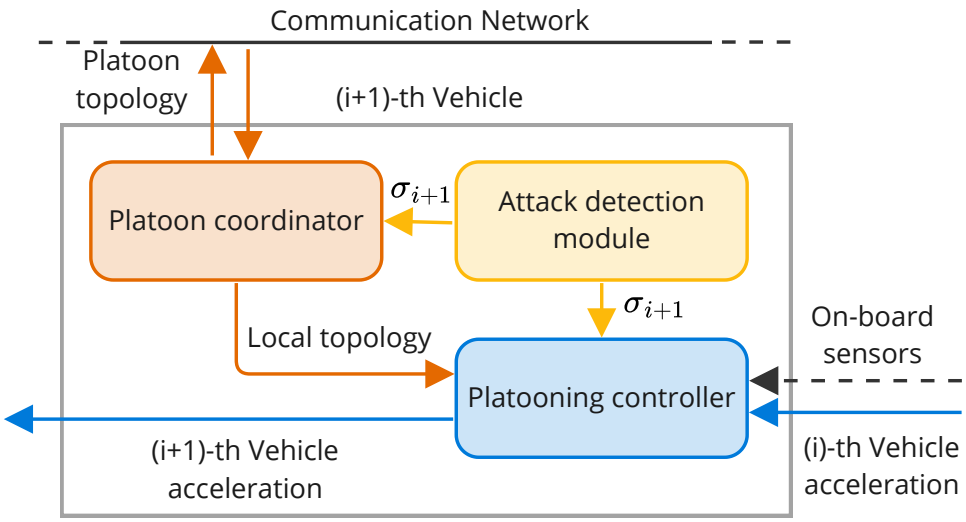


Figure 3.2: Overview of the proposed distributed platooning scheme.

feed-forward action if the communication with the preceding vehicle is deemed unreliable. Figure 3.3 shows an overview of the platoon controller. In deriving the safety conditions (e.g., Eq. 3.13), we assume constant acceleration during braking phases and neglect higher-order dynamics such as actuator lag or jerk as detailed in *assumption 3* in section 3.2. These simplifications allow us to derive interpretable and implementable safety constraints while maintaining robustness in practice, as validated by our simulations and experiments.

ACC Linear controller

In the following section we provide a linear controller to achieve ACC mode platooning, that given the desired platooning velocity v^D and inter-vehicle distance d, takes the vehicle actuation limits into account and provides the range of controller parameters such that string stability and collision avoidance are guaranteed. The linear controller is a modified version of the control law reported in [56][Sec. 4.3]. Let us indicate with $x_{i+1} = \begin{bmatrix} p_{i+1} & v_{i+1} \end{bmatrix}^\top$ the state of the $(i+1)$-th vehicle, where p_{i+1}, v_{i+1} are the $(i+1)$-th vehicle's position and velocity, respectively. The state of the preceding vehicle is x_i. According to Assumption 3, the vehicles can be controlled by acceleration commands[2] and the vehicle actuation limits are known, namely the maximum acceleration $u_{\max} > 0$, maximum braking $u_{\min} < 0$ and maximum vehicle velocity $v_{\max}$. The acceleration-controlled vehicles are represented by the following linear dynamics:

$$\dot{x}_{i+1}(t) = Ax_{i+1}(t) + Bu^{\text{lin}}_{i+1}(t), \forall i = 1, \dots, N-1, t \geq 0 \tag{3.1}$$

[2]We also assume that a reliable vehicle model, which can convert the required acceleration into the vehicle inputs, is available.

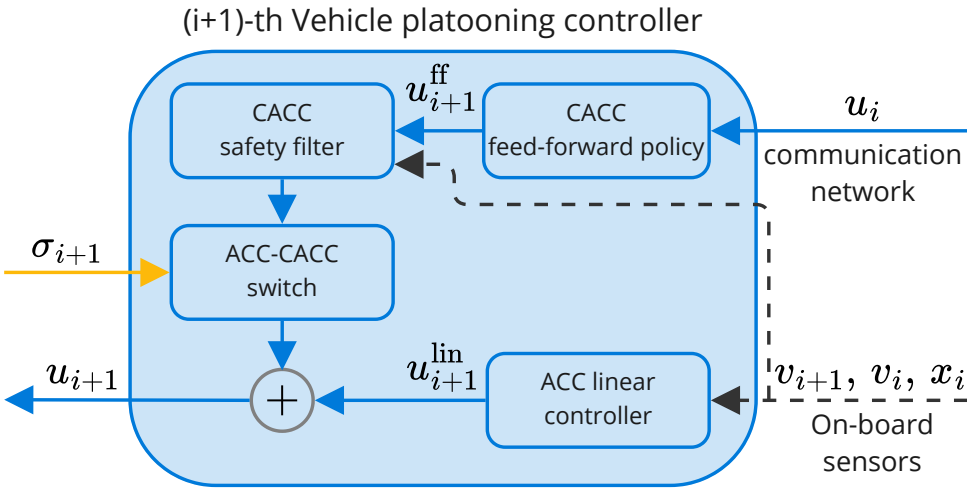


Figure 3.3: Overview of the proposed low-level platooning controller. The CACC feed-forward policy is detailed in equation (3.16), the CACC safety filter corresponds to equation (3.17), the ACC-CACC switch refers to equation (3.18) and the ACC controller is described in equation (3.2).

where $A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$, $B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$. The control action is:

$$\begin{aligned} u^{\text{lin}}_{i+1} = &- k(p_{i+1} - p_i + d) - kh(v_{i+1} - v^D) \\ &- c(v_{i+1} - v_i), \end{aligned} \tag{3.2}$$

where $k, c, h > 0$ are the tuning parameters. To enforce string stability we first derive an expression of the transfer function associated to equation (3.1). Since the objective of the platoon is to travel at a desired speed v^D we define the time-dependent reference position for vehicle i as $p^{\text{ref}}_i = p_i(t=0) + v^D t$ and since vehicle $(i+1)$ should follow vehicle i at a distance of d we define $p^{\text{ref}}_{i+1} = p_i(t=0) + v^D t - d$. We then define $\hat{p}_i = p_i - p^{\text{ref}}_i$ and $\hat{p}_{i+1} = p_{i+1} - p^{\text{ref}}_{i+1}$. In equation (3.2) we rewrite u^{lin}_{i+1} as $u^{\text{lin}}_{i+1} = \ddot{p}_{i+1}$ and substitute the expressions for $p_i = \hat{p}_i + p^{\text{ref}}_i$, $p_{i+1} = \hat{p}_{i+1} + p^{\text{ref}}_{i+1}$ and their derivatives to obtain the expression of the transfer function $G_{i+1}(s)$:

$$G_{i+1}(s) = \frac{\hat{p}_{i+1}(s)}{\hat{p}_i(s)} = \frac{cs + k}{s^2 + (c + hk)s + k}. \tag{3.3}$$

The magnitude of the transfer function $G_{i+1}(s)$ maps the scale factor between the deviation from the reference position—and consequently also reference velocity v^D of the $(i+1)$-th vehicle with respect to its preceding vehicle. Notice that $G_{i+1}(s)$ also applies to relative quantities:

$$\frac{p_{i+2} - p_{i+1} + d}{p_{i+1} - p_i + d} = \frac{\hat{p}_{i+2} - \hat{p}_{i+1}}{\hat{p}_{i+1} - \hat{p}_i} = \frac{\hat{p}_{i+1}(G-1)}{\hat{p}_i(G-1)} = \frac{\hat{p}_{i+1}}{\hat{p}_i}, \tag{3.4}$$

where we dropped the (s) dependency to compact the notation. According to the definition of string stability, we require that $\|G_{i+1}(j\omega)\| < 1$, $\forall\ \omega > 0$, $\forall i = 1, \dots, N-1$. This can be

achieved by ensuring that at least one pole is smaller than the zero. In this way, there is at least one pole that produces a negative slope change before the zero's positive slope increase in the bode diagram, keeping the overall magnitude less than 1. Additionally, we further require that the controlled system is not underdamped, this avoids the follower's position overshooting it's reference during transients and leading to collisions. These two requirements can be written as:

$$\frac{1}{2}(c+hk) - \frac{1}{2}\sqrt{(c+hk)^2 - 4k} < \frac{k}{c} \tag{3.5}$$

$$(c+hk)^2 - 4k > 0. \tag{3.6}$$

Tuning the controller to ensure string stability does not yet guarantee absence of inter-vehicle collisions when the actuators are bounded. We thus need to derive additional conditions on the tuning parameters from the collision avoidance requirements. Let us introduce new state variables $\tilde{v}_{i+1} = v_{i+1} - v_i$ and $\tilde{p}_{i+1} = p_{i+1} - p_i + d$. By defining $\tilde{x} = \begin{bmatrix}\tilde{p} & \tilde{v}\end{bmatrix}^{\mathrm{T}}$, we can subtract the absolute dynamics (3.1) of the i-th vehicle from the $(i+1)$-th vehicle and write the system's relative dynamics as:

$$\dot{\tilde{x}}_{i+1}(t) = A\tilde{x}_{i+1}(t) + B(u_{i+1}(t) - u_i(t)), t \geq 0. \tag{3.7}$$

Note that we also added d to either side of the first row in (3.7). Under the effect of actuator saturation, without explicitly considering braking time, the reachable set of $\tilde{p}_{i+1}$ becomes unbounded. This can be seen from equation (3.7). If $u_{i+1} = u_i = u_{\min}$, that is, when both vehicles exert their maximum braking capabilities, $\dot{\tilde{v}}_{i+1} = 0$. If $\tilde{v}_{i+1} > 0$ (which is the case for a string stable controller) then $\tilde{p}_{i+1}$ will keep growing until a collision occurs, that is, $\tilde{p}_{i+1} > d$. We now provide an expression for the maximum reachable $\tilde{p}_{i+1}$, and provide conditions on k, c, h such that $\tilde{p}_{i+1} \leq d \ \forall \ t \in [0, \infty)$. The controller braking saturation line in the $\tilde{v}$-$\tilde{p}$ plane is

$$-k\tilde{p} - c\tilde{v} - kh(v - v^D) = u_{\min}.$$

We chose k such that the braking saturation line stays always below the point $(0, d) \ \forall \ v \in [0, v_{\max}]$, this ensures that for $\tilde{v} \geq 0$, braking saturation will always occur before a potential collision. This can be written as:

$$k = \frac{-u_{\min}}{d - hv^D}. \tag{3.8}$$

Let us consider an emergency braking scenario where the preceding vehicle suddenly brakes by applying $u_i = u_{\min}$. We define the maximum reachable $\tilde{p}$ as

$$\tilde{p}_{\max} = \tilde{p}^* + \Delta\tilde{p}_{\text{brake}} + \Delta\tilde{p}_{\text{stop}} \tag{3.9}$$

Where $\tilde{p}^*$ is the initial value of $\tilde{p}$ when the $(i+1)$-th vehicle reaches braking saturation, $\Delta\tilde{p}_{\text{brake}}$ is the distance travelled while both vehicles are braking and $\Delta\tilde{p}_{\text{stop}}$ is the distance travelled by the $i+1$-th vehicle once the preceding vehicle has already stopped. In Figure 3.4 we provide a visualization of the absolute velocity v and relative position $\tilde{p}$ during an emergency brake.

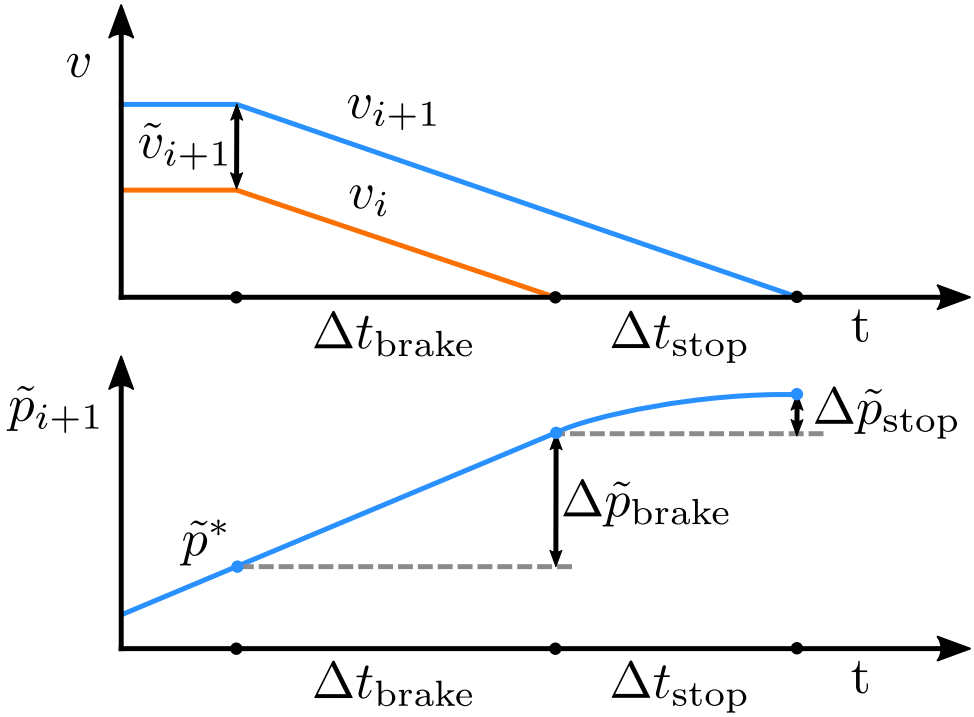


Figure 3.4: Maximum reachable $\tilde{p}$ during an emergency brake.

We define $\tilde{p}^*$ as:

$$\tilde{p}^* = d - \frac{c}{k}\tilde{v}. \tag{3.10}$$

Note that this is a conservative estimate since saturation will actually occur for lower values of $\tilde{p}$ for $v > 0$. $\Delta\tilde{p}_{\text{brake}}$ is defined as $\Delta\tilde{p}_{\text{brake}} = \tilde{v}\Delta t_{\text{brake}}$, where Δt_{brake} is the time it takes the preceding vehicle to stop, since $v_{i+1} = v_i - \tilde{v}$ we can write $\Delta\tilde{p}_{\text{brake}}$ as:

$$\Delta\tilde{p}_{\text{brake}} = \tilde{v}\frac{v_{i+1} - \tilde{v}}{-u_{\min}}. \tag{3.11}$$

Once the preceding vehicle has stopped, the $i+1$-th vehicle has velocity $v_{i+1} = \tilde{v}$, thus $\Delta\tilde{p}_{\text{stop}}$ is defined as:

$$\Delta\tilde{p}_{\text{stop}} = \frac{1}{2}\frac{\tilde{v}^2}{-u_{\min}} \tag{3.12}$$

By substituting equations (3.10), (3.11) and (3.12) in equation (3.9) we can write $\tilde{p}_{\max}$ as:

$$\tilde{p}_{\max} = d - \left(\frac{c}{k} - \frac{v}{-u_{\min}}\right)\tilde{v} - \frac{1}{2}\frac{\tilde{v}^2}{-u_{\min}} \tag{3.13}$$

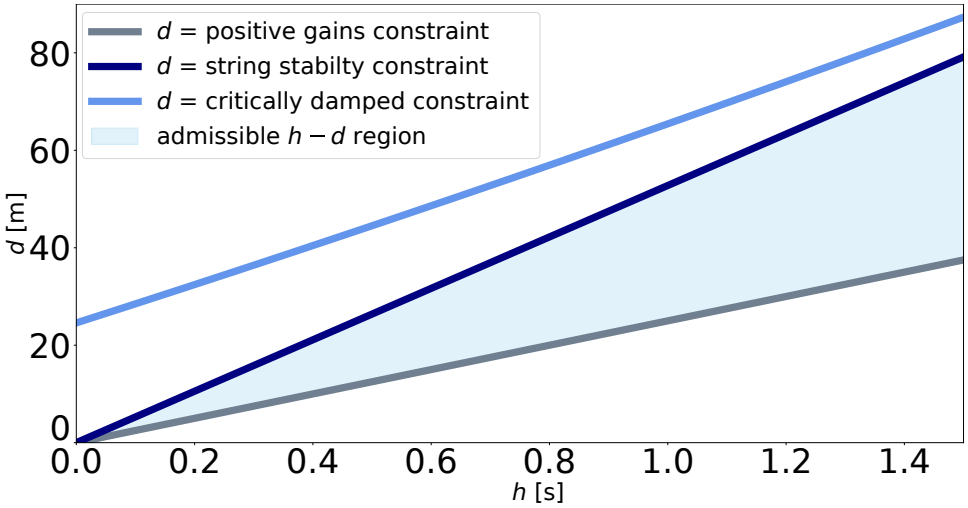


Figure 3.5: Gain tuning graph showing the combinations of d and h that guarantee string stability and collision avoidance for vehicle parameters $v_{\max} = 100$ (km/h), $u_{\min} = -0.8g$, $u_{\max} = 0.5g$, and platooning velocity $v^D = 90$ (km/h).

Notice that $\tilde{p}_{\max}$ is monotonically increasing with respect to v, so by requiring that $\tilde{p}_{\max} \leq d$ for $v = v_{\max}$, we guarantee collision avoidance. This can be achieved by choosing c as:

$$c = \frac{v_{\max}}{d - hv^D}. \tag{3.14}$$

We now have expressions for k and c as a function of d and h, so to conclude the tuning procedure we represent conditions (3.5), (3.6) and $k, c, h > 0$ in the h-d plane. By choosing a valid combination of h and d we can guarantee string stability and collision avoidance while accounting for saturation constraints. Figure 3.5 shows the possible valid h-d combinations for vehicle parameters $v_{\max} = 100$ (km/h), $u_{\min} = -0.8g$, $u_{\max} = 0.5g$, and platooning velocity $v^D = 90$ (km/h). Note that a typical passenger vehicle would be able to accelerate at $0.5g$ only at low speeds, yet, for the sake of simplicity, we take conservative bounds in order consider actuator limits that don't depend on the velocity. It is worthwhile mentioning that for a given d the closer h is to the $k, c > 0$ line, the higher the gains will be, for $h \to \frac{d}{v^D}$ $k, c \to \infty$. Yet it is still possible to chose very small values of d, in the range $5 - 10$m such to provide high aerodynamic and fuel efficiency benefits to the platoon.

Feed-forward policy and safety filter

To enhance performance, we rely on V2V communication to achieve CACC platooning. We introduce a communication-based feed-forward term u^{ff} and define the $(i+1)$-th vehicle's acceleration as $u_{i+1} = u^{\text{lin}}_{i+1}(t) + u^{\text{ff}}_{i+1}(t)$. We can now rewrite Eq. (3.7) as follows:

$$\dot{\tilde{x}}_{i+1}(t) = A\tilde{x}_{i+1}(t) + B(u^{\text{lin}}_{i+1}(t) + u^{\text{ff}}_{i+1}(t) - u_i(t)), \tag{3.15}$$

The relevance of u^{ff}_{i+1} can be clearly appreciated by viewing the preceding vehicle's acceleration u_i as a disturbance, indeed if a reliable measure of the latter is available, we

can design u_{i+1}^{ff} to compensate for it and even improve the platooning performance. In this section, to derive the safety conditions, we need to make no assumptions on the specific choice of feed-forward policy, it could be based on MPC, sliding mode controller [81], or even a simpler policy such as $u_{i+1}^{\text{ff}}(t) = u_i(t)$. We refer to the generic feed-forward policy as:

$$u_{i+1}^{\text{ff}}(t) = \pi(u_i(t)). \tag{3.16}$$

Note that in (3.16) π may also depend on the i-th and $(i+1)$-th vehicles states, yet we highlight the dependency on $u_i(t)$ because this data is strictly necessary for any kind of feed-forward policy and needs to be shared over the communication network. A well designed policy should improve the overall performance of the platoon, yet if the $(i+1)$-th vehicle receives compromised u_i acceleration data, a malicious entity could exploit u_{i+1}^{ff} to take control of the $(i+1)$-th vehicle and potentially lead it to unsafe states.

To guarantee collision avoidance under corrupted u_i acceleration data we implement two additional conditions on u_{i+1}^{ff}, leading to the following *safety filter* definition:

$$u_{i+1}^{\text{ff}} = \begin{cases} 0 & \text{if } \tilde{p}_{i+1} \geq d - \frac{c}{k}\tilde{v}_{i+1} \quad (3.17a) \\ \hat{u}_{\max}^{\text{ff}}(v_{i+1}) & \text{if } \pi(u_i) \geq u_{\max}^{\text{ff}}(v_{i+1}) \quad (3.17b) \\ \pi(u_i) & \text{otherwise} \quad (3.17c) \end{cases}$$

Where we define $u_{\max}^{\text{ff}}(v_{i+1}) = k(\alpha d + h(v_{i+1} - v^D))$, $\alpha \in [0,1]$. Condition (3.17a) essentially triggers an emergency braking maneuver and ensures that the collision avoidance guarantees provided by the ACC linear controller are still verified even in the presence of a compromised u_i measure. This is because the feed-forward action is disabled if the current state in the $(\tilde{v}, \tilde{p})$ plane is above the linear controller braking saturation line for $v_{i+1} = 0$. In this case $u_{i+1} = u_{i+1}^{\text{ff}} + u_{i+1}^{\text{lin}} = u_{\min}$. Condition (3.17b) is added to limit the number of times the emergency braking condition (3.17a) is triggered. By setting the maximum value of u_{i+1}^{ff} to $k(\alpha d + h(v - v^D))$, we ensure that $u_{i+1}^{\text{lin}} + u_{i+1}^{\text{ff}} \leq 0 \ \forall \ \tilde{p} \geq \alpha d - \frac{c}{k}\tilde{v}$, i.e. under a compromised u_{i+1}^{ff}, the state of vehicle $(i+1)$ will remain below the line described in equation (3.10) and condition (3.17a) will not be activated. This means that in order to trigger an emergency maneuver, an attacker needs to compromise both the vehicle $(i+1)$, inducing it to accelerate, and the vehicle i, inducing it to brake. Values of $\alpha < 1$ will introduce an additional safety margin, since the minimum inter-vehicle distance under a compromised u^{ff} will be $(1-\alpha)d$, at the expense of more stringent limitations on u^{ff}. Note that even if $\alpha = 1$ collision avoidance is still guaranteed thanks to condition (3.17a). In Figure 3.6 we show an example state trajectory in the $\tilde{p}$-$\tilde{v}$ plane for $\alpha = 1$.

3.3.3 Attack detection module

The platooning controller presented in the previous sections is robust to adversarial attacks, in particular, it can provide collision avoidance guarantees even when the communication channel between all vehicles is compromised. However, a malicious entity that carries out an attack at the communication network level could still compromise the performance of the platoon. For example by injecting false acceleration data, it could increase the inter-vehicle distance or induce large oscillations in the velocity. To mitigate such an attack we rely on an *attack detection module*, that runs on each vehicle, constantly evaluates the

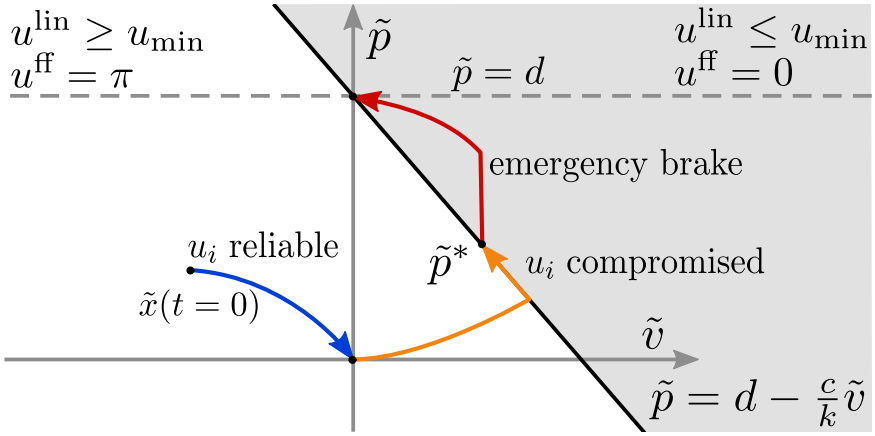


Figure 3.6: State trajectory example in the $\tilde{p}$-$\tilde{v}$ plane. From an initial condition $\tilde{x}(t=0)$ the system converges to the origin while a reliable measure of u_i is available (blue arrow). Subsequently a malicious attack communicates false acceleration data and induces vehicle $(i+1)$ to approach its predecessor, yet no collision occurs thanks to additional constraints that limit the control authority of u^{ff}_{i+1} (orange arrow). Finally, the leading vehicle performs an emergency brake (red arrow) but collision is avoided thanks to the linear controller design process that accounts for actuation saturation (shaded area).

reliability of u_i, and outputs a variable $\sigma_{i+1} \in \{0, 1\}$, where $\sigma_{i+1} = 1$ if vehicle $i+1$ trusts it's predecessor's u_i information and $\sigma_{i+1} = 0$ otherwise. We thus redefine the feed-forward controller in (3.17) as:

$$u^{\text{ff}}_{i+1} = \sigma_{i+1} u^{\text{ff}}_{i+1}, \tag{3.18}$$

that is, we disable the feed-forward action u^{ff}_{i+1} if $\sigma_{i+1} = 0$ in order to protect against attacks aimed at degrading the performance of the platoon. Note that this strategy is intrinsically robust to DoS attacks, as this is equivalent to setting $\sigma_{i+1} = 0$.

There are multiple ways to estimate the trustworthiness of the data received from the preceding vehicle, such as checking consistency between communication and on-board sensors, or evaluating and recognizing a performance degradation. More sophisticated techniques can be designed based on particle filters [18], sliding mode observers [82] or set membership estimation [83]. In the present work we opted to implement an observer based approach due to its simplicity. Additionally, as in [83] and in the present paper, in vehicular platooning it is typical to assume that a malicious attack is most likely to occur on the communication channels, rather than on the on-board sensors. Since vehicles can rely on the latter to estimate the current state, we exploit this peculiarity to separate the state estimation from the attack detection functionality and design a tailored Kalman filter-based detector module.

Since each vehicle trusts its own sensors and actuators, the effects of a compromised u_i measure are visible when estimating $\tilde{v}_{i+1}$. As pointed out in [84], the filter gain K reaches an equilibrium after a certain number of time steps based solely on the ratio between the process noise matrix Q and the measurement noise n, it is thus reasonable to evaluate the equilibrium gain offline and use it for state estimation. A constant gain Kalman filter that

estimates $\tilde{v}_{i+1}$ is represented by following equations:

$$\tilde{v}_{i+1}^{-} = \hat{\tilde{v}}_{i+1}^{k} + dt(u_{i+1}^{k} - u_{i}^{k}) \tag{3.19}$$

$$\hat{\tilde{v}}_{i+1}^{k+1} = (1-K)\tilde{v}_{i+1}^{-} + K\tilde{v}_{i+1}^{k+1}, \tag{3.20}$$

Where k indicates the time instant, dt is the time step, $\tilde{v}_{i+1}^{-}$ is the expectation, $\hat{\tilde{v}}_{i+1}$ is the estimated relative velocity and $\tilde{v}_{i+1}^{k+1}$ is the measured relative velocity. As discussed in [84], the effect of a malicious attack that remains undetected is bounded by the Kalman filter properties, i.e., by the value of K. In vehicular platooning this is clear by looking at equation (3.19), since u_i is the only quantity that can be affected by a malicious attacker, for $K \to 1$ the estimated $\hat{\tilde{v}}_{i+1}$ will predominantly be affected by the measured state $\tilde{v}_{i+1}$. Conversely if $K \to 0$ the predominant contribution to the $\hat{\tilde{v}}_{i+1}$ update will be based on $\tilde{v}_{i+1}^{-}$ and the estimated relative velocity will be more sensitive to a malicious attack on u_i. On a real system the available sensors are typically sufficiently accurate and provide a reliable measure of $\tilde{v}_{i+1}$ without the need to run the Kalman filter described in equations (3.19) and (3.20), thus by choosing lower values of K it can be focused towards verifying the trustworthiness of u_i, rather than providing a good estimate of $\tilde{v}_{i+1}$. This is achieved by evaluating the residual r, defined as:

$$r_{i+1}^{k} = |\hat{\tilde{v}}_{i+1} - \tilde{v}_{i+1}|. \tag{3.21}$$

We then consider the communication channel to be under attack if r_{i+1}^{k} exceeds a certain threshold value $\bar{r}$. Setting the values K and $\bar{r}$ entails striking a compromise between reactiveness and false detections. In particular lower values of K also increase sensitivity to noise on u_i, u_{i+1} and $\tilde{v}_{i+1}$, and $\bar{r}$ needs to be calibrated in order to avoid missed detections while minimizing false positives.

3.3.4 Platoon coordinator

An additional mechanism to mitigate attacks can be implemented by relying on the platoon coordinator. In fact, the platoon coordinator can reorganize the platoon to isolate the effects of a compromised vehicle, i.e. a vehicle whose outbound communication is unreliable. We leverage the PF topology and rearrange the platoon such that the compromised vehicle is in the last position, in this way it has no follower to communicate with. In addition it can handle merging and splitting requests.

We propose a distributed strategy that guarantees the successful execution of the aforementioned operations, even in the presence of a single malicious vehicle that communicates false information at the platoon coordinator level.

Each vehicle, numbered i, needs to build a vector δ_i, which includes information about the preceding and the following vehicles, and broadcasts it to the network. The broadcasting operation can be done in a distributed fashion by relying on existing methods [85].

The entries $\delta_i = \begin{bmatrix} \mathrm{id}_{i,i-1} & \mathrm{id}_{i,i+1} \end{bmatrix}$ correspond to a unique number associated with the preceding and the following vehicle (no preceding or no following vehicle is indicated with $\mathrm{id}_{i,i-1} = 0$ and $\mathrm{id}_{i,i+1} = 0$, respectively). Additionally, when the trustworthiness value, provided by the estimator module, becomes $\sigma_i = 0$, to indicate a severed communication link, we set $\mathrm{id}_{i,i-1} = 0$.

We define *correct platooning conditions* when: *1.* there is only one vehicle without the preceding vehicle (the leader) and only one without the following vehicle (the last vehicle), *2.* the matrix $\mathcal{D} = [\delta_1, \delta_2, \dots, \delta_N]^\mathsf{T}$ has to describe a connected topology and finally, *3.* it has to be consistent, that is, if vehicle A is the preceding vehicle of B, then B is the following vehicle of A.

If the *correct platooning conditions* are violated, the platoon coordinator comes into play. We inspect three cases of interest that require the intervention of the platooning coordinator: *(i) reorganization to isolate a compromised vehicle*: the i-th vehicle receives unreliable acceleration data, and sets $\mathrm{id}_{i,i-1} = 0$, the correct platooning conditions are violated, since there are two vehicles with $\delta_i = \begin{bmatrix} 0 & \mathrm{id}_{i,i+1} \end{bmatrix}$. *(ii) merging request*: a vehicle wants to join the platoon. In this case the vehicle sends a request to the network by communicating its vector $\delta_i = \begin{bmatrix} 0 & 0 \end{bmatrix}$ and a new row is added to the $\mathcal{D}$ matrix; *(iii) splitting request*: a vehicle wants to leave the platoon. The vehicle decides to not communicate with the network anymore and its row is removed from the $\mathcal{D}$ matrix. The preceding and the following vehicle of the departing agent remain without a following and a preceding vehicle, respectively. This leads to a violation of the conditions on $\mathcal{D}$, that is, two vehicles with $\delta_i = \begin{bmatrix} 0 & \mathrm{id}_{i,i+1} \end{bmatrix}$ and two vehicles with $\delta_i = \begin{bmatrix} \mathrm{id}_{i,i-1} & 0 \end{bmatrix}$.

We assumed that each vehicle in the system has a copy of the matrix $\mathcal{D}$. The platooning coordinator on each vehicle, constantly checks the correctness of the $\mathcal{D}$ matrix, that is, the correctness of the platooning conditions. When one or more conditions are violated each vehicle solves the following optimization problem:

$$\begin{aligned} &\arg\max_{\mathcal{D}^\star} \sum_{i=1}^{N} f(\mathrm{id}_{i,i-1}, \mathrm{id}^\star_{i,i-1}) + f(\mathrm{id}_{i,i+1}, \mathrm{id}^\star_{i,i+1}) \\ &\text{s.t. } \textit{correct platooning conditions}, \end{aligned} \tag{3.22}$$

where $f(a, b) = 1$, if $a = b$, $f(a, b) = 0$, if $a \neq b$. Notice that the optimization cost function weights the difference between the old configuration $\mathcal{D}$, which violates the *correct platooning conditions*, and the new configuration $\mathcal{D}^\star$. This means that the new solutions will differ from the previous graph topology as little as possible to ensure *correct platooning conditions*. The algorithm scalability in general may be an issue, however, for our application there are no stringent computation time constraints, since the platooning coordinator does not affect safety nor stability.

Example 1. Figure 3.7 depicts the possible solutions of (3.22) for the three cases: platoon reorganization, merging request and splitting request.

Platoon reorganization

Let us consider Figure 3.7-a, the case where the trustworthiness value $\sigma_3 = 0$. In this particular condition, the optimal solution of (3.22) would reattach the removed link, yet since that communication channel is now considered unreliable we remove the latter from the admissible solutions. The solution of (3.22) is thus unique, placing vehicle 2, which communicates suspiciously, in the last position, where it can no longer influence any other vehicle. Notice that despite the attack on the communication channel has been isolated, the vehicle itself is still part of the platoon, since we assume it is still able to receive inbound communication from it's predecessor and remains cooperative. In this scenario there is

no external way to assess whether the attack on the last vehicle has been resolved or not. Nevertheless, the platoon has now been alerted and vehicle 2 is now aware that its own outbound communication has been compromised. In this situation the compromised vehicle could initiate a full system check or implement other ad hoc strategies, yet we consider this to be out of scope of the present paper.

$$\delta = \begin{bmatrix} 0 & 2 \\ 1 & 3 \\ 0 & 4 \\ 3 & 5 \\ 4 & 0 \end{bmatrix}, \quad \delta^\star = \begin{bmatrix} 5 & 2 \\ 1 & 0 \\ 0 & 4 \\ 3 & 5 \\ 4 & 1 \end{bmatrix}$$

Merging case

Let us consider Figure 3.7-b, the case where vehicle 6 wants to join the platoon. By solving (3.22) we get two possible solutions. In fact, since the optimization problem minimizes the changes in the platooning configuration, vehicle 6 could take the place of the leader or of the last vehicle. To resolve the ambiguity, a simple rule could be to not change the platoon leader, hence solution $\mathrm{id}^{\star,1}$ would be preferred.

$$\delta = \begin{bmatrix} 0 & 2 \\ 1 & 3 \\ 2 & 4 \\ 3 & 5 \\ 4 & 0 \\ 0 & 0 \end{bmatrix}, \quad \delta^{\star,1} = \begin{bmatrix} 0 & 2 \\ 1 & 3 \\ 2 & 4 \\ 3 & 5 \\ 4 & 6 \\ 5 & 0 \end{bmatrix}, \quad \delta^{\star,2} = \begin{bmatrix} 6 & 2 \\ 1 & 3 \\ 2 & 4 \\ 3 & 5 \\ 4 & 0 \\ 0 & 1 \end{bmatrix}$$

Splitting case

Let us consider Figure 3.7-c, the case where vehicle 3 wants to leave the platoon. Also in this case, by solving (3.22), we get two possible solutions, one is simply to link the gap left by vehicle 3, the other solution instead, is to promote vehicle 4 as the leader and move vehicle 2 to the end of the platoon. Similarly to the merging case the $\delta^{\star,1}$ solution would be preferred, since the platoon leader should not change.

$$\delta = \begin{bmatrix} 0 & 2 \\ 1 & 0 \\ 0 & 5 \\ 4 & 0 \end{bmatrix}, \quad \delta^{\star,1} = \begin{bmatrix} 0 & 2 \\ 1 & 4 \\ 2 & 5 \\ 4 & 0 \end{bmatrix}, \quad \delta^{\star,2} = \begin{bmatrix} 5 & 2 \\ 1 & 0 \\ 0 & 5 \\ 4 & 1 \end{bmatrix}$$

■

Security issues

Until now we did not discuss how the platoon coordinator can manage the case where a vehicle in the network broadcasts false topology information. Let us consider the case where the i-th vehicle is compromised and communicates a false δ_i. This kind of attack is easily detectable in our proposed framework. In fact, the platoon coordinator constantly checks the correctness of the D matrix, and in particular its consistency, as we described before

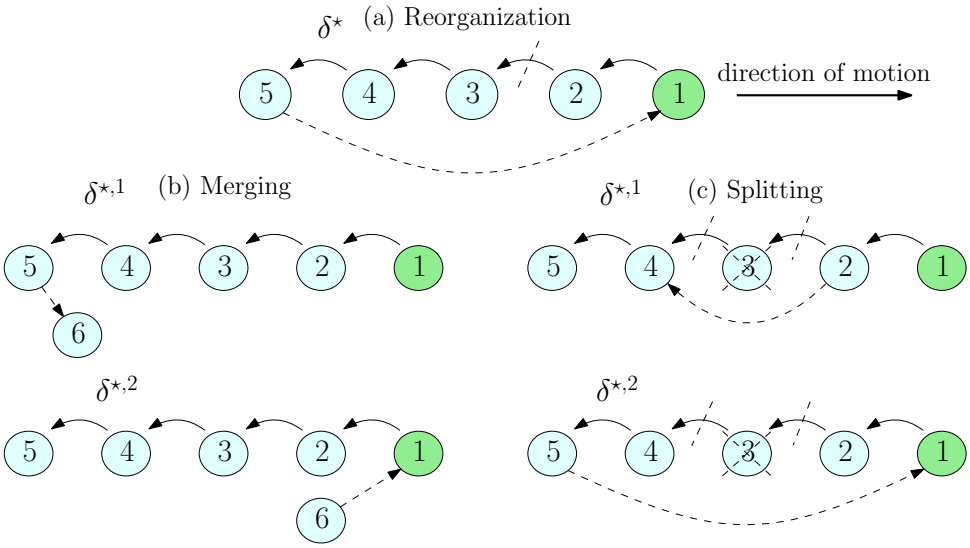


Figure 3.7: Three example of the platoon coordinator intervention. In (a) we depicted the platoon reorganization, in (b) the merging case, in (c) we represented the splitting case. The dashed arrows are obtained by solving (3.22).

(Property *3.*). In case of inconsistencies it means that a δ_i message has been manipulated or a trustworthiness value σ_i dropped to 0. If an attacker communicates a false configuration to the platoon, assuming $N > 3$, we have to distinguish two cases: *1)* If even a single entry of δ_i is changed with another id, there are at least other two vehicles that are in contrast with the information provided by the attacker. By trusting the majority we can identify the attacker and override its messages; *2)* A subtler attack is to communicate to the neighbours to not have a preceding vehicle, emulating a drop in the preceding vehicle's trustworthiness. Since the σ values are not shared between the vehicles, it is not trivial to recognize it as an attack. However, we argue that this is not a significant threat, in fact it would involve a single rearrangement of the platoon, after that, the attacker would no longer have a chance to rearrange the platoon any further, since it would become the new leader of the platoon and thus would not have a predecessor in any case.

3.4 Simulation results

In this section we compare our low-level controller against the MPC-based platooning controller [57] and the reachability-based robust control approach [76]. Notice that for these simulations we don't include any attack-detection components in order to highlight the safety guarantees of the CACC-ACC controller even if an attack remains undetected.

The vehicle parameters are set as in figure 3.5, i.e. actuator limits are $v_{\max} = 100$km/h, $u_{\min} = -0.8g$, $u_{\max} = 0.5g$ for full scale passenger vehilces. All controllers try to keep an inter-vehicle distance of $d = 6$m while traveling at a target velocity of $v_D = 90$km/h. For our controller we select the lowest value of h compatible with these parameters. This yields $k = 2.457$ $h = 0.112$ $c = 8.69$. The CACC policy $\pi(u_i)$ is chosen as the relatively simple expression $u_{i+1}^{\text{ff}} = u_i$. For the MPC in [57] we manually tuned the gains until adequate

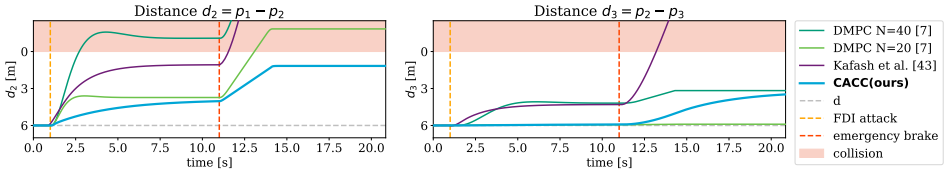


Figure 3.8: Comparison between different platooning controllers on a platoon of 3 vehicles. From $t = 1$s the leader is subject to an additive attack $u_0 = u_0 + u_{\min}$ that induces it to brake, while the first follower is subject to an additive attack equal to $u_1 = u_1 + u_{\max}$ that induces it to accelerate. The last vehicle is not affected by the attack. Additionally, at $t = 11$s, the leader performs an emergency braking maneuver, i.e. $u_0 = u_{\min}$. We show the distance between the leader and the first follower d_2 and between the first and second follower d_3. A collision occurs when the distance is less than 0.

performance in the nominal case is reached. To highlight how the choice of parameters may affect the performance, we show the results for a time horizon of $N = 20$ steps and for $N = 40$ steps, where the simulation time discretization is $dt = 0.05$s. Concerning [76], the method needs symmetric actuator bounds to work, so we set them equal to the maximum braking capacity, as this is a more conservative choice. In all simulations we consider that the platoon starts from steady state conditions, $v_i = v^D$, $\forall\, i \in \{1, ..., n\}$ and $\tilde{p}_i = 0\, \forall\, i \in \{2, ..., n\}$. Where n is the number of vehicles in the platoon.

As an illustrative example we consider a platoon of $n = 3$ vehicles where the leader is subject to an additive attack $u_0 = u_0 + u_{\min}$ that induces it to brake, while the first follower is subject to an additive attack equal to $u_1 = u_1 + u_{\max}$ that induces it to accelerate. Furthermore, after some time the leader performs an emergency brake, i.e. $u_0 = u_{\min}$. This could happen if the leader encounters an obstacle on the road such as another human-driven traffic participant, or if it is affected by a more severe sensor-tampering attack such as Lidar spoofing [77]. Figure 3.8 shows the distance between the first follower and the leader $d_2 = p_1 - p_2$ and between the second and first follower $d_3 = p_2 - p_3$ for the different methods.

To provide more statistical relevance to our comparison, we have conducted an extensive simulation study. We now consider a platoon of $n = 11$ vehicles. We simultaneously subject all the vehicles in the platoon to an attack that replaces the true inbound acceleration signal from the predecessor with either a constant value $u_i(t) = c_i$, a sinusoidal signal defined as $u_i(t) = a_i \sin(\phi_i + f_i 2\pi t)$ or a random value $u_i(t) = e_i \in [u_{\min}, u_{\max}]$ taken from a uniform random distribution, which is then filtered using a first order filter with time constant τ_i. For each method we run 1000 simulations of 100s for each attack type. We randomize the attack parameters for each vehicle in the platoon in every simulation run, ensuring that the attack signal remains within the actuation limits, i.e. $u \in [u_{\min}, u_{\max}]$. Furthermore at $t = 100$s the leader performs an emergency brake. The results are presented in table 3.1.Note that the table shows the aggregated data for all vehicles across all simulation runs, since we did not notice any significant difference among vehicles.

By looking at the data we can see how the MPC in [57] can be tuned to perform adequately for most attack types. However, since it does not have explicit safety guarantees, this process would require lengthily *a posteriori* analysis. Interestingly, a shorter time horizon improves the safety margin both during the attack and during the emergency brake. This is because of the terminal set constraint in [57]. Which forces the vehicle to

reach the desired position at the end of the MPC horizon. A shorter horizon entails for a more reactive controller.

Concerning [76], we notice that during the attack phase of the simulation it performs well, outperforming its competitors in most cases. However, the emergency brake scenario falls outside the assumptions of the method, leading to collisions in our simulations. Indeed the method relies in bounding the communication-based contribution of the controller, while the sensor-based contribution is not bounded. In practice, however, they both contribute to the same actuation pool. An emergency brake can be considered as the result of an infinite negative additive attack that saturates the braking capabilities, violating the method's assumptions. Furthermore, computing the distance between vehicles when they have all reached a standstill, yields a solution of the form $d_i = d_{i+1} - \delta$, where $\delta > 0$. This means that the inter-vehicle distance keeps decreasing moving from the tail to the head of the platoon. So for an arbitrarily long platoon a collision is unavoidable.

Our method achieves 100% safety under all tested attack conditions, including during emergency braking scenarios, while maintaining comparable performance to the baselines. These results validate the effectiveness of the safety filter introduced in Equation 3.17. While these findings are promising, we note that they are based on extensive but simulated scenarios. Real-world deployment may introduce additional uncertainties not captured in this study, which we aim to address in future work.

	mean dist. [m]	std. dev. [m]	max dist. [m]	min dist. [m]	safe runs (attack)	safe runs (brake)
Attack type: constant						
DMPC N=40	6.07	4.25	-2.79	44.50	91.35%	8.16%
DMPC N=20	**6.00**	1.31	3.76	8.21	**100%**	53.98%
Kafash et al.	6.03	**1.12**	1.19	10.55	**100%**	0%
CACC (ours)	**6.01**	1.15	**4.00**	**7.98**	**100%**	**100%**
Attack type: sinusoidal						
DMPC N=40	**6.00**	0.78	-2.79	33.03	**100%**	0.01%
DMPC N=20	6.25	0.86	0.25	16.40	**100%**	27.50%
Kafash et al.	**6.00**	**0.26**	**4.93**	**7.27**	**100%**	0%
CACC (ours)	**6.01**	0.37	4.70	**7.27**	**100%**	**100%**
Attack type: random						
DMPC N=40	6.04	1.17	0.02	15.70	**100%**	0%
DMPC N=20	6.02	0.27	4.23	8.91	**100%**	49.57%
Kafash et al.	**6.00**	**0.13**	**5.17**	**6.81**	**100%**	0%
CACC (ours)	6.07	0.19	**5.17**	7.04	**100%**	**100%**

Table 3.1: Simulation study results under different attack types. For each method and for each attack type we performed 1000 simulations of 100 sec. For these simulations we disable the attack detection module to show the inherent robustness properties of the platooning controller. At the end of each simulation the leader performs an emergency brake. The mean represents the time-averaged value across all vehicles, and the standard deviation quantifies the temporal variation around this mean. Safe runs (attack) refers to the percentage of collisions over all vehicles when the attack is active, while safer runs (brake) refers to the collisions during the emergency brake. DMPC refers to [57], and Kafash et al. to [76].

3.5 Experimental results

We tested our algorithm by using four scaled-down car-like robots shown in Figure 3.1. We performed three experiments aimed at verifying different properties of the proposed platooning strategy: *Experiment 1* shows the baseline platooning performance of the linear controller and the improvements that an additional feed-forward strategy can bring; *Experiment 2* demonstrates the collision avoiding guarantees of the proposed strategy under a FDI attack; *Experiment 3* shows the attack detection and mitigation strategies, as well as threat isolation by rearranging the platoon. Videos of the experiments are available at [75]. The details of the experimental set-up are described in the appendix. Concerning the communication protocol, we implemented the V2V communication layer over a standard wireless local area network (Wi-Fi) using ROS (Robot Operating System) messaging. While this setup does not replicate the exact physical and MAC-layer behavior of DSRC or C-V2X, it captures the essential characteristics of V2V communication relevant to our study—namely, message latency, packet loss, and the ability to inject or filter malicious data.

Experiment 1

The first experiment we conducted is aimed at highlighting the string stability property of the proposed sensor-based platooning algorithm and showing how a communication-based feed-forward term can improve performance.

We chose the linear controller gains as described in section 3.3.2, with the following vehicle actuation limits: $u_{\max} = +1\text{m/s}^2, u_{\min} = -1\text{m/s}^2, v_{\max} = +1.4\text{m/s}, v^D = 1\text{m/s}$. These limits were set according to the hardware capabilities. We then selected a target inter-vehicle distance of 0.5m and the lowest admissible value of h, see Figure 3.5 for reference. The resulting gains are $k = 3.45$, $h = 0.21$ and $c = 4.83$. The gain selection code is available on the GitHub repository [74]. We define the feed-forward policy simply as $u^{\text{ff}}_{i+1}(t) = u_i(t)$. To induce oscillations in the platoon and highlight the string stability properties we provide the leader with a sinusoidal velocity reference, initially without any feed-forward action. The experiment results are shown in Figure 3.9. Note that activating u^{ff} should not result in perfect relative distance tracking, indeed u^{ff}_{i+1} will compensate for the disturbance coming from u_i, but the external damping term $-kh(v_{i+1} - v^D)$ of the linear controller will still enforce string stability, damping out absolute velocity oscillations around v_D. If perfect relative distance tracking is desired, u^{ff}_{i+1} should also compensate for the absolute damping term, i.e. $u^{\text{ff}}_{i+1} = u_{i+1} + kh(v_{i+1} - v^D)$, this can be seen from the simulations we performed in the GitHub repository [74]. We also repeated the experiment and collected the distances data in table 3.2. The string stability property of both algorithms can be seen from the reducing standard deviation values along the platoon, e.g. Vehicle 3 has smaller std dev than Vehicle 2. The performance increase when activating CACC mode is highlighted bold. The mean gets closer to $d = 0.5\text{m}$, the std dev reduces and the min and max value get closer to $d = 0.5\text{m}$. Notice that "Vehicle $i + 1$" refers to the distance $p_i - p_{i+1}$.

Experiment 2

The second experiment shows the collision avoiding properties of the proposed algorithm in the presence of an undetected FDI attack. While u^{ff} is activated we simulate a FDI attack between the leader and the second vehicle in the platoon and disable the attack detection

	Vehicle 2		Vehicle 3		Vehicle 4	
	ACC	CACC	ACC	CACC	ACC	CACC
mean [m]	0.45	**0.46**	0.52	**0.51**	0.53	**0.51**
std dev [m]	0.09	**0.06**	0.06	**0.04**	0.06	**0.04**
max [m]	**0.64**	0.67	0.69	**0.63**	0.74	**0.67**
min [m]	0.25	**0.30**	0.37	**0.40**	0.38	**0.41**

Table 3.2: Distances for ACC and CACC platooning with leader executing a sinusoidal velocity profile. Notice that "Vehicle $i+1$" refers to the distance $p_i - p_{i+1}$. The data is collected over a 60s period for each control algorithm. The mean represents the time-averaged value, and the standard deviation quantifies the temporal variation around this mean.

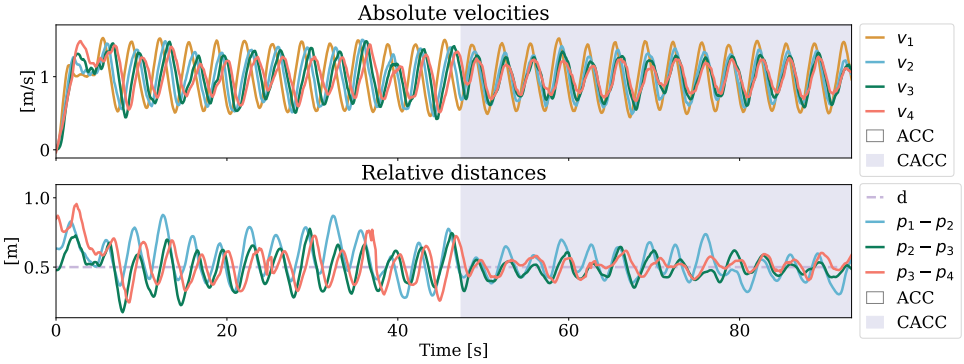


Figure 3.9: Experiment 1: the leader of the platoon follows a sinusoidal velocity reference, while other vehicles initially rely only on the linear sensor-based feedback controller. String stability properties are evident from the reduction in amplitude of the absolute velocity oscillations going towards the tail of the platoon. When the feed-forward action is activated (shaded area) the platooning performances increase, as can be seen in the reduction of relative position and velocity oscillations.

module. Instead of the real u_1 value the leader communicates alternating values $u_1(t) = u_{\max}$ or $u_1(t) = u_{\min}$ every 5s. The experiment results are shown in Figure 3.10. Collision avoidance properties are visible from the fact that even when the leader communicates $u_1(t) = u_{\max}$ and vehicle 2 is tricked into accelerating and gets closer, no collision occurs thanks to constraint (3.17a) being activated, that is, the relative distance remains $p_i - p_{i+1} \geq 0$. However, we notice that without an attack detection module the attacker is still able to manipulate the distance, degrading platooning performance.

Experiment 3

In the last experiment we introduce the attack detection module and show the attack mitigation strategies of the proposed algorithm, namely the disabling of the compromised u_i channel and the platoon rearrangement.

To select appropriate values of K and $\bar{r}$ for the attack detection module we used data from *Experiment 2* and simulated the residual dynamics according to different K values and chose corresponding appropriate $\bar{r}$ threshold values. For the current experiment we chose $K = 0.05$ and $\bar{r} = 0.75$. Note that properly tuning the low-level longitudinal controller in equation (3.23) in the appendix will strongly affect the ability of the vehicles

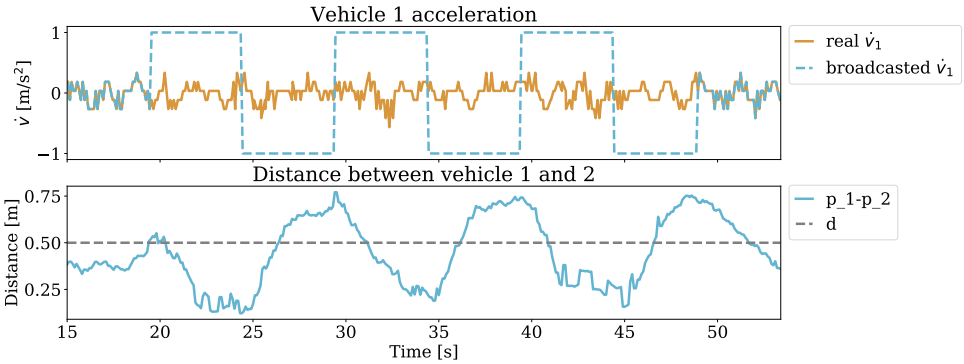


Figure 3.10: Experiment 2: We simulate a FDI attack between the leader and vehicle 2. The collision avoidance properties of the proposed algorithm prevent vehicle 2 from crashing into the leader. Without any attack detection module however, the attacker is still able to manipulate the distance between the vehicles.

to accurately track the $u_i(t)$ and $u_{i+1}(t)$ profiles outputted by the platooning controller. Poor tracking performance will thus hinder the attack detection module, as the effect of a malicious attack will be "hidden" by the baseline process noise. To further validate our tuning process we also recorded data during nominal platooning conditions and simulated different kinds of attacks, namely by replacing the recorded u_1 with a sinusoidal signal, and by adding a Gaussian noise, both with different sets of parameters such as frequency and intensity. The healthy system data and the attack simulation code is available on the GitHub repository [74].

We now demonstrate the attack mitigation properties of the proposed algorithm by enabling the attack detector module and subjecting the platoon to the same FDI attack as in *Experiment 2*. As shown in Figure 3.11, the attack is detected within few seconds, note that for this experiment we additionally require that $r > \bar{r}$ for 0.5s consecutively to consider the system under attack (and set $\sigma = 0$) as measure to reduce false positives. Since while the attack is detected u^{ff} is disabled, the attack no longer has the effects visible in *Experiment 2*. Note that we disable the feed-forward action only on vehicle 2, the rest of the platoon maintains nominal functionality. The Platoon Coordinator onboard each vehicle then changes the platoon topology and the compromised vehicle is assigned to the end of the platoon. To perform this maneuver vehicles $2, 3, 4$ switch to an overtaking lane and then switch back to the original platooning lane as shown in the video of the experiments [75]. To execute the platoon rearranging in practice we have designed a fully distributed logic that runs onboard each vehicle. The outputs of this logic are mainly the lane the vehicle should be in, i.e. the slow lane or the overtaking lane, and the velocity reference v_D. The latter is needed to speed up the vehicle once it is in the overtaking lane, or to slow it down when waiting to be overtaken by other vehicles. Details of the platoon rearranging logic are discussed in appendix 3.7.

As shown in Figure 3.11, after the new topology has been reached, since the compromised vehicle has no follower to communicate with, the attack has been isolated and the platoon recovers complete nominal functionality, i.e. all the residuals are below the threshold. Notice that vehicle 2 is the new platoon leader and thus has no inbound communication,

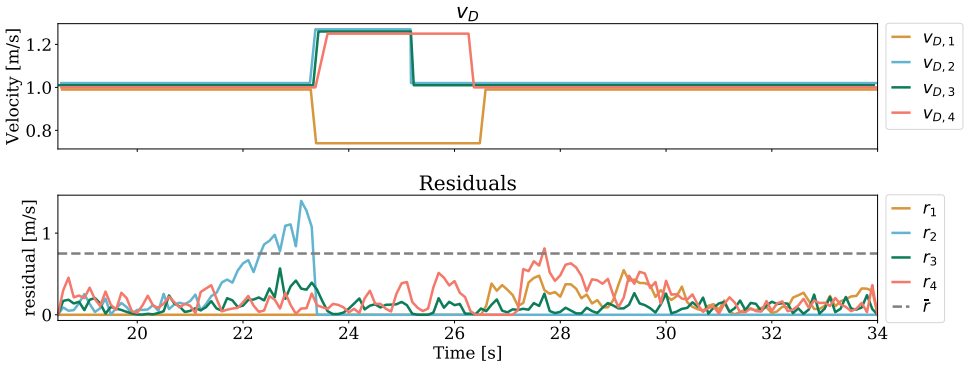


Figure 3.11: Experiment 3: We show the attack mitigation properties of the proposed platooning strategy by activating the attack detection module and subjecting the system to the same fake data injection attack as in Experiment 2. The attack is detected within a few seconds , as can be seen from the sudden drop in r_2 after it exceeds the threshold, signaling that the communication-based u_2^{ff} has been disabled. This stops the attack from taking control of vehicle 2 as in Experiment 2. The platoon coordinator module then changes the topology and assigns the compromised vehicle to the end of the platoon. The *platoon rearranging logic*, detailed in appendix 3.7, then executes the overtaking maneuver. Vehicles 2, 3, 4 switch to the overtaking lane, while vehicle 1 slows down to facilitate the overtaking. Once all vehicles have merged back into the slow lane and the new topology has been achieved, the compromised vehicle has no follower to communicate with. The attack has thus been isolated and the platoon completely recovers nominal functionality. Notice that the platoon leader (first vehicle 1 and then vehicle 2 after the topology change) has no inbound communication to assess, to indicate this we set $r = 0$. For visual clarity we also offset the v_D by a small amount.

to indicate this we set $r_2 = 0$.

3.6 Conclusions

We present our distributed attack-resilient platooning strategy. We combine a safe and string stable platoon controller with a platoon coordinator, which is able to reorganize the platoon. Our proposed method preserves string stability and safety guarantees in case of a false acceleration data attack, as supported by the simulation and experimental results. Future works may include considering other communication topologies between vehicles and other attacker models, working on more sophisticated trustworthiness estimation algorithms, and testing our algorithm with an increased number of robots.

3.7 Appendix

Experimental set-up. For the experiments we used DART, the robotic platform presented in our previous work [54]. The robots are equipped with a wheel encoder to measure their own speed and a Lidar to perceive the surrounding environment. The team of robots follows a predefined path while the platooning controller regulates the longitudinal distances. Each robot runs the perception pipeline and control algorithms on-board, i.e. the platooning strategy runs in a fully distributed fashion. Communication among robots is handled by a dedicated ROS network. The code used during the experiments can be found in the publicly available GitHub repository [74].

The proposed platooning algorithm is designed considering the vehicles' longitudinal

dynamics as a linear system that can be controlled with acceleration inputs. To run the proposed platooning algorithm on real robots we must first design three additional modules, namely: a lateral controller to follow the predefined path, a perception pipeline to measure the relative distance and velocity from the preceding vehicle, and a low level controller that enforces the longitudinal acceleration commands coming from the platooning controller. Figure 3.12 shows an overview of the processes running on board each robot. We will now address each component.

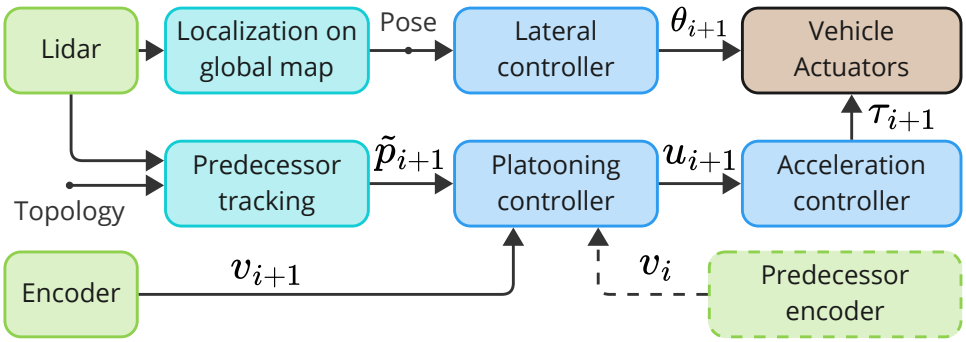


Figure 3.12: Overview of the processes running on-board the $(i+1)$-th vehicle.

Lateral controller. To replicate a highway scenario the predefined path features straight sections and gentle curves. Under these circumstances the lateral and longitudinal vehicle dynamics can be decoupled, allowing us to design the lateral controller separately from the platooning controller. To perform path tracking we rely on the standard ROS libraries to estimate each robot's pose and position on a previously built map. The steering angle θ_{i+1} is then evaluated using a pursuit controller [86]. Due to delays in the Lidar-based perception pipeline, implementing a leader-follower tracking lateral controller resulted in significant lateral string stability issues, for this reason we designed the lateral controller on each vehicle to track the lane centerline independently of its predecessor, circumventing any possible lateral string stability issues. Furthermore this allowed us to replicate a two lane highway scenario and the relative platoon rearranging algorithm presented in Figures 3.13 and 3.14.

Perception pipeline. The longitudinal controller requires three inputs, namely the $(i+1)$-th vehicle velocity v_{i+1}, and the relative velocity and position of the preceding vehicle, $\tilde{v}_{i+1}$ and $\tilde{p}_{i+1}$ respectively. v_{i+1} is readily available since each vehicle is equipped with a wheel speed encoder. To evaluate $\tilde{v}_{i+1}$ a full scale vehicle would typically rely on sensors such as frequency-modulated RADAR and Depth cameras, yet due to the limited sensing capabilities of the platform we use communication, i.e. we evaluate $\tilde{v}_{i+1}$ as $v_{i+1} - v_i$ where v_i is communicated by the i-th vehicle. The relative distance $\tilde{p}_{i+1}$ is measured with the on-board Lidar. Each vehicle processes the incoming 2D Lidar scan and extracts clusters of points that represent different objects. The robots choose the vehicle to follow (VTF) as detailed in the *Platoon rearranging logic* at the end of this section. The distance to the vehicle to follow is then measured as the distance to the centre of the cluster of points closest to the VTF's advertised position on the global map. This can be seen in the experiments

video [75].

Low-level acceleration tracking controller. The platooning controller receives v_{i+1}, $\tilde{v}_{i+1}$, $\tilde{p}_{i+1}$ and produces a control action u_{i+1} in $\frac{m}{s^2}$, yet the vehicle's actuators only accept throttle values $\tau \in [0, 1]$. We thus need a low-level controller that tracks u_{i+1}. We designed the low-level longitudinal controller as:

$$\tau_{i+1} = \tau^{\text{ff}}_{i+1}(u_{i+1}, v_{i+1}) + \tau^{\text{fb}}_{i+1}. \tag{3.23}$$

Where $\tau^{\text{ff}}_{i+1}(u_{i+1}, v_{i+1})$ is a feed-forward term obtained by solving the inverse system dynamics, i.e. by inverting the dynamic model $u = f(\tau, v)$. To obtain the dynamic model of the vehicles f we relied on the system identification procedure presented in [54]. Due to hardware low quality, time-dependent ware of the compomenents and lack of battery state of charge estimation, the robotic platforms' show a significant variation in their behavior, both among different vehicles and over time. To address this issue we introduce the feedback term τ^{fb}_{i+1}. The latter is a corrective integral action that acts as an adaptive term, effectively setting a different initial throttle offset for each vehicle. Note the design and tuning of the low level controller strongly affects the overall performance of the platooning controller, since it will affect the resulting acceleration profile. It may even affect the string stability property if the requested u_{i+1} is tracked with some time delay. This is why we advise against using a Proportional-Derivative structure in the low level controller and suggest using small gains for the integral action τ^{fb}_{i+1}.

Platoon rearranging logic. In *Experiment 3* once an attack is detected and one of the communication channels is closed, the platoon coordinator can isolate the compromised vehicle by moving it to the end of the platoon. To perform this rearranging maneuver in practice, we have designed a fully distributed logic that runs onboard each vehicle. The outputs of this logic are which lane the vehicle should follow, its velocity reference v_D and what vehicle it should follow. The latter is mainly needed to avoid abrupt changes in velocity since when a new desired topology is outputted by the platoon coordinator, the i-th vehicle's assigned predecessor may not be in sight yet. The inputs to the logic are: *1)* the desired topology coming from the platoon coordinator (that also runs on each vehicle in a distributed fashion) that indicates the assigned predecessor (AP), *2)* the observed predecessor (OP), that is the closest vehicle in front of the ego vehicle regardless of the lane it is in, *3)* the lane (L) the ego vehicle is currently in, L=Slow Lane (L=SL) or L=Fast Lane (L=FL), *4)* the lane its assigned predecessor is currently in (AP L), *5)* if its assigned predecessor is signaling with the right indicator, i.e. if it intends to merge into the slow lane (MSL). When a vehicle is in the slow lane (L=SL) and it's observed predecessor is its assigned predecessor, i.e. L=SL and AP=OP, the ego vehicle has reached the correct platooning position (CPP). Notice that AP=0 indicates the ego vehicle should be the leader of the platoon. On a real full-scale vehicle all the required inputs can be gathered using cameras, since all the data concerns the immediate surroundings of the ego vehicle. For the experiments we rely on the advertised position of each vehicle on the global map and on communication, for example to detect if MSL is active or not.

We define three separate logic controllers, each regulating the lane choice, v_D and vehicle to follow respectively. *Lane manager.* To determine in which lane the vehicle should be in, we have identified 10 different scenarios the vehicles may encounter and programmed the required actions. We first differentiate if the ego vehicle is the new leader

or not, if it is (AP = 0) it will encounter scenarios 1-3, shown in figure 3.13. If not (AP ≠ 0) it will encounter scenarios 4-10 shown in figure 3.14. When changing lanes, merging to the fast lane is always allowed because in these experiments we don't consider other traffic participants that may be overtaking. When merging to the slow lane instead we must check that there is enough space between the ego vehicle and the following vehicle. In this case MSL=True but the vehicle is still in the fast lane L=FL. The information about MSL is also necessary to disambiguate scenario 5 from scenario 7 that would otherwise be identical.

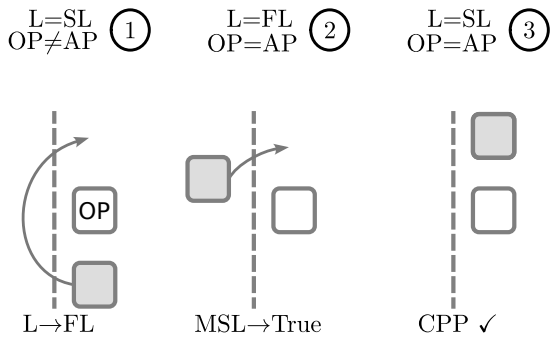


Figure 3.13: Leader lane management logic. The ego vehicle is represented as a shaded rectangle.

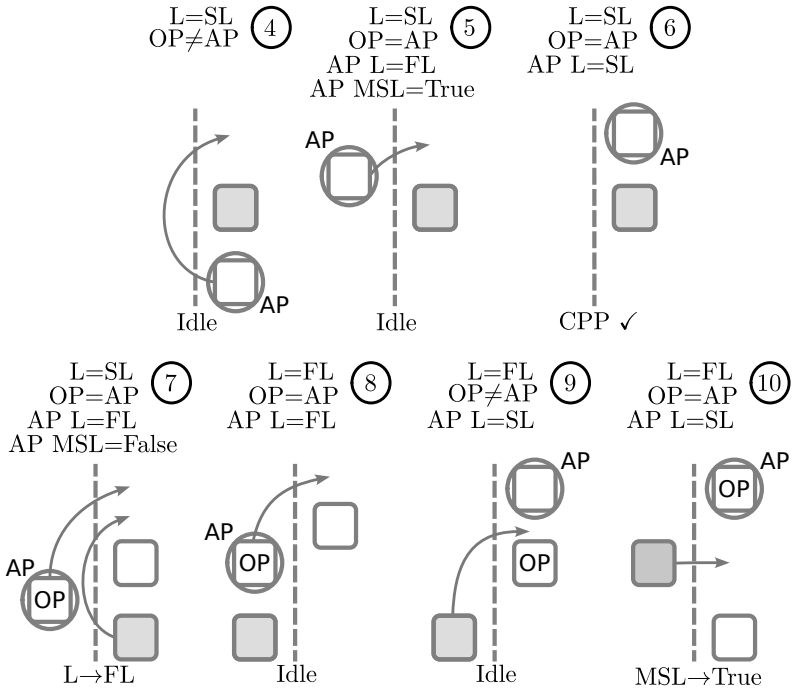


Figure 3.14: Follower lane management logic. The ego vehicle is represented as a shaded rectangle.

Velocity manager. We apply a simple logic to regulate the speed of the ego vehicle in order to facilitate overtaking, shown in algorithm 1.

Algorithm 1 Decision logic for v_D

```
if L = SL then
    if AP = OP then
        v_D ← default
    else
        v_D ← slow
    end if
else
    if APL = SL then
        v_D ← default
    else
        v_D ← fast
    end if
end if
```

Vehicle to follow manager. Vehicle to follow (VTF) refers to which vehicle the ego vehicle tries to keep the desired distance d from. To avoid abrupt changes the logic shown in algorithm 2 enables to switch only if the assigned predecessor is in front of the ego vehicle.

Algorithm 2 Decision logic for vehicle to follow

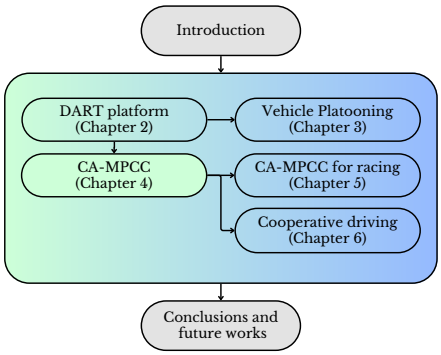
```
if AP = OP then
    VTF ← AP
else
    if L=SL then
        VTF ← 0
    else
        VTF ← closest preceding vehicle in FL
    end if
end if
```

4

Curvature-Aware Model Predictive Contouring Control

Overview

This Chapter introduces a *Curvature-Aware* Model Predictive Contouring Control[1] (CA-MPCC) formulation for mobile robot motion planning, extending the traditional Model Predictive Contouring Control (MPCC). Standard MPCC has been successfully transferred from the field of machining, where it was first invented, to autonomous driving applications, but it relies on the assumption of small lateral deviations from the reference path. This assumption, valid in high-precision machining in industrial contexts, often breaks down in real-world driving scenarios where sharp turns, actuator limits, and dynamic obstacles force larger deviations. As a result, conventional MPCC may mis-evaluate lane boundary constraints or degrade performance due to inaccuracies in path progress prediction.



To address these limitations, the proposed CA-MPCC explicitly incorporates local path curvature into the definition of the path progress variable. A novel approximation of the path derivative is derived, which corrects for curvature effects and remains accurate even when deviations from the reference path are significant. This reformulation leads to

[1]This paper is based on *Curvature-aware model predictive contouring control* by Lorenzo Lyons and Laura Ferranti [87]. L. Lyons' contributions consist in developing the methodology, implementing the algorithm in simulation and on the robotic platform, performing the experiments and writing the manuscript. Laura Ferranti supervised the research.

two key benefits: (i) improved robustness in maintaining lane constraints and avoiding obstacles, and (ii) simplified controller tuning, as one cost term (lag error) can be eliminated thanks to the richer information encoded in the curvature-aware formulation.

The method is validated both in simulation and on DART, the 1:20 scaled car-like robotic platform presented in chapter 2. In simulation, extensive parameter sweeps demonstrate that CA-MPCC achieves consistently higher success rates in overtaking dynamic obstacles on tight turns, with less sensitivity to tuning parameters than baseline MPCC. In experiments, CA-MPCC shows more reliable closed-loop behavior, tracking path progress accurately even under sharp turns, while maintaining real-time feasibility despite slightly higher computational demands.

Overall, the results confirm that CA-MPCC is a robust and practical alternative to traditional MPCC for autonomous driving and robotics. It enhances reliability in constrained, curved environments, reduces the burden of controller tuning.

4.1 Introduction

Automated vehicles, such as self-driving cars, have the potential to revolutionize our urban mobility. They can help us regulate traffic in big cities, reduce fatalities caused by human errors, and make transportation more accessible to elderly population or people with disabilities [88]. While we can gain a lot out of these technologies, the majority of commercial automated vehicles are still far from reaching the highest level of automation (i.e., SAE level 5) [89]. Safety is still a major concern for further developing these vehicles, especially in complex urban environments or in rural areas. While on the highway the vehicle is rarely required to perform harsh turns and avoid obstacles on narrow lanes, these situations often occur in our cities or in back-country roads. To navigate an autonomous vehicle in these challenging environments, the local motion planner plays a fundamental role since it acts as a bridge between how the vehicle perceives the environment and how it reacts to it. The motion planner has the task to keep the vehicle on the road, providing the right references to the actuators to stay within the lane boundaries, while avoiding collisions with obstacles. Precisely accounting for the road curvature is one of the challenges that the motion planner must face to keep the vehicle on track. This paper proposes a *curvature-aware* local motion planner based on nonlinear model predictive control (NMPC) to address these challenges.

Traditional approaches to path following include feedback controllers [90] or iterative linear quadratic regulator (iLQR) [91]. These approaches, however, consider decoupling lateral and longitudinal dynamics and/or do not account for state and actuator constraints. MPC is a valid alternative to these methods. MPC is an optimization-based control technique that allows one to formulate desired control objectives and optimize the behavior of the vehicle over a predefined time window (refer to [92] for an overview). A useful comparison among PID, iLQR and MPC can be found in [93].

In this paper, we build on a NMPC formulation suitable for path planning, known as model predictive contouring control (MPCC). Compared to reference-tracking MPC [94], MPCC includes an analytical description of the reference path, which is not parameterized with respect to time but with respect to a tailored path parameter. MPCC was originally applied in machining[31, 95, 96]. In this field path following had been addressed prior to these works, yet the focus was on providing geometrical convergence to the path by means

of feedback controllers' (e.g.,[97]). Understanding the origin of MPCC is relevant to the present work, since the formulation developed by [95] for machining has been transferred almost unaltered to the field of autonomous vehicles, for example in [98–100]. These approaches showed the potential of MPCC for robot navigation in dynamic environments. Classical MPCC, however, relies on the assumption of small lateral deviations of the vehicle from the reference path. This is surely the case in a high-precision application where the quality specifications often require lateral errors in the order of microns. In autonomous driving (and mobile robotics in general) this assumption may not hold since the presence of obstacles or actuator saturation may force the vehicle to deviate significantly from the reference path. Efforts to handle this issue can be found in the field of autonomous racing, where race tracks usually feature sharp turns. In [32] an effective strategy to account for high curvatures is presented, yet it is used offline to define an optimal path. The results are validated in simulation and path tracking is enforced by a low-level controller integrated in the simulation software. In [101] and [102] a similar approach is taken and the findings are validated on a full scale autonomous racing car. Exploiting the prior knowledge of the full track is a successful strategy in a racing context, yet this detailed knowledge is generally not available in an urban driving scenario. Handling tight curves relying only on local curvature information is the challenge addressed in the present work.

The main contribution of the present work is to develop a MPCC formulation for mobile robot navigation capable of accounting for sharp turns in, or deviating from, the reference path without any off-line computations. Thanks to our *curvature-aware* reformulation we show how the proposed MPCC method is more reliable (e.g., to keep the vehicle within road boundaries and avoid obstacles) and easier to tune than the traditional MPCC derived from machining. We support these claims both in simulation and with experiments on a real robotic platform.

The paper is structured as follows. Section 4.2 presents the problem formulation and details the limitations of traditional MPCC. Section 4.3 presents our method. Section 4.4 compared the MPCC baseline with our method, both in simulation and in experiments. Finally, Section 4.5 concludes the paper.

4.2 Problem Formulation

We consider a robot described by the following discretized dynamic equations:

$$\boldsymbol{x}(t+1) = f(\boldsymbol{x}(t), \boldsymbol{u}(t)), \tag{4.1}$$

where $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{u} \in \mathbb{R}^m$ are the state and control input of the vehicle, respectively. The state and the control input are subjected to limitations, that is, $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^n$ (e.g., lane boundaries) and $\boldsymbol{u} \in \mathcal{U} \subseteq \mathbb{R}^m$ (e.g., steering and acceleration limits). The vehicle moves in a complex dynamic environment. In this respect, we consider dynamic obstacles moving with a constant velocity[2]. We represent the obstacles as circles with radius r_{obst} and with state $\boldsymbol{x}^{\text{o}}$. The set of neighboring obstacles is $\mathcal{O}$.

Our goal is to design a local motion planner for the ego vehicle to safely navigate in a tight dynamic environment efficiently, penalizing large deviations from the reference path

[2]Our method can be interfaced with any perception algorithm able to provide predictions of the behavior of the obstacles. Predicting obstacles' behavior is outside the scope of this work.

$\boldsymbol{x}^{\text{ref}}$ provided by a high-level motion planner, in our work we define the latter as the lane centerline. This can be formalized as follows:

$$\min_{\boldsymbol{x}_k, \boldsymbol{u}_k} \sum_{k=0}^{N} J_k(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{x}^{\text{ref}}) \tag{4.2a}$$

$$\text{s.t.: Eq. (4.1)}, \tag{4.2b}$$

$$\boldsymbol{x} \in \mathcal{X},\ \boldsymbol{u} \in \mathcal{U}, \tag{4.2c}$$

$$\|\boldsymbol{c}_k^i - \boldsymbol{x}_k^o\|_2 \geq r + r_{\text{obst}}, i = 1, ..., n_{\text{disc}}, o \in \mathcal{O}, \tag{4.2d}$$

$$\boldsymbol{x}_0 = \boldsymbol{x}_{\text{init}}, \tag{4.2e}$$

where J indicates the planner objectives we aim to define and Constraint (4.2d) ensures collision avoidance between the neighboring obstacles and the center of each disc $\boldsymbol{c}^i$ of radius r used to represent the vehicle (following a similar strategy to [98]). Finally, $\boldsymbol{x}_{\text{init}}$ indicates the measurements the planner receives from the sensors at every sampling instant.

The problem above can be solved with MPCC, which is a well-established technique for motion planning. The remainder of the section provides a summary of the traditional MPCC formulation (Section 4.2.1) and discusses its limitations (Section 4.2.2).

4.2.1 Overview of MPCC

MPCC recursively solves Problem (4.2): in each control loop a fresh current vehicle state measurement is provided, updating constraint (5.9e) accordingly. A numerical solver is then used to provide the optimal solution to this instance of the problem, that is, to provide the optimal time sequence of future control inputs $\boldsymbol{u}_k$ that minimize the cost function over the fixed time horizon. The MPCC controller then selects the first control input $\boldsymbol{u}_0$, that refers to the current time, and applies it to the system. This process is then repeated at every control loop in a receding-horizon fashion. We refer to the solution of a single instance of problem (4.2) as the open-loop solution, while we define closed-loop behavior the resulting behavior of the system.
The cost J defined as follows:

$$J_k = q_1(v_k - v_t)^2 + q_2(\varepsilon_k^c)^2 + q_3(\varepsilon_k^{\text{lag}})^2 + \boldsymbol{u}_k^T Q_u \boldsymbol{u}_k \tag{4.3}$$

The cost is given by the sum of *(i)* three errors, that are, the deviation of v (i.e., the euclidean norm of the velocity of the mobile robot) w.r.t. the desired velocity v_t, the contour error ε^c and lag error ε^{lag} (defined below); and *(ii)* a quadratic penalty on the control action $\boldsymbol{u}$. The scalars q_1, q_2 and q_3, and the matrix Q_u are tuning parameters of the controller. The use of the lag and contour errors in the cost of the controller is what differentiates the MPCC controller from a standard NMPC for trajectory tracking. These errors are defined as follows:

$$\varepsilon^c = (\boldsymbol{p} - \boldsymbol{\phi}(s)) \cdot \hat{\boldsymbol{n}}(s),\ \varepsilon^{\text{lag}} = (\boldsymbol{p} - \boldsymbol{\phi}(s)) \cdot \hat{\boldsymbol{t}}(s),$$

where $\boldsymbol{p}$ is the position of the robot (i.e., $\boldsymbol{p} = [x\ y]^{\text{T}}$), $\boldsymbol{\phi}(s)$ is the reference path parameterized on the arc length s, $\hat{\boldsymbol{t}}(s)$ and $\hat{\boldsymbol{n}}(s)$ are respectively the tangent and normal unit vectors to the path at a given value of s, that is, they constitute the Frenet frame of reference [103]. We can thus interpret ε^c and ε^{lag} as attractive terms that pull $\boldsymbol{p}$ towards $\boldsymbol{\phi}(s)$.

The value of s is defined as $s = \text{argmin}_s \|\boldsymbol{p} - \boldsymbol{\phi}(s)\|_2$. However, solving this minimization problem as a sub-problem of (4.2) results in a non-differentiable cost function, which is an issue for gradient based methods. For this reason it is necessary to provide an expression of the dynamics of s, which are traditionally approximated as $\dot{s} \approx v$ and added to the dynamic Constraints (4.1).

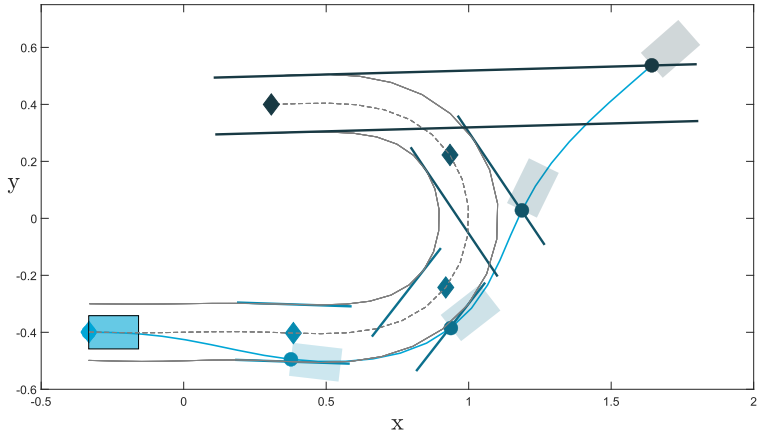
4.2.2 Limitations of MPCC

As stated in the introduction, the standard MPCC formulation was developed in the field of machining, where acceptable lateral deviation from the path are small, thus the *traditional* approximation $\dot{s} \approx v$ is reasonable. However, in the context of mobile robot navigation this approximation is not as easily verified, since large lateral deviations from the path may occur. When the assumption of $\dot{s} \approx v$ fails, two undesired behaviours may arise depending on the values of q_2 and q_3. The first undesired behaviour is that since the predicted value of s is no longer accurate, any s-dependent constraints will be inaccurately evaluated. A relevant example in the context of autonomous driving are lane boundaries, that are often defined as bounds on ε^c [98, 104], namely $\varepsilon^c_{\text{min}} \leq \varepsilon^c(s) \leq \varepsilon^c_{\text{max}}$. The erroneous evaluation of the lane boundaries may lead the open-loop solution to violate these constraints without the solver returning any infeasibility errors. This issue is typical for low q_2 and q_3 gain values. In Figure 4.1(a) we show the open-loop solution obtained for a scenario in which a vehicle enters a tight curve. Note that the lane boundaries are expressed with respect to the vehicles' rear axle position (i.e., it is this point that should remain within the bounds). It is clear how the inaccurate s prediction induces the controller to violate the real lane constraints, yet it is very important to note that the solver converges to a feasible solution since according to it the lateral error remains within the bounds. This type of *failure* is particularly dangerous since it is not detectable by looking at the solver outcome.

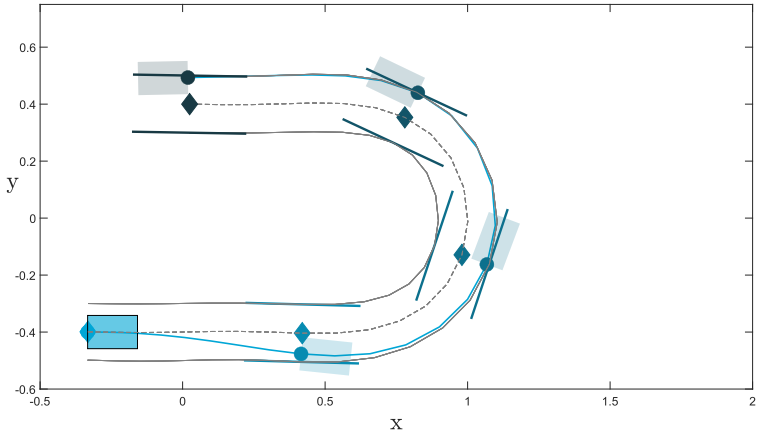
The second undesired effect is connected with the fact that $\boldsymbol{p}$ and s are closely related, as the following definition shows:

$$\boldsymbol{p}_k = \sum_{i=0}^{k} \boldsymbol{v}_i dt, \; s_k = \sum_{i=0}^{k} \|\boldsymbol{v}_i\|_2 dt$$

When the vehicle is unable to closely follow the path (e.g., due to an obstacle), the optimal solution to Problem (4.2) may be to reduce the value of $\boldsymbol{v}$ in order to reduce $\boldsymbol{p} - \boldsymbol{\phi}(s)$, and in turn ε^c and ε^{lag}. This effect is more evident for higher q_2 and q_3 gains. Figure 4.2(a) shows the open-loop solution in a scenario in which the vehicle enters a turn with a certain amount of initial lateral error. Steering limitations force the vehicle to keep a certain distance from the path, as a result we can see how $\boldsymbol{\phi}(s)$ is wrongly evaluated. Figure 4.2(b) shows the open-loop velocity profile. Note that the vehicle starts from a stationary condition. We can see how in the first stages the algorithm fails to track the velocity, only to recover once the vehicle exits the curve. Notice that both limitations can lead to dangerous driving scenarios. In addition, the occurrence of these problems is highly dependent on the tuning of the controller. This makes the tuning of the controller extremely challenging and scenario dependent. Our method aims to solve these limititations, as discussed in the next section.



(a) MPCC



(b) CA-MPCC

Figure 4.1: Comparison of the open-loop trajectories. The dots are the vehicle rear axle positions for solver stages 1, 12, 18, 23, 30 ordered from lighter to darker shades of blue, the diamonds are the corresponding $\boldsymbol{\phi}(s_k)$ as evaluated by the solver and the solid lines are the corresponding local lane boundaries.

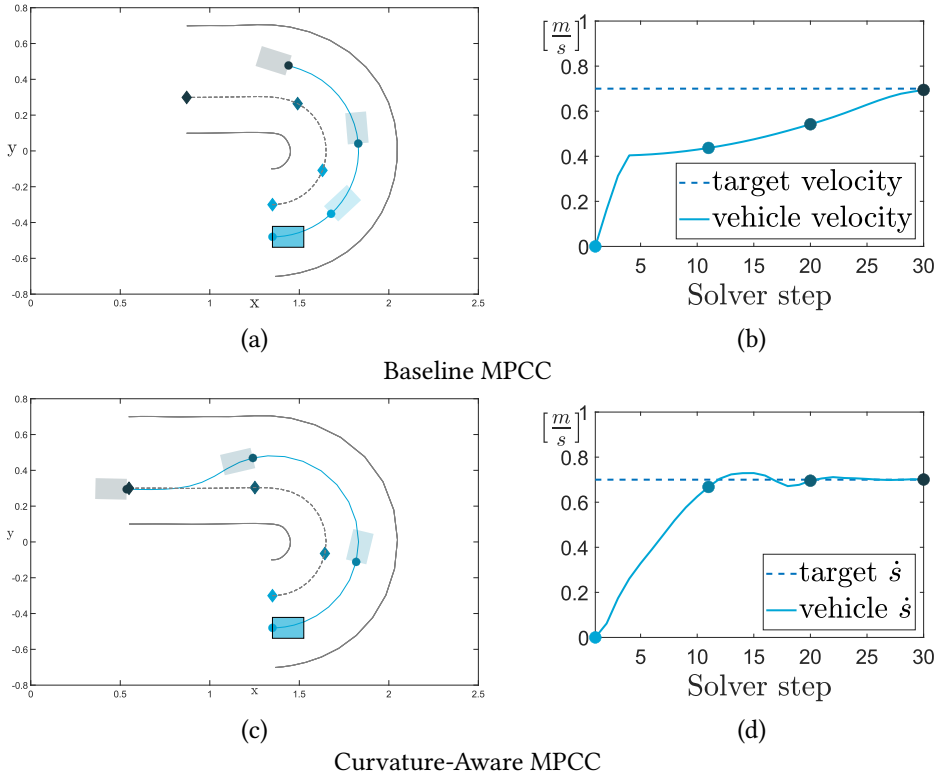


Figure 4.2: Plots (a) and (c) show the open-loop solutions. Plots (b) and (d) show the velocity and $\dot{s}$ profile, respectively. The dots match the predictions plotted in (a) and (c).

4.3 Curvature-Aware MPCC

Accurately evaluating $\dot{s}$ is key to overcome the issues listed in the Sec. 4.2.2. The analytical expression of $\dot{s}$ is:

$$\dot{s} = \frac{\boldsymbol{v} \cdot \hat{\boldsymbol{t}}(s)}{1 - \kappa(s)\varepsilon^c}, \tag{4.4}$$

where $\kappa(s)$ is the curvature of the path. As described in [32] this expression can be obtained by re-writing $\boldsymbol{p}$ in the $\hat{\boldsymbol{t}}$ and $\hat{\boldsymbol{n}}$ frame and taking the first time derivative. Using Eq. (4.4) directly in Problem (4.2), however, comes with a significant increase in computation time required for online optimization. This is because $\hat{\boldsymbol{t}}$ and $\kappa(s)$ have to be evaluated starting from an analytical expression of $\boldsymbol{\phi}(s)$ that is generally computationally expensive. For this reason we provide a lightweight approximation of $\dot{s}$. According to Eq. (4.4) $\dot{s}$ can be interpreted as the projection of $\boldsymbol{v}$ on $\hat{\boldsymbol{t}}$ times a curvature-dependent scaling ratio. We can define the scaling ratio σ as:

$$\sigma = \frac{1}{1 - \kappa(s)\varepsilon^c} = \frac{R(s)}{R(s) - \varepsilon^c},$$

where we have rewritten κ as $\frac{1}{R}$, R is the curvature radius. Let $\boldsymbol{C}$ be the centre of curvature of the path. We consider an infinitesimal change in position $d\boldsymbol{p} = \boldsymbol{v}dt$ and we denote the corresponding infinitesimal change in the path parameter as derived from Eq. (4.4) as ds^*. Let the point $\tilde{\boldsymbol{p}}_t$ and $\tilde{\boldsymbol{s}}$ be defined respectively as:

$$\tilde{\boldsymbol{p}}_t = \boldsymbol{p} + \boldsymbol{v}dt \cdot \hat{\boldsymbol{t}},$$
$$\tilde{\boldsymbol{s}} = \boldsymbol{\phi}(s) + \hat{\boldsymbol{t}}ds^*.$$

We can now give a geometrical interpretation to Eq. (4.4) by noticing that the length of the segments $\overline{\boldsymbol{C}\boldsymbol{\phi}(s)}$ and $\overline{\boldsymbol{C}\boldsymbol{p}}$ are respectively R and $R - \varepsilon^c$. Since the triangles ${}_{\Delta}\boldsymbol{C}\boldsymbol{p}\tilde{\boldsymbol{p}}_t$ and ${}_{\Delta}\boldsymbol{C}\boldsymbol{\phi}(s)\tilde{\boldsymbol{s}}$ are similar we can write:

$$\frac{ds^*}{v_t dt} = \frac{\|\tilde{\boldsymbol{s}} - \boldsymbol{\phi}(s)\|_2}{\|\tilde{\boldsymbol{p}} - \boldsymbol{p}_t\|_2} = \frac{R}{R - \varepsilon^c} = \sigma,$$

where for ease of notation purposes we have defined $v_t = \boldsymbol{v} \cdot \hat{\boldsymbol{t}}$ and dropped the $R(s)$ dependency. From this expression we can see how σ can be interpreted as a projection ratio, see Figure 4.3. This geometrical interpretation highlights how even the analytical expression of $\dot{s}$ becomes inaccurate for large Δt values and high path curvature. We thus propose a novel $\dot{s}$ expression that is exact under the assumptions of constant curvature radius and robot velocity, and is therefore a suitable approximation for large Δt values when curvature and robot velocity vary smoothly. To simplify notation (without loss of generality) we assume that $t_0 = 0$. In this way we can rewrite Δt as t. We also define the point $\tilde{\boldsymbol{p}} = \boldsymbol{p} + \boldsymbol{v}t$. The proposed method relies on first evaluating the angle θ between $\boldsymbol{p}$, $\boldsymbol{C}$ and $\tilde{\boldsymbol{p}}$ and then evaluating ds as $ds = R\theta$. Defining $v_n = \boldsymbol{v} \cdot \hat{\boldsymbol{n}}$, as shown in Figure 4.3, the angle θ can be calculated as:

$$\theta = \arctan\left(\frac{v_t t}{R - \varepsilon^c - v_n t}\right) \tag{4.5}$$

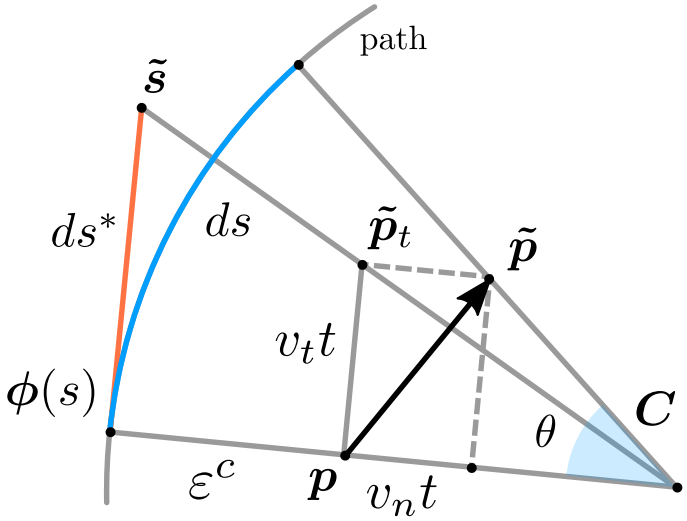


Figure 4.3: Geometrical interpretation of the traditional (orange) and proposed (blue) $\dot{s}$ formulation.

To obtain $\dot{s}$ we simply take the derivative of $ds = R\theta$ with respect to t, that is:

$$\dot{s}(t) = R\frac{v_t(R - \varepsilon^c - v_n t) + v_t v_n t}{(R - \varepsilon^c - v_n t)^2 + (v_t t)^2} \tag{4.6}$$

Notice that for $t \to 0$ Eqs. (4.4) and (4.6) coincide. It is also important to highlight that since v_t and v_n are fixed, (4.6) is the time derivative of s for a point moving in a constant direction. Expressions (4.4) and (4.6) have similar computational complexity since they both require the evaluation of $\hat{\mathbf{n}}(\mathbf{s})$ and $\hat{\mathbf{t}}(\mathbf{s})$ to obtain R, v_t and v_n. However we propose to use Eq. (4.6) in its integral form, that is, $ds = R\theta$, for the whole integration step. After evaluating $d\boldsymbol{p}$ by means of any integration scheme, we evaluate $v_t t$ and $v_n t$ as $d\boldsymbol{p} \cdot \hat{\boldsymbol{t}}$ and $d\boldsymbol{p} \cdot \hat{\boldsymbol{n}}$, respectively, and use them in Eq. (4.5) to evaluate $ds = R\theta$. Notice that by construction (4.6) has the property $\int_0^t \dot{s} dt = R\theta$. This approach has similar computational cost as using (4.4) in a Forward Euler method, yet it achieves much higher accuracy for small R and large t values, as can be seen in Figure 4.3. Another advantage of this formulation is that the integration scheme used to evaluate the system dynamics ($d\boldsymbol{p}$) is decoupled from the one used to evaluate ds, since the latter is evaluated after the former. This also implies that any dynamic model can be chosen, as long as it provides the change in position of a chosen reference point on the vehicle.

The new definition of $\dot{s}$ in (4.6) can be used to derive the following Curvature-Aware MPCC formulation, namely a CA-MPCC, suitable for generic motion planning applications:

$$\min_{\boldsymbol{u}_k} \sum_{k=0}^{N} q_1(\dot{s}_k - \dot{s}_t)^2 + q_2(\varepsilon_k^c)^2 + \boldsymbol{u}_k^T Q_u \boldsymbol{u}_k \tag{4.7a}$$

$$\text{s.t.} \quad \begin{bmatrix} \boldsymbol{x}_k \\ s_k \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}) \\ s_{k-1} + R_{k-1}\theta_{k-1} \end{bmatrix} \tag{4.7b}$$

$$\boldsymbol{x} \in \mathcal{X}, \ \boldsymbol{u} \in \mathcal{U}, \tag{4.7c}$$

$$\|\boldsymbol{c}_k^i - \boldsymbol{x}_k^o\|_2 \geq r + r_{\text{obst}}, i = 1, ..., n_{\text{disc}}, o \in \mathcal{O}, \tag{4.7d}$$

$$\boldsymbol{x}_0 = \boldsymbol{x}_{\text{init}}, \tag{4.7e}$$

where R_{k-1} and θ_{k-1} are the respective quantities defined for time instant $k-1$. Equation 4.7b is also suitable for straight paths, since for $R \to \infty$, $R\theta = R \arctan \frac{v_t t}{R - \varepsilon^c - v_n t} \to v_t t$. Notice that the definition of the cost differs from Eq. (4.3). We replace the velocity and lag errors' penalties with an $\dot{s}$ tracking term. This is now possible since thanks to (4.6) we have access to this quantity. Notice that in the cost function, $\dot{s}_k$ is required. This is equivalent to taking the limit for $t \to 0$ in Eq. (4.6), that is equivalent to (4.4) as previously pointed out. It also worth mentioning that if the application requires a trade-off between v and $\dot{s}$ tracking, it is straightforward to add the velocity tracking term back into the cost function. We were also able to eliminate the lag error term since $\dot{s}$ also carries information on the heading direction of robot relative to the path. Having removed such an additional cost term simplifies the tuning of the CA-MPCC, as further discussed in Section 4.4.1.

Figure 4.1(b) shows the open-loop solution obtained with our CA-MPCC in the same conditions and same q_1, q_2, Q_u as for the baseline MPCC. Notice how, compared to the baseline MPCC, the lane boundaries are correctly evaluated thanks to a more precise s prediction. As a result the open-loop solution does not violate the lane boundary constraints. Figure 4.2(c) shows the solution to an initial standstill condition featuring some initial lateral error. Steering limitations do not allow the solver to reduce the initial lateral error, yet, as Figure 4.2(d) shows, the open-loop solution can accurately track the desired $\dot{s}$, leading to a safer driving experience.

4.4 Results

This section compares the closed-loop performance of the baseline- and CA-MPCC formulations.

Model. We tested the method on an autonomous driving application. The model used by both methods (i.e., in (4.2b) and (4.7b)) is given by the following equations:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\eta} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} v\cos\eta \\ v\sin\eta \\ \frac{v\tan(\delta)}{l} \\ -cv + \alpha\tau - \beta \end{bmatrix}$$

where the states x, y, η and v are respectively the position of the rear axle, orientation, and longitudinal velocity of the vehicle. The inputs τ and δ are throttle and steering, respectively; l is the length between the front and rear axle of the robot. The longitudinal

acceleration is defined as $\dot{v} = -cv + \alpha\tau - \beta$, where c is the damping coefficient, and α and β are motor coefficients. We evaluated their values experimentally through step-response tests a model of the platform, and on the real platform that will be used for the experiments (Section 4.4.2). Actuator limitations are defined as $\tau_{\min} \leq \tau_k \leq \tau_{\max}$ and $\delta_{\min} \leq \delta_k \leq \delta_{\max}$.

Local Path. Both methods require a strategy to evaluate $\boldsymbol{\phi}(s)$. For this, we decided to describe the local path with Chebyshev polynomials [105]. We split $\boldsymbol{\phi}(s)$ into its x and y components, and perform the respective polynomial fits. This fitting operation is repeated at each control loop.

We also add a terminal cost term $\boldsymbol{x}_N^T Q_P \boldsymbol{x}_N$ in (4.2a) and (4.7a), which approximates the cost-to go from $k = N$ to $k = +\infty$ and thus ensures that the closed-loop behaviour resembles the open-loop predictions [106].

In the remainder of the section, we test the baseline MPCC and our CA-MPCC both in simulation and with real-life experiments on an autonomous driving scenario.

4.4.1 Simulations

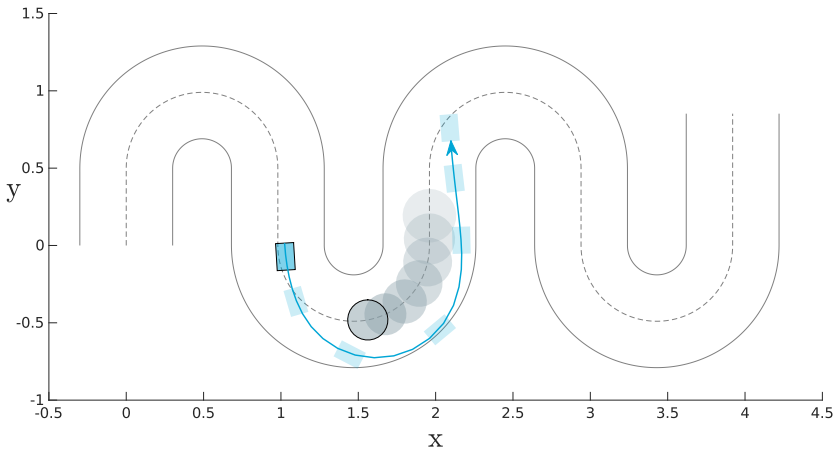


Figure 4.4: Track used for parameter sweep. The solid gray circle is the dynamic obstacle, while the faded gray circles are the dynamic obstacle's predicted positions.

To compare the reliability and the ease of tuning of the two approaches we tasked the algorithms with guiding an autonomous vehicle through a racetrack at a desired speed. The racetrack features some sharp turns that almost match the steering capabilities of the vehicle and some slow-moving dynamic obstacles that need to be overtaken. Figure 4.4 shows the chosen track. We have subsequently performed a parameter sweep on the weights. For both methods and for all tests, we kept $q_1 = 1$, and $q_\tau = 0.1$, $q_\delta = 0.1$, where q_τ and q_δ weight the respective inputs. Since $q_1 = 1$ the other weights are normalized with respect to the latter. Table 4.1 indicates the relevant parameters for the tests. R_{track} is the radius of the curves in the track, while $R_{\max}$ is the maximum curvature radius of the vehicle. The dynamic obstacle moves along the path at a speed $\dot{s}_{\text{obst}}$, and we simulate uncertainty in its occupancy predictions by increasing its radius at a rate $\dot{r}_{\text{obst}}$ along the prediction

Table 4.1: Overview of the tuning parameters used for the baseline MPCC and for the CA-MPCC.

Weights sweep parameters		
	MPCC	CA-MPCC
q_2	$q_2 = i,\ i \in \{0, ..., 11\}$	$q_2 = 0.2i,\ i \in \{0, ..., 25\}$
q_3	$q_3 = 0.5i,\ i \in \{0, ..., 11\}$	
Trials per data point	20	20
Other parameters		
Problem	Vehicle	Obstacle
$v_t = 0.75$ m/s (MPCC) $\dot{s}_t = 0.75$ m/s (CA-MPCC) $R_{\text{track}} = 0.5$ $\varepsilon^c_{\text{max,min}} = \pm 0.3$ m $Q_u = 0.1 I_{2\text{x}2}$	$\delta_{max} = 20^\circ$ $l = 0.175$ m $R_{\text{max}} = 0.48$ m	$r_{obst} \in [0.10, 0.15]$ m $\dot{r}_{obst} \in [0.01, 0.02]$ m $\dot{s}_{obst} \in [0.2, 0.3]$ m/s

horizon. Once a dynamic obstacle is overtaken, it is re-positioned ahead of the mobile robot, each time randomly drawing new obstacle parameters from the sets described in Table 4.1. A test is considered to be a success if no lane boundary violation occurs, and if the vehicle reaches the end of the track within a given time interval. This time interval is chosen such that if the average velocity of the robot was less than 60% of v_t or $\dot{s}_t$, the test results in a failure. This occurs if the vehicle is unable to overtake the dynamic obstacle.

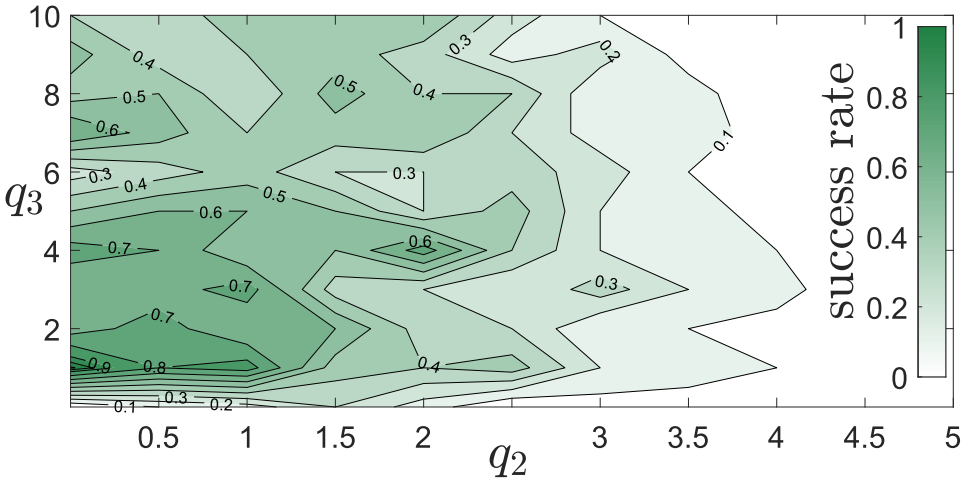


Figure 4.5: Baseline MPCC parameter sweep.

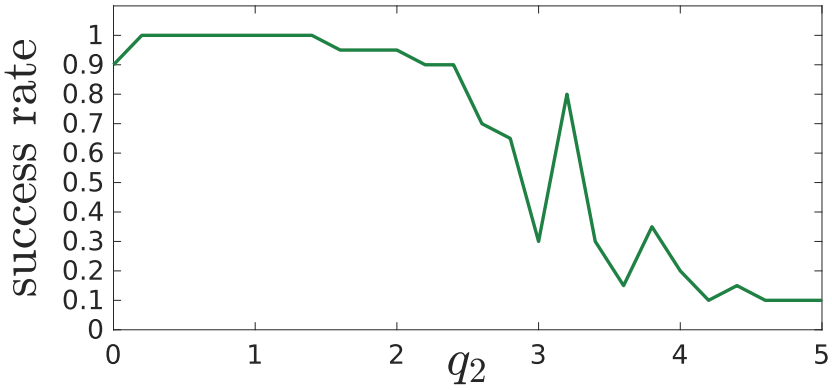


Figure 4.6: Curvature-Aware MPCC parameter sweep.

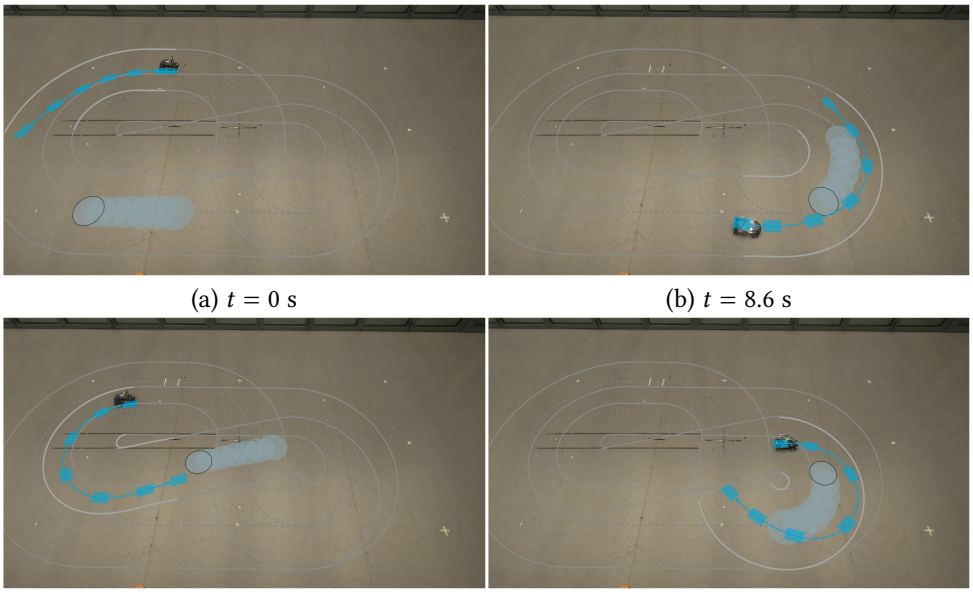
Figure 4.5 shows the results for the baseline MPCC. As evident from the figure, the tuning of the baseline MPCC can be extremely challenging. In contrast, Figure 4.6 shows the results for the Curvature-Aware MPCC. This figure shows that below a certain threshold of q_2, the behaviour of the closed loop system is very reliable, since the success rate is close or equal to 100%. Over this threshold the success rate gradually decreases. The failures are entirely due to the vehicle no longer being able to overtake the dynamic obstacle. This behaviour is to be expected for very large ε^c weights. These results support our claim that the CA-MPCC is easier to tune, since it features one less parameter in the cost function, and the closed-loop behaviour is much more consistent within a certain q_2 range. Furthermore, this range can be conservatively estimated by keeping $q_2 \leq q_1$ (in the tests the weights are normalized with respect to q_1). This would generally be the case in real life application since lower lateral error gains result in smoother robot trajectories.

4.4.2 Experiments

We validate our findings on a real mobile robot, a 1:10 scaled-down car-like vehicle, a modified Waveshare JetRacer Pro AI kit. Since localization is not the focus of the present work, we relied on a Motive OptiTrack motion capture system to provide the vehicle state estimate to the closed-loop controller, that is, we use external sensors to provide the position and velocity states of the system. To control the vehicle a dedicated ROS network was set up. The navigation algorithm runs on a laptop featuring an AMD Ryzen 7 5800H processor. The control loop data flow is thus: *1* The external motion capture system measures the vehicle position and velocity, *2* sends it over a LAN cable to the laptop, *3* optimization algorithm is solved using forcespro software, *4* input signal is sent over WiFi to the vehicle, *5* on-board signal converter translates the inputs to actuator PWM values. We set a time horizon of $1.5s$ discretized in $0.1s$ steps. To numerically solve the optimization problem we rely on forcespro [107]. As for the simulations we task both the baseline MPCC and the CA-MPCC with driving along a track that features sharp turns while overtaking a slow-moving virtual dynamic obstacle. Figure 4.7 shows the CA-MPCC

performing a circuit lap.

We fixed the lateral error weight $q_2 = 0.5$ for the baseline MPCC. The value was selected after some trial-error experiments to achieve a consistent vehicle behavior throughout all the experiments. Figure 4.8 examines the effects of different q_3 weights on the velocity tracking error of the MPCC. For low q_3 values the velocity drops significantly during some turns ($t \approx 18s, 27s$). This is due to the error in lane boundary evaluation, as anticipated in Section 4.2.2. During the whole experiment the solver converged to a seemingly valid solution, making this kind of failure difficult to identify. For higher q_3 weights this behaviour becomes progressively less frequent and is absent in the final test. However, a drop in velocity is still present in the tightest bend of the circuit ($t \approx 23s$). This behaviour is due to the lag error term being too large compared to the velocity tracking term. The open-loop solutions are similar to those presented in Figures 4.1(a) and 4.2(a) for low q_3 and high q_3 respectively, as can be seen in the video attached to this submission. To test the CA-MPCC algorithm we ran some tests changing the lateral error weight. Figure 4.9 shows the $\dot{s}$ tracking performance during the experiments. We notice drops in $\dot{s}$ in the experiments featuring $q_2 = 0.5$ and $q_2 = 1$ ($t \approx 24s$), yet this failure is due to modelling errors in the system dynamics, rather than to the control algorithm. This can be seen from the difference between the open-loop and closed-loop behaviour as is evident from the video footage[108]. In both sets of experiments we observe a constant bias in the velocity or $\dot{s}$ tracking, this is likely due to a modelling error in the correlation between τ and $\dot{v}$ (which affects the experiments with both algorithms). Compared to the standard MPCC formulation we notice that the closed-loop behaviour is much more consistent, even for very large parameter variations, this is in accordance with the simulations. Concerning the computation times, we measured the full control loop (i.e. including the polynomial fitting operation, which takes on average 3 ms). The results are summarised in table 4.2. The CA-MPCC does feature higher computation times, yet is still achieves real-time feasibility. It is important to mention that it may be possible to reduce computation time by tuning the number of Chebyshev polynomial bases that represent the local path. For our experiments we used 20 polynomial bases, yet 10 could also have been sufficient.



(a) $t = 0$ s (b) $t = 8.6$ s

(c) $t = 16.6$ s (d) $t = 22.4$ s

Figure 4.7: CA-MPCC performing the circuit at different time instants.

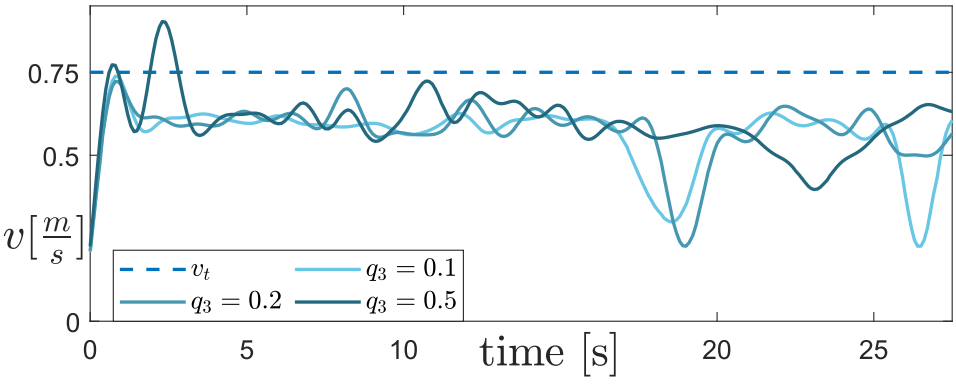


Figure 4.8: Velocity profile during baseline MPCC experiments.

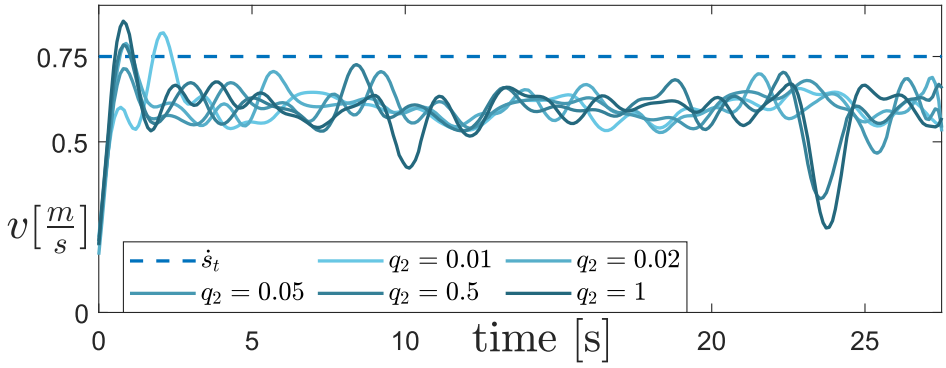


Figure 4.9: $\dot{s}$ profile during CA-MPCC experiments.

	Mean(ms)	**95th Percentile**(ms)	**Worst-case**(ms)
MPCC	19.7	28.2	44.2
CA-MPCC	28.6	52.3	65.0

Table 4.2: Comparison of computation times (ms) for MPCC and CA-MPCC experiments.

4.5 Conclusions

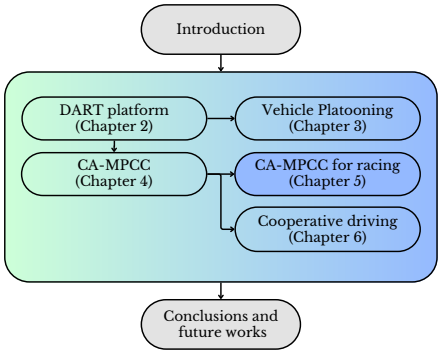
This work presented an online Curvature-Aware MPCC formulation capable of handling sharp turns in the reference path, while avoiding collisions with dynamic obstacles. We compared our method against a baseline MPCC formulation and proved in both simulation and experiments that the proposed method is both easier to tune and more reliable. Future research directions are to extend this method to 3D motion planning and racing where no previous knowledge of the racetrack is available, yet effort should be directed towards further reducing computation times.

5

Curvature-Aware MPCC for racing

Overview

In this chapter, we present rCA-MPCC[1], an extension of the CA-MPCC algorithm presented in chapter 4 designed for high-speed autonomous navigation through the lens of autonomous racing. Racing is used here as a proxy for challenging real-world scenarios in which robots must operate near the limits of their dynamic capabilities, typically in emergency manoeuvres.



The proposed rCA-MPCC framework extends the recently introduced CA-MPCC formulation in three key ways. First, it incorporates a curvature-aware terminal cost that anticipates how the track evolves beyond the prediction horizon, allowing the controller to cut corners effectively without relying on a pre-computed racing line. Second, it introduces a lightweight, curvature-based representation of the reference path that improves numerical consistency. Third, the method is generalized to both 2D ground-vehicle dynamics and 3D quadrotor flight, providing unified formulations together with publicly available implementations and datasets.

We evaluate rCA-MPCC through a comprehensive comparison with MPCC++ [33], a recent state-of-the-art contouring-control formulation developed for autonomous drone

[1]This chapter is based on *Curvature-Aware MPCC for autonomous racing* by Lorenzo Lyons, Andrei Papuc, Sihao Sun and Laura Ferranti. L. Lyons' contribution is developing the method, performing the dynamic modeling of the autonomous car racing platform, performing the experimental validation of the autonomous car racing, performing the method validation in simulation and writing the manuscript. A. Papuc's contribution is providing the drone's dynamic model. S. Sun and L. Ferranti supervised the research. At the time of publication of this thesis, the paper has not yet been submitted for publication.

racing, adapting it for the ground-vehicle experiments when needed. Within this framework, we study the influence of model fidelity, including actuator dynamics modelling and Gaussian-process-based residual modelling prediction horizon length, the choice between single, and multi-layer MPC architectures, delay-compensation strategies at runtime, and automatic gain tuning based on Bayesian optimization. Across all experiments, rCA-MPCC exhibits more repeatable open-loop predictions, improved solver consistency, and a more effective exploitation of track curvature, resulting in faster and more stable lap times.

Overall, rCA-MPCC offers a robust and computationally efficient formulation that reduces dependence on prior knowledge of the track and improves closed-loop control at the limits of handling. By releasing all software and datasets, this chapter provides a reproducible reference for high-speed navigation research and contributes practical insights that extend beyond racing to real-world robotic systems operating in emergency scenarios.

5.1 Introduction

Autonomous systems such as autonomous vehicles, ground mobile robots (AGVs) and drones (UAVs) are becoming ever more ubiquitous, with worldwide reports of a growing uptake in many sectors, such as passenger transportation [109], logistics [110], warehousing [111], agriculture [112] and surveillance [113].

Despite this growing trend and the already widespread adoption of autonomous mobile robots, typical speeds in real-world deployments remain low, leaving the hardware capabilities largely under-utilized, limiting the benefits this technology can bring to our society [114–117].

Conversely, in autonomous racing settings there is a significant number of works where the overall system performance is operating on the edge of the physical limits of the platform, this is the case for both ground [118] and aerial vehicles [119]. This is because in racing it's typical to have more prior information about the environment, compared to real-world applications. For example, the racing track is usually known in advance, thus reducing the scope of possible scenarios the robot will encounter, allowing for more tailored solutions. Despite a significant number of works from the autonomous racing literature consider the racetrack as a proxy for challenging navigation in general, often stating 'bridging the hardware-utilization gap' as a motivation for their findings, generalization beyond the racetrack is often deferred to future work. Factors that limit this knowledge transfer are the perfect prior knowledge of the racetrack, the lack of detail concerning key implementation choices in the control algorithm design and the impact they may have on overall racing performance, the lack of publicly shared code, and the separation between the autonomous vehicle and drone racing communities. As a result the knowledge transfer from racing to real-world applications remains an ongoing process.

Our work focuses on control algorithms for autonomous racing, and contributes to the knowledge transfer towards real-world use cases by presenting a novel racing control algorithm that relaxes the assumptions on the prior knowledge of the racing track. We address the divide between different autonomous racing communities by formulating the same algorithm for both 2D ground-robot and 3D aerial robot navigation, sharing all the code in a public repository. Furthermore, we present an extensive testing campaign on real robotic platforms that shows the impact of technical implementation details, such as different vehicle dynamics models, and different controller structures.

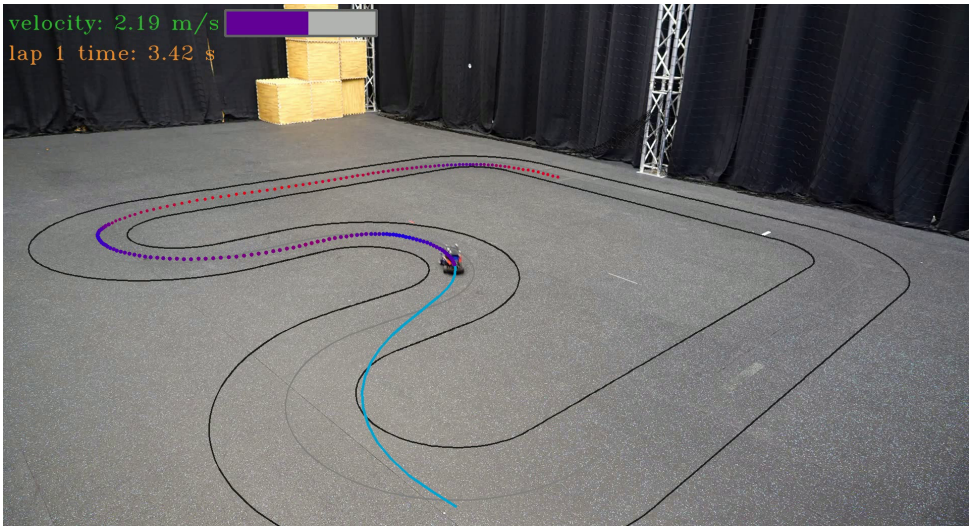


Figure 5.1: Experimental lab setup for autonomous car racing on a scaled-down car-like robot. Our rCA-MPCC open-loop prediction is shown in light blue, while the past vehicle trajectory is colour coded according to the vehicle's velocity.

5.1.1 Related works

When approaching autonomous racing, it is easy to be overwhelmed by the vastness of the field. It is a complex multidisciplinary task, posing challenges across different robotics sub-fields such as perception, state estimation, dynamics modelling and control algorithm design. We refer the interested reader to [119] and [118] for an overarching view on how these different aspects contribute to the overall performance in drone and car racing, respectively. While all of these aspects are essential for achieving competitive racing performance, we restrict our study to control algorithm design and vehicle dynamics modelling.

Concerning controllers for autonomous racing, there is a grand divide among the community over model-based and learning-based approaches. Both have their merits and their shortcomings. On one hand, learning-based approaches offer the ability to directly process high-dimensional sensory inputs and jointly solve perception, planning, and control without the need for explicit models. They also tend to be more robust to sensor noise and delays when such effects are included in training, and reduce the engineering effort associated with precise system modeling. On the other hand, they feature limited interpretability, and a lack of formal safety and stability guarantees. Most data-driven policies, such as RL, require numerous trials during training, often with initial policies producing unsafe behaviours. Despite works such as [120] and [121] have proposed methods to mitigate the associated risks, in both ground and aerial applications respectively, training RL policies on real hardware remains impractical, leaving this kind of approach suffering from the so called sim-to-real gap.

Conversely, traditional model-based approaches, such as MPC [122], typically rely on physics-based modeling of the system dynamics. These methods offer low computational

cost at inference, but require expert domain knowledge and often substantial effort to develop accurate models.

Despite the differences between model-based and learning-based approaches, several studies, such as [33], have shown that both can achieve comparable racing performance. Beyond a certain threshold, the main bottleneck is no longer the control algorithm, but the hardware itself. Consequently, once this threshold is reached, it is reasonable to expect similar overall performance across approaches, whether learning or model-based.

In our work we have narrowed the scope of our research to MPC-based approaches, since they are more explainable, are more intuitive to tune, mitigate the sim-to-real transfer problem, and are more adaptable since they do not depend on task-specific training datasets, making them more suited for generalizing beyond the racetrack.

In a typical racing context, where the racetrack is known in advance, an effective strategy is to solve the optimal trajectory offline, and subsequently extract a portion of this optimal trajectory to use it as a reference for a low-level tracking MPC [123]. Differently from traditional MPC formulations, where the reference is time-scheduled, the low-level controller in racing applications is typically designed to track a *spatially*-scheduled reference. The centreline of the racetrack is parametrized on its natural arc length s. The vehicle's progress is then defined as the arc length s corresponding to the point on the centreline that lies closest to the vehicle's current position. This particular MPC formulation is referred to as *Contouring* MPC (MPCC) and was originally designed for machining applications [124]. Owing to its original application, MPCC is reliable if the reference can be tracked very closely, but may fail if the lateral deviation from the path becomes significant, as pointed out in [50]. To mitigate this effect, approaches such as [33] directly incentivise making progress along the path in the MPCC cost function, promoting high values of $\dot{s}$ in the stage-wise cost, and overall achieved path progress in the terminal cost. To ensure recursive feasibility however, the authors still rely on optimizing the race line offline and a terminal set constraint that ensures the robot's last predicted state lies on this pre-optimized trajectory.

In this work we have identified [33] as a competitive baseline to evaluate our proposed method, yet we aim to remove the need for prior knowledge of the track completely by relaxing the terminal constraint. To do so, we build upon the Curvature-Aware MPCC (CA-MPCC) in [50], originally designed for urban driving, and leverage the knowledge of path curvature to design a terminal cost that better approximates the evolution of the path beyond the last open-loop stage.

In order to provide a direct comparison between the two approaches in both autonomous vehicle and drone racing, and to bridge the gap between the two communities, we formulate our racing algorithm for both scenarios. To compare our method against [33] in ground vehicle racing, we reformulate the latter, that was designed for dorne racing, to fit the 2D case. To favour our algorithm's uptake beyond pure racing scenarios, we conduct an extensive experimental study to evaluate the impact of various implementation techniques and guide future engineering efforts toward the most critical aspects.

5.1.2 Contributions

This paper presents a novel Curvature-Aware MPC algorithm tailored to ground and aerial robot racing, namely rCA-MPCC. We build upon the MPC presented in [50] for

autonomous vehicle navigation in urban scenarios. We address its current limitations in order to successfully deploy it in the field of autonomous racing. The main contributions of this work are summarized as follows:

- We extend the urban-driving MPC in [50] to autonomous racing on ground and aerial platforms. Namely:

  i) We remove velocity-tracking terms from the stage-wise cost function J_k;

  ii) We implement a terminal cost that incentivises progress along the path, and, by relying on the curvature information, we add a term that pushes trajectories toward the curve interior, resulting in closed-loop trajectories that fully exploit "cutting corners" to make faster progress along the racetrack.

  iii) We provide a novel approach to describe the reference path, i.e. the racetrack centerline, by integrating its second order derivative. This reduces the complexity required to describe the reference path, leading to more stable MPC open-loop trajectories and reduced overall lap time.

  iv) We extend the framework to 3D drone applications.

- Concerning autonomous car racing, we test our rCA-MPCC on the experimental platform presented in [36], shown in Figure 5.1. In order to push the vehicle to the limits of handling we augment the dynamical bicycle model provided in [36] by identifying the steering actuation dynamics, allowing us to collect data during high speed operations and build a Gaussian Process (GP) residual dynamic model. We furthermore provide a working code package to integrate the GP model in the MPC solver, this is shared in the public GitHub repository [125].

- We evaluate our rCA-MPCC against the state of the art MPC in [33], adapting the latter to 2D navigation for the car racing tests. We provide an extensive test campaign on real platforms, also showing the impact of critical design choices on the overall lap time performance. In particular we investigate: single-layer versus hierarchical MPC, the effect of delay-compensation strategies at runtime, the effect of automating gain tuning with Bayesian Optimization (BO) strategies, and the effect of adding an SVGP model of the system's residual dynamics, both in the BO gain-tuning phase and in the MPC dynamic constraint.

5.1.3 Paper organization

The remainder of this paper is structured as follows. Section 5.2 introduces the problem formulation. Section 5.3 presents the proposed rCA-MPCC scheme, detailing the curvature-aware terminal cost and reference-path representation. Section 5.4 describes the dynamic models for ground and aerial robots. Section 5.5 provides validation on ground and aerial platforms. Finally, Section 5.6 summarizes the main findings.

5.2 Problem formulation

We consider a generic mobile robot described by the following continuous-time dynamic equations:

$$\begin{bmatrix} \dot{\boldsymbol{x}}^{\text{rob}}(t) \\ \dot{\boldsymbol{v}}(t) \end{bmatrix} = f^{\text{sys}}\left(\begin{bmatrix} \boldsymbol{x}^{\text{rob}}(t) \\ \boldsymbol{v}(t) \end{bmatrix}, \boldsymbol{u}(t) \right). \tag{5.1}$$

Where f^{sys} denotes the dynamic model of the mobile robot, $\boldsymbol{u}(t)$ is the control input at time t, $\boldsymbol{v}(t)$ is the velocity state vector of the system, and $\boldsymbol{x}^{\text{rob}}(t)$ includes the robot-specific states, such as actuator states or the attitude angles of a quadrotor, that are necessary to compute the robot's velocity $\boldsymbol{v}(t)$. Note that for mobile robots, the dynamics f^{sys} generally do not depend on the position of the robot, but only on its velocity. We now extend the state vector by adding the position dynamics, obtained by directly integrating the velocity states.

$$\begin{bmatrix} \dot{\boldsymbol{x}}^{\text{rob}}(t) \\ \dot{\boldsymbol{v}}(t) \\ \dot{\boldsymbol{x}}^{\text{pos}}(t) \end{bmatrix} = \begin{bmatrix} f^{\text{sys}}\left(\begin{bmatrix} \boldsymbol{x}^{\text{rob}}(t) \\ \boldsymbol{v}(t) \end{bmatrix}, \boldsymbol{u}(t) \right) \\ \boldsymbol{v}(t) \end{bmatrix}, \tag{5.2}$$

where $\boldsymbol{x}^{\text{pos}}$, $\boldsymbol{v}$, $\dot{\boldsymbol{x}}^{\text{rob}}$, $\dot{\boldsymbol{v}} \in \mathbb{R}^n$, n=2, 3 for ground and aerial robots, respectively.

5

5.2.1 Contouring MPC

In a Contouring MPC formulation, the objective is to steer the position of a mobile point $\boldsymbol{x}^{\text{pos}}(t)$ along a reference path $\boldsymbol{x}^{\text{ref}}(s)$, where the latter is parametrized by the arc-length parameter s. In addition to the states in Eq. (5.2), we therefore augment the system's state with the value of $s(t)$ that describes the point on $\boldsymbol{x}^{\text{ref}}(s)$ closest to the robot's current position $\boldsymbol{x}^{\text{pos}}(t)$. Finding $s(t)$ as the closest point to the robot on the reference path entails solving the following optimization problem:

$$s(t) \;=\; \underset{s \in \mathcal{S}}{\arg\min} \; \left\| \boldsymbol{x}^{\text{ref}}(s) - \boldsymbol{x}^{\text{pos}}(t) \right\|_2 . \tag{5.3}$$

Directly solving equation (5.3) inside the MPCC optimization step would make the latter non-differentiable and is thus not a viable solution in practice. The typical workaround is to solve Eq. (5.3) once at the beginning of each MPCC control loop to initialize the augmented state, i.e. to find $s(0)$, and the subsequent values of $s(t)$ in the open-loop solution are then predicted by approximating $\dot{s}(t) \approx \dot{\tilde{s}}$. We can now write the full augmented system state as

$$\begin{bmatrix} \dot{\boldsymbol{x}}^{\text{rob}}(t) \\ \dot{\boldsymbol{v}}(t) \\ \dot{\boldsymbol{x}}^{\text{pos}}(t) \\ \dot{s}(t) \end{bmatrix} = \begin{bmatrix} f^{\text{sys}}\left(\begin{bmatrix} \boldsymbol{x}^{\text{rob}}(t) \\ \boldsymbol{v}(t) \end{bmatrix}, \boldsymbol{u}(t) \right) \\ \boldsymbol{v}(t) \\ \dot{\tilde{s}} \end{bmatrix}. \tag{5.4}$$

In the traditional MPCC formulation [124], $\dot{s}(t)$ is approximated as the Euclidean norm of the velocity, that is,

$$\dot{\tilde{s}} = \left\| \boldsymbol{v}(t) \right\|_2 . \tag{5.5}$$

This approximation is sufficiently accurate as long as the predicted distance between the robot and the path, i.e. the *contouring* error ϵ_c, remains small. As ϵ_c increases and

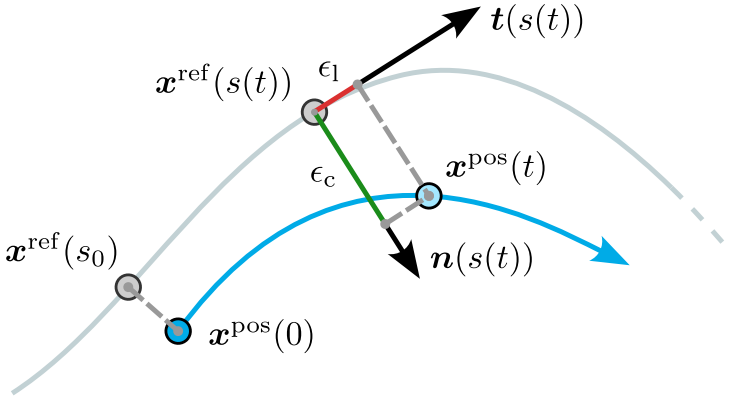


Figure 5.2: Traditional MPCC open-loop trajectory (light blue) during sharp centerline curve (gray).

5

Eq. (5.5) accumulates integration errors, and the predicted closest point on the reference path deviates from the actual one (see [50] for further details). The contouring and lag errors ϵ_{c} and ϵ_{l} are then defined as the projection of the distance between the robot and the reference path in the local path frame:

$$\begin{aligned} \epsilon_{\text{c}}(t) &= \boldsymbol{n}(s(t)) \cdot \left(\boldsymbol{x}^{\text{pos}}(t) - \boldsymbol{x}^{\text{ref}}(s(t))\right), \\ \epsilon_{\text{l}}(t) &= \boldsymbol{t}(s(t)) \cdot \left(\boldsymbol{x}^{\text{pos}}(t) - \boldsymbol{x}^{\text{ref}}(s(t))\right). \end{aligned} \tag{5.6}$$

Where $\boldsymbol{n}(s(t))$ and $\boldsymbol{t}(s(t))$ denote the unit normal and tangent vectors to the reference path at $\boldsymbol{x}^{\text{ref}}(s(t))$ respectively, see Figure 5.2. Large values of ϵ_{l} are undesirable, since they may lead to erroneous evaluation of lane boundary constraints and cause the robot's open-loop solution to violate the real lane boundaries as described in [50]. Additionally, Eq. (5.5) will systematically underestimate $\dot{s}$ when travelling on the inside of a curve, and likewise systematically overestimate the progress when travelling outside the curve. As a result, using Eq. (5.5) in an MPCC for racing applications will result in an under-exploitation of the racetrack's curvature, i.e. the algorithm is blind to the advantage of "cutting corners".To circumvent this issue we have found two approaches in the literature.

MPCC++. The first solution is the one adopted in [33], where the authors provide $\dot{\tilde{s}}$ as an input rather than a function of the robots' state, and then subsequently enforce the condition (5.3) as a soft constraint that penalizes non-zero values of $\epsilon_{\text{l}}(t)$. This formulation shares strong similarities with the typical approach for offline optimal racing trajectory

optimization, where the problem is typically formulated in discrete-time as:

$$\min_{\boldsymbol{x}_k,\, \boldsymbol{u}_k,\, \Delta t_k} \quad \sum_{k=0}^{N-1} \Delta t_k, \tag{5.7a}$$

$$\text{s.t.} \quad \boldsymbol{x}_{k+1} = f^{\text{sys}}(\boldsymbol{x}_k,\, \boldsymbol{u}_k)\,, \tag{5.7b}$$

$$\boldsymbol{x}_{k+1} \in \text{track boundaries}, \tag{5.7c}$$

$$\boldsymbol{n}_{k+1} \cdot \left(\boldsymbol{x}_{k+1} - \boldsymbol{x}^{\text{ref}}(s_{k+1})\right) = 0, \tag{5.7d}$$

$$\boldsymbol{u}_{\min} \leq \boldsymbol{u}_k \leq \boldsymbol{u}_{\max}, \tag{5.7e}$$

$$\boldsymbol{x}_{\min} \leq \boldsymbol{x}_{k+1} \leq \boldsymbol{x}_{\max}, \tag{5.7f}$$

$$\boldsymbol{x}_0 = \boldsymbol{x}_N. \tag{5.7g}$$

5

Where the progress along the racetrack s_k is scheduled a priori by spatially discretizing the racetrack, and Δt_k, the time step between the two points, is the optimization variable to be minimized. Similarly, in [33], $\dot{s}$ is treated as an optimization variable, and the algorithm is encouraged to select large positive values by introducing a negative penalty in the stage-wise cost function, serving as a proxy for lap-time minimization. Notice that the constraint (5.7d) is equivalent to $\epsilon_{\text{l}}(t) = 0$, and enforces that the robot's position is correctly projected on the reference path. This constraint is relaxed in [33], and enforced by adding a high penalty on $\epsilon_{\text{l}}(t)$ in the stage-wise cost function. Constraint (5.7g) enforces periodicity and consequently recursive feasibility over following laps. In [33] the recursive feasibility property is enforced by adding a terminal constraint (Eq.(8) in [33]) that forces the last robot state to lie on the offline optimal solution obtained by solving problem (5.7).

CA-MPCC. The second way of addressing the issues related to the use of Eq. (5.5) in the standard MPCC formulation, is to increase the accuracy of the $\dot{s}$ approximation. This is the approach adopted in [50], where the authors also consider the curvature information of the reference path, proposing a Curvature-Aware approximation of the estimated progress along the path. Owing to this better approximation of $\dot{s}$, the lag error remains negligible along the open-loop prediction and is removed from the stage-wise cost function, simplifying the gain tuning process. Furthermore, the expression for ϵ_{c} can be simplified to:

$$\epsilon_{\text{c}} = \|\boldsymbol{x}^{\text{pos}} - \boldsymbol{x}^{\text{ref}}\|_2. \tag{5.8}$$

This can be proven by considering that $\|\boldsymbol{x}^{\text{pos}} - \boldsymbol{x}^{\text{ref}}\|_2^2 = \epsilon_{\text{c}}^2 + \epsilon_{\text{l}}^2$, so if $\epsilon_{\text{l}} \approx 0$ Eq. (5.8) can be trivially derived.

5.3 Curavature-Aware mpcc for racing

In this section, we build on the urban driving MPCC in [50] to formulate our racing Curvature-Aware MPCC (rCA-MPCC). We formulate the rCA-MPCC algorithm as follows:

$$\min_{\boldsymbol{x}_k, \boldsymbol{u}_k} J_N + \sum_{k=0}^{N} q_c \epsilon_c + \boldsymbol{u}^\top Q_u \boldsymbol{u} \tag{5.9a}$$

$$\text{s.t.: Eq. (5.4)}, \tag{5.9b}$$

$$\boldsymbol{x} \in \mathcal{X},\ \boldsymbol{u} \in \mathcal{U}, \tag{5.9c}$$

$$\epsilon_c^2 \leq r \tag{5.9d}$$

$$\boldsymbol{x}_0 = \boldsymbol{x}_{\text{init}}, \tag{5.9e}$$

where Q_u is a diagonal matrix containing the actuation weights. Eq. (5.9c) constraints the robot to remain within a lane of half width r, and Eqs.(5.9c) are state and actuator bounds. rCA-MPCC differs from [50] in the following ways:

i) We modify the stage-wise cost function J_k, where k refers to the stage $k \in (0, N)$ in the MPC formulation, by removing the $\dot{s}$ tracking term in the stage-wise cost function J_k.

ii) We implement a terminal cost, defined as:

$$J_N = -q_s^t(s_N - s_0) + q_\kappa^t \alpha_\kappa^2 + q_{\text{pos}}^t \|\boldsymbol{x}_N^{\text{ref}} - \boldsymbol{x}_N^{\text{pos}}\|_2^2, \tag{5.10}$$

where q_s^t and q_κ^t are tuning parameters and $(s_N - s_0)$ is the total progress along the path. The term α_κ is defined as:

$$\alpha_\kappa = 1 - \epsilon_c \kappa, \tag{5.11}$$

where κ is the path curvature. Notice that α_κ is the denominator in the expression for the so-called projection ratio, referred to as σ in [50], the latter is part of the expression for $\dot{\tilde{s}}$:

$$\dot{\tilde{s}} = (\boldsymbol{v} \cdot \boldsymbol{t})\sigma. \tag{5.12}$$

Minimizing α_κ in the final stage of the open-loop prediction maximizes σ, thus pushing the end-point of the open-loop trajectory towards the inside of a curved path. Notice that if the path presents no curvature, then $\alpha_\kappa = 1$, and has no effect on the optimization process. The term $q_{\text{pos}}^t \|\boldsymbol{x}_N^{\text{ref}} - \boldsymbol{x}_N^{\text{pos}}\|_2^2$ pushes the final position of the robot towards the centre of the reference path. This term provides the recursive feasibility property, similarly to the hard constraint in Eq.(8) in [33], yet is implemented as a soft constraint without relying on off-line solutions.

Characterizing the algorithm for 2D or 3D applications is relatively straightforward from a conceptual point of view. The main difference lies in the definition of the mobile robot (5.4), where the system specific component f^{sys} needs to be carefully modelled. Consequently, depending on the dimensionality of the output $\boldsymbol{v}$, the quantities $\boldsymbol{x}^{\text{pos}}$, $\boldsymbol{x}^{\text{ref}}$, $\boldsymbol{t}$, $\boldsymbol{n}$ will match the system's out dimensionality. Note that the definition of ϵ_c for CA-MPCC frameworks in (5.8) remains valid for both 2D and 3D settings.

iii) We modify the reference path representation. The typical approach in many MPCC frameworks, including [33], is to provide the reference path by using a separate function

to describe each component of $\boldsymbol{x}^{\text{ref}}(s)$ and its first order derivatives $\boldsymbol{t}(s)$, while [50] also requires the second order derivatives $\boldsymbol{n}(s)\kappa$. In order for the MPC solver software to solve the optimization problem reliably and quickly, the functions used to describe each component of the reference path needs to be continuously differentiable, while still flexible enough to represent any reference path. To guarantee these properties, any effective approach relies on describing the required 1-dimensional quantities with a sum of continuously differentiable basis functions:

$$f(s) = \sum_{i=1}^{K} w_i \, \phi_i(s), \tag{5.13}$$

where $\phi_i(s)$ are the basis functions that can be hard-coded in the MPCC solver, thus providing differentiability and fast inference times, while the weights w_i are left as free parameters and are passed on to the solver at each MPC control loop, providing the flexibility needed to describe different reference paths. The most common choice for ϕ_i in the literature is to use B-splines [126], as in [33], yet other choices are possible, for example in [50] the authors use Chebyshev polynomials [127]. In this work, we use kernelized linear regression, that coincides with the posterior mean of a Gaussian process (GP) with the same kernel and regularization settings [128]. This choice lets us modulate the smoothness and expressiveness of the basis functions $\phi_i(s)$ simply by tuning the kernel hyperparameters and adjusting the number of the data points used to describe the path, rather than explicitly increasing the number of basis functions as in [33] and [50].

Regardless of the specific choice of the basis functions, evaluating and differentiating through Eq. (5.13) remains numerically expensive, since it effectively entails evaluating K basis functions. To reduce the number of f functions in the optimization problem, we exploit the geometric meaning of the path curvature and only use a single f function. For the 2D navigation case we define the signed curvature as:

$$\kappa^{\text{s}} = \kappa(\boldsymbol{n} \cdot R^{+90}\boldsymbol{t}), \tag{5.14}$$

where R^{+90} is the rotation matrix corresponding to a 90°counter clockwise rotation. The value of κ^{s} will therefore be $\pm\kappa$ depending on whether the inside of the curvature $\boldsymbol{n}$ points in the same direction as the "left" direction with respect to the current path direction $\boldsymbol{t}$. This allows to describe the reference path as a dynamical system:

$$\dot{\theta} = \kappa^{\text{s}} \, \dot{s}, \tag{5.15a}$$

$$\dot{\boldsymbol{x}}^{\text{ref}} = \boldsymbol{t}(\theta) \, \dot{s}, \tag{5.15b}$$

where θ is the angle that describes the orientation of $\boldsymbol{t}$, i.e.

$$\boldsymbol{t}(\theta) = \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix}. \tag{5.16}$$

We thus further augment the state of the robot in Eq. (5.4) with θ and $\boldsymbol{x}^{\text{ref}}$. This implementation technique not only reduces the computational complexity owing to fewer f function evaluations, but also ensures strict coherence between the components of $\boldsymbol{x}^{\text{ref}}$ and its first and second order derivatives, leading higher solver convergence rates for the 2D

implementation. Notice however that this approach doesn't scale well for 3D applications. This is because, while κ^s can describe the curvature in the osculating plane in the same way for 2D and 3D, an additional f function is needed to describe the rotation of the osculating plane around $\boldsymbol{t}$ for the 3D case. Furthermore, while in 2D only one angle θ is enough to univocally describe the tangential direction $\boldsymbol{t}$, in 3D settings, all 3 attitude angles are needed, and deadlock may still occur if the pitch angle reaches $\pm\pi/2$. For this reasons we use Eqs. (5.15) only for the 2D case, and use the standard approach of one f function for each component of $\boldsymbol{x}^{\text{ref}}$ and its derivatives in the 3D case.

5.4 Dynamic modelling

In this section, we characterize the dynamic models used in the MPCC formulations, i.e. the f_{system} component in Eq. (5.4).

5.4.1 Vehicle dynamics

Concerning ground vehicle racing, we test our algorithm on DART [36], a 1:20 scale car-like robot, yet our method is suitable for any autonomous vehicle platform. In their work, the authors perform quasi-static tests to identify the tire model and use it to build a dynamic bicycle model. The tests consist in driving the vehicle in circles, slowly changing the steering angle and measuring the tire forces. This approach yields a reliable dynamic model that is accurate up to medium-high speeds of up to 3m/s. In this work however, we intend to push the vehicle to the limits of handling, and thus need to refine the model by collecting data during high speed cornering.

Actuator dynamics modelling. Collecting data during high-speed operation presents an additional challenge: since in [36] the authors perform quasi-static tests, the dynamics of the steering servomotor can be neglected, while at high speed they become critical to collect meaningful data. On the left side plot in Figure 5.3 we show the lateral wheel force data for the front tire during a preliminary test using the rCA-MPCC. The model now assumes that the actual steering angle coincides with the steering input coming from the controller, leading to overestimating the slip angle during aggressive cornering. The measured force will therefore be associated with a much larger slip angle, producing the horizontal data dispersion visible in the plot.

In order to collect usable data during high-speed cornering, we must identify the steering dynamics in order to accurately estimate the real slip angle. Note that the dynamics of the longitudinal electric motor have a less evident effect, as longitudinal accelerations are modest compared to lateral ones, however, they are still included in the model to improve overall accuracy.

To capture the actuator dynamics, we use a finite impulse response (FIR) model to filter the steering and throttle inputs, and feed the filtered inputs to the dynamic bicycle model. A FIR model is expressed as:

$$\hat{u}[k] = \sum_{m=0}^{M} b_m \, u[k-m], \quad \text{s.t} \sum_{m=0}^{M} b_m = 1, \tag{5.17}$$

where $\hat{u}[k]$ is the filtered input at time k, $u[k-m]$ is the input supplied to the system at time $k-m$, and b_m is the weighting coefficient corresponding to a delay of m steps in the

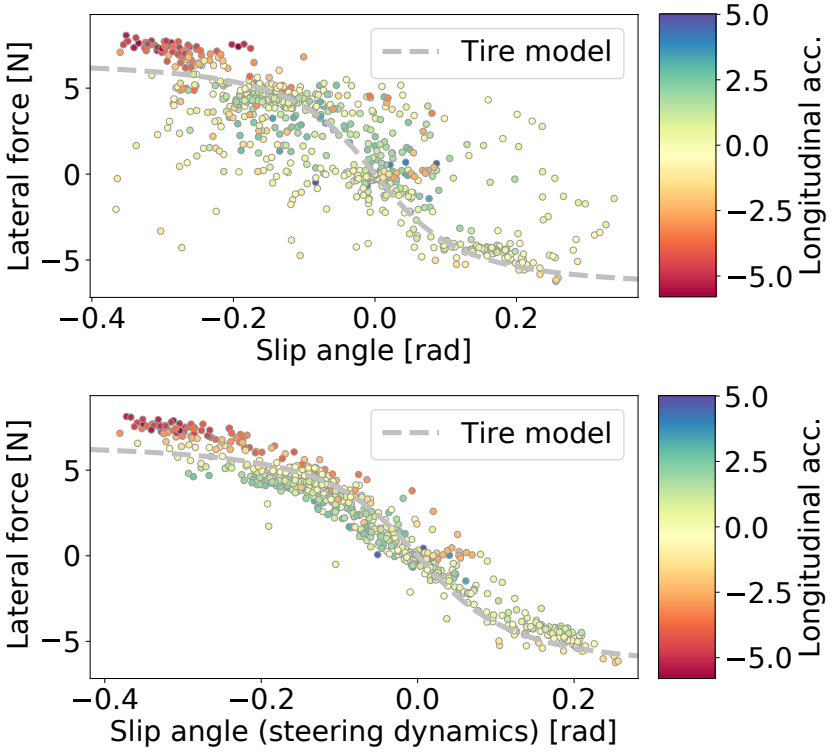


Figure 5.3: Measured front wheel lateral force data as a function of the slip angle recorded during rCA-MPCC operating on the racetrack. Left: not accounting for actuator dynamics. Right: with actuator dynamics. Not considering the actuator dynamics leads to a mismatch between the measured lateral force and the assumed slip angle, producing the data dispersion visible in the left plot. The figure also shows the effect of the longitudinal acceleration (colorbar). High braking (negative longitudinal acceleration) transfers more weight to the front wheel, increasing the lateral force. Conversely, high positive longitudinal accelerations shift weight to the rear tires, resulting in less lateral force on the front tires.

past. Notice that $\sum_{m=0}^{M} b_m = 1$ enforces a conservation of energy between the original and the filtered signal.

To identify the coefficients b_m we use the M data points shown in Figure 5.3 to solve the following minimization problem:

$$\min_{\mathbf{b_m}} \sum_{n=0}^{N} \tfrac{1}{2}\, \mathbf{e}_n^\top \mathbf{W}\, \mathbf{e}_n, \tag{5.18}$$

where $\mathbf{b_m}$ collects the b_m coefficients, $\boldsymbol{W}$ is a diagonal weighting matrix, and $\mathbf{e}_n$ is the error between the measured vehicle acceleration and the estimated ones by feeding the filtered throttle $\hat{\tau}$ and steering angle $\hat{\delta}$ into the dynamic bicycle model in [36]:

$$\mathbf{e}_n = \begin{bmatrix} \ddot{x}_n - \ddot{\hat{x}}(v_{x,n}, v_{y,n}, \omega_n, \hat{\tau}, \hat{\delta}) \\ \ddot{y}_n - \ddot{\hat{x}}(v_{x,n}, v_{y,n}, \omega_n, \hat{\tau}, \hat{\delta}) \\ \dot{\omega}_n - \dot{\hat{\omega}}(v_{x,n}, v_{y,n}, \omega_n, \hat{\tau}, \hat{\delta}) \end{bmatrix}. \tag{5.19}$$

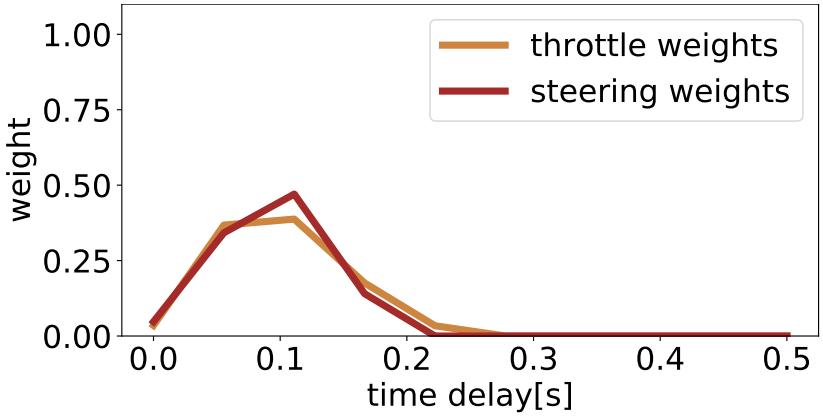


Figure 5.4: Finite Impulse Response model of the steering actuator dynamics. The delayed peak indicates a deadbeat delay of 0.1s, while the trailing exponential decay indicates a first-order-like system dynamics behaviour.

The resulting FIR filtering coefficients are shown in Figure 5.4.

Gaussian Process residual dynamics. Once the actuation dynamics have been successfully identified, we can collect valuable data during high-speed manoeuvres and use it to refine the dynamic model for racing applications. The bicycle model presented in [36] is an effective representation of the vehicle dynamics, yet it is still based on several simplifying assumptions and is built using data collected in quasi-static conditions. For example, it does not consider longitudinal weight shift during acceleration and braking. This is visible in the right side plot in Figure 5.3. We notice that the tire model in [36] (dashed gray line) is accurate for small values of longitudinal acceleration, yet for high negative values (during harsh braking) the measured lateral force is higher than expected since there is more normal force acting on the front tire. Likewise, the measured lateral force is lower than the predicted value during high positive acceleration values.

To capture these and other unmodelled dynamics effects, we use a Stochastic Variational Gaussian Process (SVGP) to model the residual dynamics. We do this by training the SVGP to predict the value of $\mathbf{e}$ in Eq. (5.19). The theory behind SVGP modelling is a fascinating topic, yet we consider it to be out scope, referring the reader to [128] for an introduction, and to the GPytorch library [129] for the practical implementation details. For the purposes of the current paper, we can consider an SVGP model as a function that takes the same inputs as the dynamic bicycle model, and outputs a corrective term $\mathbf{e}$ expressed as a Gaussian probability distribution. The dynamic model of the vehicle is thus:

$$\ddot{x} = \ddot{\tilde{x}}\left(v_x, v_y, \omega, \hat{\tau}, \hat{\delta}\right) + \mathrm{e}_x^{\mathrm{GP}}\left(v_x, v_y, \omega, \hat{\tau}, \hat{\delta}\right) \tag{5.20}$$

$$\ddot{y} = \ddot{\tilde{y}}\left(v_x, v_y, \omega, \hat{\tau}, \hat{\delta}\right) + \mathrm{e}_y^{\mathrm{GP}}\left(v_x, v_y, \omega, \hat{\tau}, \hat{\delta}\right) \tag{5.21}$$

$$\dot{\omega} = \dot{\tilde{\omega}}\left(v_x, v_y, \omega, \hat{\tau}, \hat{\delta}\right) + \mathrm{e}_\omega^{\mathrm{GP}}\left(v_x, v_y, \omega, \hat{\tau}, \hat{\delta}\right), \tag{5.22}$$

where e_x^{GP}, e_y^{GP}, e_ω^{GP} are 3 separate multi-input single-output SVGP models. Notice that the

output of each GP is a Gaussian distribution:

$$e^{GP} = \mathcal{N}(\mu, \sigma). \tag{5.23}$$

5.4.2 Drone dynamics

To model the drone dynamics we rely on the unified thrust model. The forces acting on the quadrotor are defined as:

$$\tilde{T} = mg + T, \tag{5.24}$$

$$F_x = \tilde{T}\,(\sin\psi \sin\phi + \cos\phi \sin\theta \cos\psi) - k_{\text{drag}} v_x, \tag{5.25}$$

$$F_y = \tilde{T}\,(-\sin\phi \cos\psi + \cos\phi \sin\psi \sin\theta) - k_{\text{drag}} v_y, \tag{5.26}$$

$$F_z = \tilde{T}\,(\cos\phi \cos\theta) - mg - k_{\text{drag}} v_z, \tag{5.27}$$

where m is the mass, g is the acceleration due to gravity, the input T is the combined thrust of the propellers, ϕ, θ and ψ are the roll pitch and yaw, respectively, and k_{drag} is the air drag linear coefficient. The full dynamic model can thus be written as:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \\ \dot{v}_x \\ \dot{v}_y \\ \dot{v}_z \end{bmatrix} = \begin{bmatrix} \omega_\phi \\ \omega_\theta \\ \omega_\psi \\ F_x/m \\ F_y/m \\ F_z/m \end{bmatrix}, \tag{5.28}$$

where ω_ϕ, ω_θ and ω_ψ are control inputs. Note that the full rotational dynamics are described as:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin(\phi)\tan(\theta) & \cos(\phi)\tan(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \dfrac{\sin(\phi)}{\cos(\theta)} & \dfrac{\cos(\phi)}{\cos(\theta)} \end{bmatrix} \begin{bmatrix} \omega_\phi \\ \omega_\theta \\ \omega_\psi \end{bmatrix}, \tag{5.29}$$

yet in Eqs. (5.28), we assume that the attitude angles remain small, and simplify the rotational dynamics as

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \approx I \begin{bmatrix} \omega_\phi \\ \omega_\theta \\ \omega_\psi \end{bmatrix}. \tag{5.30}$$

Furthermore, when testing on the real system, the total thrust T and the attitude angle rates are tracked by a low-level controller that converts them into actuator inputs, relying on the full rotational dynamics. Since the low-level controllers reliably track the open-loop trajectories from the MPCC, we consider a GP residual dynamics model to be superfluous.

5.5 Experimental validation

In this section, we compare our rCA-MPCC to the MPCC++ scheme in [33]. We articulate our study across different MPC implementation choices, providing the reader with a quantitative measure of how different MPC design choices impact the final racing performance.

We first provide a general overview of our experimental setup, pertaining to both ground and aerial racing, and then go into further details in the respective subsections.

5.5.1 Experimental setup.

As detailed in Section 5.2, the two methods share the same underlying strategy, maximizing the progress along a provided reference path $\boldsymbol{x}^{\text{ref}}(s)$. We slightly modify the formulation presented in [33] in order to ensure a fair comparison with our method. Concerning Eq.(5.9a), for the MPCC++ formulation we augment the control inputs with the virtual input $\dot{s}_k$, add a high penalty cost term on ϵ_{lag} in the stage-wise cost, and remove the curvature-related term $+q_\kappa^{\text{t}}\alpha_\kappa^2$ from Eq. (5.10). Note that the two methods differ mainly in the practical implementation of Eq. (5.3), yet the two formulations are equivalent from a conceptual point of view. The unique advantage of rCA-MPCC over the state of the art MPCC++ is in the terminal cost term introduced in Eq. (5.10). This term pushes the endpoint of the open-loop trajectory towards the inside of the racetrack curve. The underlying assumption is that by taking a second-order approximation of the path evolution, i.e. under the assumption of constant curvature, traveling on the inner side of the curve ensures greater overall progress.

Automatic controller tuning. To ensure a fair comparison among the two methods we automatically tune both controllers for each different tesing scenario according to the follwing procedure. We fix the stage-wise cost function weights Q_u and q_c to the same values for both formulations, and for the MPCC++ controller we fix q_{lag} to a high value. We then automatically tune the weights in the terminal cost function J_N for each testing condition. To perform the automatic gain-tuning procedure, we rely on Optuna [130], a hyperparameter optimization framework. We integrate the Optuna parameter search in our ROS-based simulator, allowing the parameter optimizer to minimize the lap time by directly changing the weights in J_N via a GUI. For each tuning study, we allow 100 trials, where the objective to be minimized is the average time over 3 laps, plus a penalty for lane boundary violation.

Lab setup. To perform the tests, we rely on off-board computing to solve the MPC problem and send the control inputs to the robots over a WiFi network, since the on-board computational capacity of the latter is insufficient for high speed operation. We additionally rely on an external motion capture system to estimate the vehicles' state.

5.5.2 Car racing

In this subsection, we compare our rCA-MPCC against MPCC++ in 2D vehicle racing applications by deploying them on the small-scale car-like robot presented in [36], as described in section 5.4.1. For our tests we design a race track that presents sharp curves that match the minimum turning radius of the vehicle, providing challenging testing conditions. Concerning the MPCC++ algorithm, we adapt the formulation presented in [33] from drone racing to vehicle racing as described in Section 5.2. For the ground vehicle tests, we fix the prediction horizon length to a large value of $1.5s$ discretized in $0.1s$ steps, this will minimize the effects of the terminal cost J_N, investigated later in detail in Section 5.5.3, and allow us to highlight the effects of the other design choices.

Hierarchical MPC architecture. We begin our study by adopting a hierarchical MPC architecture, where the high-level MPCC motion planner (based on either rCA-MPCC or

MPCC++) relies on simplified vehicle dynamics to produce a time-scheduled reference path, that is then tracked by traditional low-level MPC that uses the full vehicle dynamics. Owing to the simplified dynamics used in the high-level planner, this hierarchical MPC architecture is only suitable for low-speed applications, yet it allows us to focus specifically on the path-planning capabilities of the two algorithms.

The high-level planner solves Eq.(5.9) using the following simplified vehicle dynamics:

$$\dot{x} = V\cos(\psi), \tag{5.31}$$

$$\dot{y} = V\sin(\psi), \tag{5.32}$$

$$\dot{\psi} = u, \tag{5.33}$$

where the vehicle heading ψ is controlled directly by the only control input u, and V is a tunable fixed longitudinal velocity. We set $V = 2.5\,\mathrm{m/s}$, which was determined empirically as the largest value that ensures stable closed-loop operation with this hierarchical MPC structure.

We first conduct a preliminary lap-time study. In this experiment, both CAMPCC and MPCC++ were tuned offline relying on Optuna [130] by optimizing their respective cost weights in simulation, using the ROS physics-based dynamic bicycle model simulator provided in [36], with and without an additional Gaussian Process residual model. Note that we now focus on the effect of the gain tuning alone, therefore the controller does not incorporate the GP correction term at runtime, the effect of this choice will be later shown in Figure 5.7. The subsequent laboratory evaluation considered both algorithms with and without compensating for the communication delay introduced by our off-board computing setup. For each of the resulting eight test configurations, the vehicle completed three batches of three laps, yielding nine laps per case. The communication delay compensation strategy consists in measuring the average communication delay Δt_{com}, measured to be $\Delta t_{\mathrm{com}} \approx 0.05\mathrm{s}$ in our setup, and correcting the initial state provided to the MPC as:

$$\tilde{x}_0 = x_0 + V_0\cos(\psi_0)\,\Delta t_{\mathrm{com}}, \tag{5.34}$$

$$\tilde{y}_0 = y_0 + V_0\sin(\psi_0)\,\Delta t_{\mathrm{com}}, \tag{5.35}$$

$$\tilde{\psi}_0 = \psi_0 + \dot{\psi}_0\,\Delta t_{\mathrm{com}}, \tag{5.36}$$

where x_0 and y_0 denote the measured vehicle position at the time the state is received by the controller, and ψ_0 is the measured vehicle heading. The quantities V_0 and $\dot{\psi}_0$ represent the measured forward speed and yaw rate at time $t = 0$, respectively. The terms $\tilde{x}_0$, $\tilde{y}_0$, and $\tilde{\psi}_0$ are the delay-compensated initial states obtained by forward-integrating the measured state over this delay interval.

Table 5.1 reports the mean lap times corresponding to the different testing conditions. The results show that incorporating either the GP residual model during weight tuning or the communication delay compensation during testing, yields a similar performance improvement over the baseline. This highlights that practical implementation details can be just as critical as sophisticated simulation tools when deploying controllers on real robotic systems. Moreover, rCA-MPCC consistently outperforms MPCC++ by a small margin across all configurations.

We subsequently provide additional statistical relevance to the comparison between our rCA-MPCC and MPCC++ by selecting the best testing conditions, i.e. weight tuning

	Weight tuning: dynamic bicycle	Weight tuning: dynamic bicycle + GP
Delay compensation: yes	rCA-MPCC:7.77 MPCC++:7.81	**rCA-MPCC:7.67** MPCC++:7.74
Delay compensation: no	rCA-MPCC:7.79 MPCC++:7.91	rCA-MPCC:7.75 MPCC++:7.90

Table 5.1: Preliminary mean lap time study using the hierarchical MPC scheme. The weights for both the rCA-MPCC and MPCC++ algorithm were optimized using a simulator with a physics-based dynamic bicycle model with and without a corrective Gaussian Process residual dynamics model. During the lab tests, both algorithms were tested with and without communication delay compensation. For each of the 8 cases we recorded 9 laps in 3 batches of 3 laps each.

	rCA-MPCC	MPCC++
mean [s]	7.629	7.636
SD [s]	0.079	0.272
P95 [s]	7.753	8.183
P99 [s]	7.800	8.231

Table 5.2: Lap time statistics from Figure 5.5.

with GP residual model and delay compensation during lab deployment, and performing an additional 40 laps for each controller, collected in batches of 5 laps. The statistical distributions of the measured lap times are shown in Figure 5.5, table 5.2 reports the quantitative overview of the same data, Figure 5.6a- 5.6b shows the corresponding closed-loop trajectories and table 5.3 shows the computation time and convergence ratio (CVG).

The results show that on average the two methods have similar mean lap time. However, the rCA-MPCC is more consistent as is evident by looking at the standard deviation of the lap time distribution. Moreover, rCA-MPCC has more consistent closed-loop trajectories compared to MPCC++ and does not violate lane boundary constraints, unlike the latter, see Figure 5.6b. Concerning the computation time shown in table 5.3, rCA-MPCC shows slightly higher values, yet remains within operational limits, as does the solver convergence rate (CVG).

Single layer MPC. To push the vehicle closer to its handling limits, we implement a single-layer MPC architecture that can directly make use of the full vehicle dynamic model presented in section 5.4.1. We compare the two algorithms with and without the additional GP-based residual dynamics in the MPC dynamic constraint, showing the impact of the dynamic model accuracy on the closed-loop performance. Note that for all cases we still tune the MPC parameters in simulation with the full physics-based dynamic model provided in [36] with the additional GP-based term in Eq. (5.19) and enable communication delay compensation at runtime. We further mention that in our laboratory environment, the floor surface provides substantial grip, causing the vehicle to reach its maximum allowable roll moment before the tires approach lateral force saturation, in other words, the vehicle is more likely to roll over than to drift when cornering at high speed. To prevent this behaviour, we add a maximum centrifugal force constraint into the MPC formulation.

For each test case we perform 40 laps collected in batches of 5 laps. Figure 5.7 shows

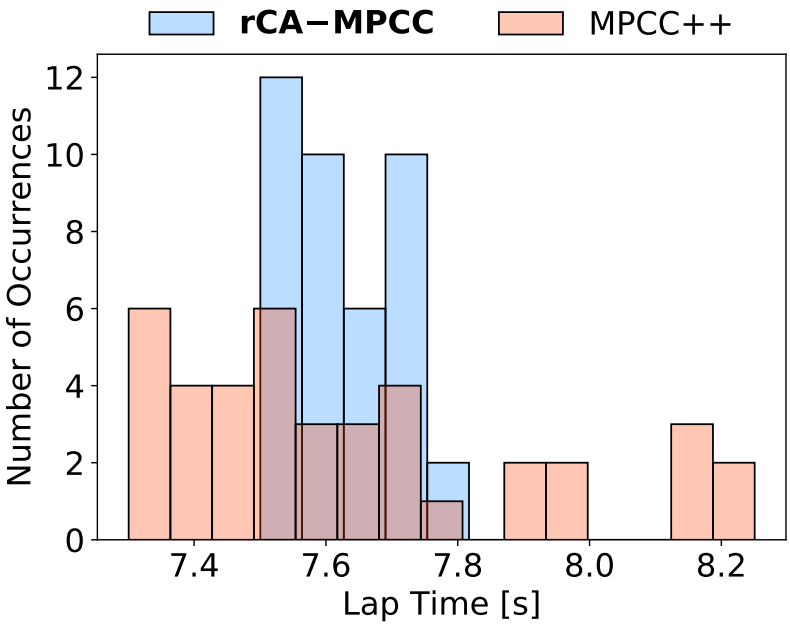
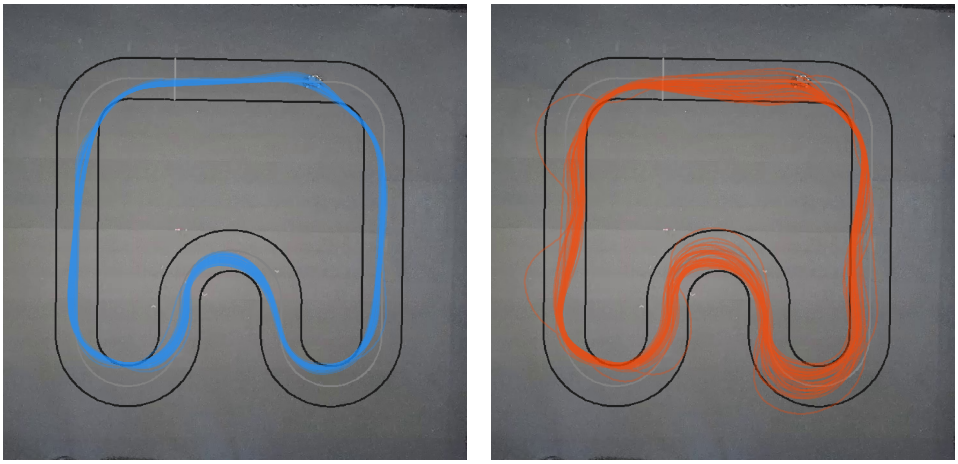


Figure 5.5: Lap time distribution comparison among the different control algorithms using a hierarchical MPC scheme. The weights were optimized with a simulator using the dynamic bicycle + GP model and we used communication delay compensation during lab deployment, as in the best case conditions in table 5.1. The high-level path planner uses either rCA-MPCC or MPCC++ and minimizes lap time while moving at a fixed speed of 2.5 m/s. Subsequently, a low level MPC uses a physics-based dynamic bicycle model to track the reference trajectory. Each experiment consists of 40 laps recorded in 5 lap batches.

the statistical distributions for the two controllers with and without the GP residual dynamics, Table 5.4 reports the quantitative overview of the same data, Figure 5.8 shows the corresponding closed-loop trajectories, and Table 5.5 shows the computation time and solver convergence rate (CVG).

Comparing the two algorithms, we see that rCA-MPCC shows a significant improvement over MPCC++ in terms of lap time. Since the prediction horizon has been chosen purposefully long, the difference in performance is not attributable to the terminal cost term in Eq. (5.10), but rather to the benefits introduced by Eq. (5.15) that conveys increased solver consistency, as can be seen in the open-loop trajectories shown in the resulting close-loop trajectories in Figure 5.8. Notice however that this reference path formulation is only viable for 2D racing. Regarding the choice of dynamic model within the MPC, the results in Figure 5.7 indicate that including the data-driven GP residual model into the MPC dynamics slightly decreases overall closed-loop performance, despite its higher modelling fidelity. This is because the advantages of the corrective GP term are outweighed by the additional computational burden introduced during online optimization, as shown in Table 5.5. It is also important to note that the GP-based residual dynamics were already accounted for during offline parameter tuning in both cases. As a result, the accuracy gained by considering the GP model at runtime does not justify the increased computational cost, making a lighter dynamic model the more favourable choice.



(a) CAMPCC (b) MPCC++

Figure 5.6: Vehicle trajectories comparing different control algorithms in a hierarchical MPC scheme: Curvature-Aware MPCC (a) and MPCC++ (b). The trajectories are from the same dataset used in figure 5.5, featuring 40 laps for each experiment.

	rCA-MPCC	MPCC++
mean [s]	0.033	0.030
SD [s]	0.006	0.008
P95 [s]	0.042	0.042
P99 [s]	0.050	0.056
CVG [%]	99.987	99.974

Table 5.3: Computation time statistics from Figure 5.5. CVG indicates the solver convergence rate.

	rCA-MPCC	**rCA-MPCC**(GP)	MPCC++	MPCC++(GP)
mean [s]	6.603	6.640	7.190	7.226
SD [s]	0.185	0.165	0.142	0.170
P95 [s]	6.927	6.943	7.379	7.558
P99 [s]	7.030	7.029	7.551	7.631

Table 5.4: Lap time statistics of data in Figure 5.7.

	rCA-MPCC	**rCA-MPCC**(GP)	MPCC++	MPCC++(GP)
mean [s]	0.025	0.029	0.016	0.020
SD	0.006	0.008	0.003	0.006
P95 [s]	0.032	0.038	0.021	0.031
P99 [s]	0.039	0.056	0.023	0.040
CVG [%]	99.803	99.819	99.993	99.972

Table 5.5: Computation time statistics and solver convergence rate (CVG) of data in Figure 5.7.

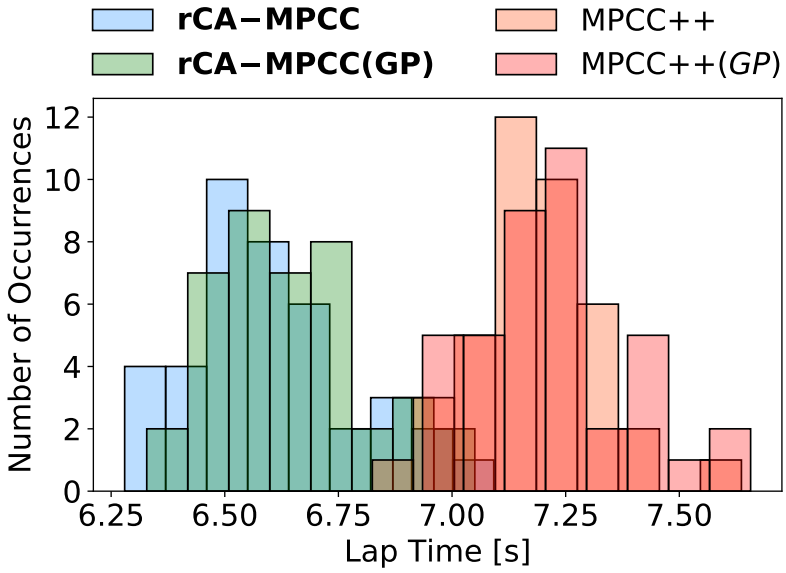
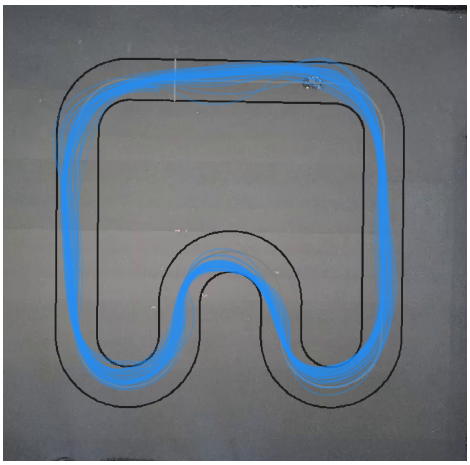
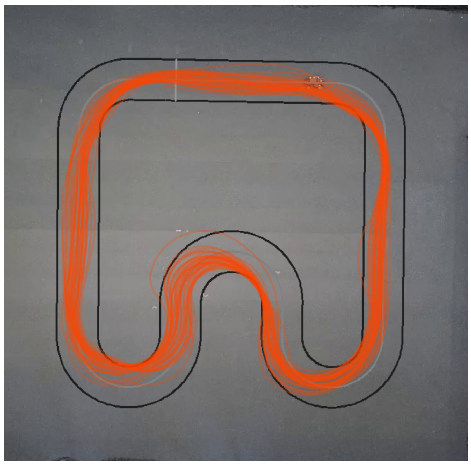


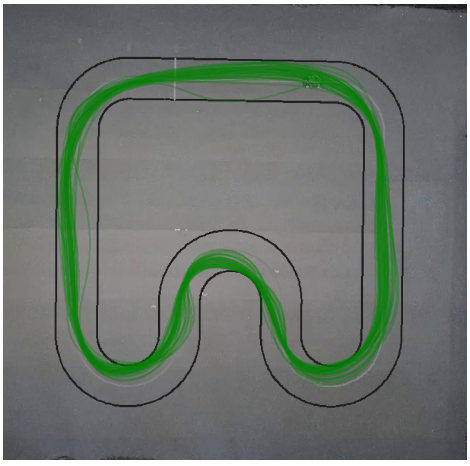
Figure 5.7: Lap time distribution comparison among the different control algorithms using a single layer MPC scheme: rCA-MPCC and MPCC++. Each algorithm was evaluated using a physics-based dynamic bicycle model, with and without an additional data-driven Gaussian Process residual dynamics model (indicated with (*GP*) in the caption). Each experiment consists of 40 laps recorded in 5 lap batches.



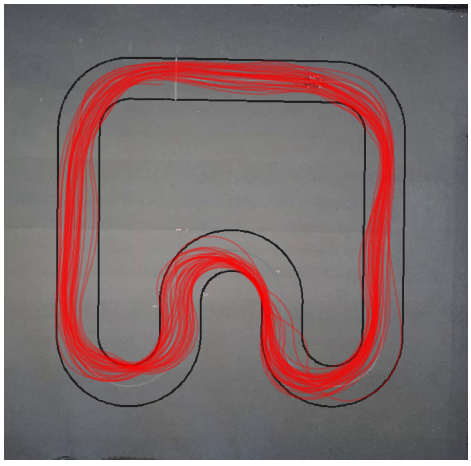
(a) CAMPCC



(b) MPCC++



(c) CAMPCC (*GP*)



(d) MPCC++ (*GP*)

Figure 5.8: Vehicle trajectories comparing different control algorithms: rCA-MPCC and MPCC++. Both algorithms were tested using a physics-based dynamic bicycle model (a)-(b), and with an additional Gaussian Process model to capture the residual dynamic (c)-(d). The trajectories are from the same dataset used in Figure 5.7, featuring 40 laps for each experiment.

5.5.3 Drone racing

In this subsection we demonstrate the effectiveness of the terminal cost in our rCA-MPCC, investigating its effects on 3D drone racing performance according to different horizon lengths.

Simulation on a circular racetack. As an illustrative example, we show the results for a circular racetrack in simulation. We set up the simulator based on integrating the dynamic model in Eq. (5.28) using an explicit Runge-Kutta 4^{th} order scheme. The track is a circle of radius $r = 3$m in the horizontal plane, placed at a height of $z = 1$m from the ground. Figure 5.9 shows the improvement in lap time as a function of the MPC prediction horizon length, while Figure 5.10 shows the difference between the open-loop trajectories of the two methods and the resulting closed-loop trajectories. As can be expected, since the main difference in the two methods is in the terminal cost, rCA-MPCC outperforms MPCC++ for shorter MPC prediction horizons, while the performance is similar above a certain prediction length, with both methods converging to the infinite-horizon optimal solution yielded by solving Eq. (5.7).

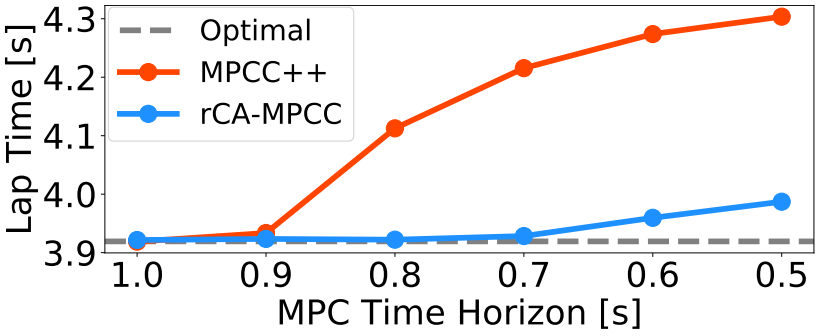


Figure 5.9: Comparison between rCA-MPCC and MPCC++ [33] on a simple circular racetrack as a function of the prediction horizon length. For high values of horizon time the performance is similar, while for short time horizon lengths the terminal cost has a more significant effect, and rCA-MPCC outperforms [33].

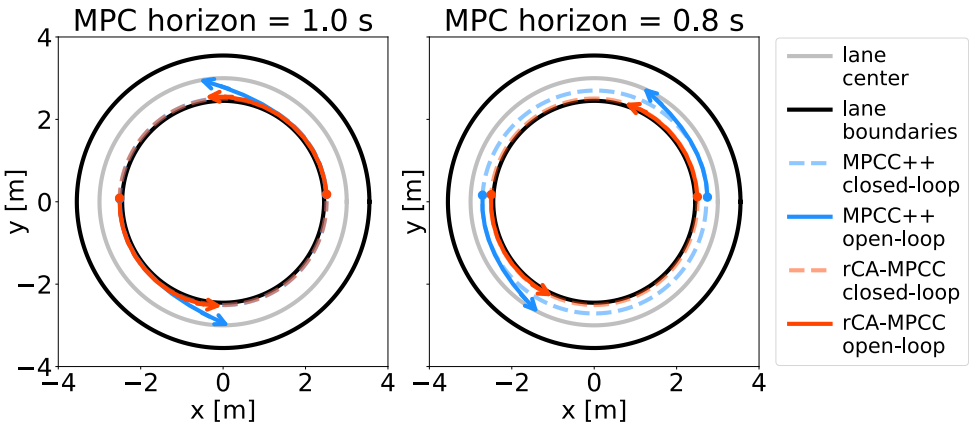


Figure 5.10: Comparison between rCA-MPCC and MPCC++ [33] open-loop and closed-loop trajectories on a simple circular racetrack for different MPC time horizons. For long prediction horizons, the two methods show similar trajectories, as the terminal cost has little influence on the closed-loop behaviour. Conversely, for short horizons the terminal cost becomes significant, and rCA-MPCC shows a tighter closed-loop trajectory due to its capability to anticipate path curvature beyond the prediction horizon.

5.6 Conclusions

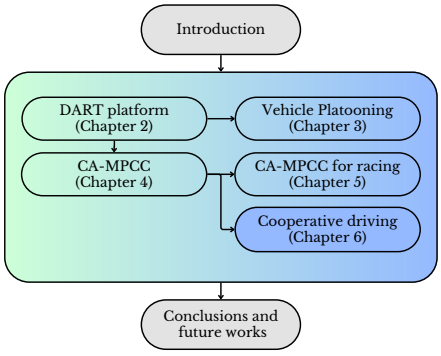
In this work we present rCA-MPCC, a Curvature-Aware MPC scheme for ground and aerial robot racing. By introducing a curvature-informed terminal cost and an efficient reference-path representation, our method achieves more consistent closed-loop behaviour and improved lap-time performance compared to the state of the art, validating our claim in an extensive experimental study. Our study also shows the impact of practical implementation details such as controller architecture (hierarchical vs single-layer MPC), dynamic model fidelity, delay compensation at runtime, and actuator modelling on real-world performance, providing a valuable guide for MPC deployment on real robots. Overall, rCA-MPCC provides a reliable MPC formulation that reduces reliance on prior knowledge of the racetrack, helping to narrow the gap between racing-oriented control methods and time-critical manoeuvres in more general navigation scenarios.

6

Cooperative driving applications

Overview

In this chapter we present the following work on multi-agent persistent monitoring in which the DART platform, introduced in Chapter 2, and adaptations of the MPC formulation from Chapter 4 have been applied to implement cooperative mobile robotics strategies in practice[1].



This case study illustrates how the DART platform and MPC-based methods can support the practical deployment of cooperative robotics applications, bridging the gap between theoretical development and experimental validation.

Method overview. This chapter proposes a distributed strategy for multi-robot persistent monitoring and target detection in rectangular, obstacle-free environments. The central idea is to combine *time-inverted Kuramoto dynamics* with *Lissajous curves* to coordinate the robots. The Kuramoto-inspired dynamics provide resilient synchronization properties, while the Lissajous curves yield smooth trajectories that guarantee complete coverage and finite-time target detection. The approach is fully distributed, requiring each robot to interact only with its neighbors in a ring communication topology. For the reader's

[1]This chapter is adapted from the full paper *Time-inverted Kuramoto model meets Lissajous curves: Multi-robot persistent monitoring and target detection* by M. Boldrer, L. Lyons, L. Palopoli, D. Fontanelli, L. Ferranti. L. Lyons was responsible for adapting the algorithms to run on the DART platform, integrating the necessary hardware and software components, ensuring that the cooperative strategies could be validated in real-world conditions, and wrote the relevant sections (Sections 6.6 and 6.7 reported in this thesis). M. Boldrer is the main contributor, being responsible for developing the method, validating in simulation and writing the manuscript. L. Palopoli, D. Fontanelli and L. Ferranti supervised the research.

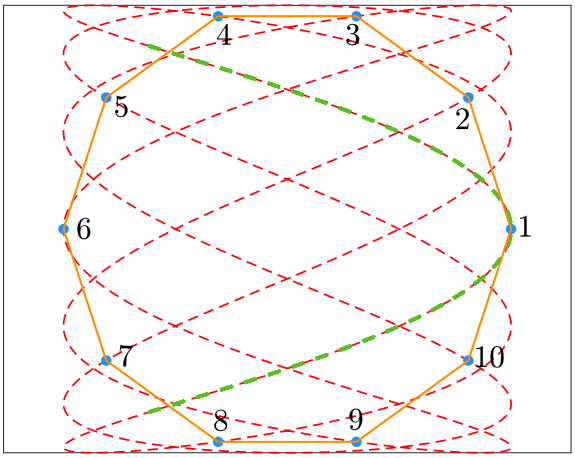


Figure 6.1: Final equilibrium configuration for the system (6.1) with $N = 10$. The Lissajous curve is represented by a red dotted line. The agents' positions on the curve are depicted as blue dots and the admissible perturbation that maintain the same equilibrium configuration is depicted as a green dashed line.

convenience we show the equations for the time-inverted Kuramoto dynamics (6.1), and an example Lissajous curve in Figure 6.1, both taken from [131].

$$\dot{\theta}_i = \omega - \sum_{j \in \mathcal{R}_i} \sin\left(\theta_j - \theta_i\right), \quad \forall i = 1, \dots, N, \tag{6.1}$$

where ω identifies the natural frequency, that is, the desired velocity of the robots at the equilibrium.

The theoretical contributions include formal conditions for collision avoidance, complete coverage, and guaranteed detection, even in the presence of disturbances or temporary agent failures. Compared to state-of-the-art approaches, the method provides stronger resiliency properties, greater flexibility in equilibrium configurations, and no reliance on centralized computation of reference trajectories.

My contribution to this project lies in implementing and validating the framework on the DART platform presented in Chapter 2. To bridge the gap between the high-level coordination strategy and the real hardware, I developed a tailored *low-level MPC controller*, building on top of the CA-MPCC presented in chapter 4. In **Section 7**, I developed a *Curvature-Aware Model Predictive Contouring Control (CA-MPCC)* formulation adapted to this problem. Unlike standard MPC, CA-MPCC avoids time-dependent references and instead directly minimizes contouring errors relative to the desired path. The CA-MPCC lends itself particularly well to this application since it can explicitly track the path progress variable speed $\dot{\theta}$ imposed by the Kuramoto dynamics, and ,unlike traditional MPCC, is able to handle the tight bends that characterize the *Lissajous curves*.

In **Section 8**, I carried out the experimental validation on the DART robotic platform presented in chapter 2. Three DART robots and two simulated agents were coordinated on Lissajous curves using the proposed strategy. The experiments demonstrate the effectiveness of the MPCC in minimizing tracking error under actuation limits, and confirm

the resiliency properties predicted in theory. The robots maintained their equilibrium configurations despite disturbances and simulated failures, successfully achieving persistent monitoring and finite-time target detection.

In summary, while the paper as a whole introduces a novel distributed framework for resilient multi-robot coordination, my specific contributions are the design of the **low-level MPC-based controller** (Section 7) and the execution and analysis of the **experimental validation** (Section 8). These parts ensure that the proposed theory is not only mathematically sound but also implementable and effective on real robotic systems. We only include sections 7 and 8 in this thesis, and refer the interested reader to [132] for the full paper.

6.6 Low level MPCC design

The time-inverted Kuramoto model considers robots as points moving along the Lissajous curve, that is, it considers a one dimensional problem. A lower control routine is needed to enforce path following and velocity tracking. We rely on Model Predictive Control (MPC). MPC allows our method to consider the full dynamic model of the robots. Furthermore, thanks to its constraint-handling abilities, the MPC controller is able to account for actuator saturation. This is particularly valuable since the turns featured in the Lissajous path can exceed the steering angle limits of the robots. For this work, we chose a particular formulation of MPC referred to as Model Predictive *Contouring* Control (MPCC) [133]. Compared to traditional MPC for tracking, MPCC does not require an explicitly time-dependent reference trajectory. These features make MPCC particularly well-suited to act as a low level controller for our purposes. In [133] however, the controller aims at tracking a given longitudinal speed, while in our case we require to track a certain $\dot{\theta}$. We have therefore modified the relative term in the cost function. The optimization problem solved by the MPCC, for the i-th agent, is the following:

$$\min_{\delta_i, \tau_i} \sum_{k=0}^{M} J_{\dot{\theta}_i}^k + q_1 {\varepsilon_{i,\text{lag}}^k}^2 + q_2 {\varepsilon_{i,\text{lat}}^k}^2 + q_3 \delta_i^{k2} + q_4 \tau_i^{k2} \tag{6.2a}$$

$$\text{s.t. } X_i^k = f(X_i^{k-1}, \delta_i^{k-1}, \tau_i^{k-1}), \text{ for } k = 1, ..., M \tag{6.2b}$$

$$\delta_{\min} \leq \delta_i \leq \delta_{\max}, \quad \tau_{\min} \leq \tau_i \leq \tau_{\max}, \tag{6.2c}$$

where k is the time index, M is the number of stages in the optimization problem, $J_{\dot{\theta}_i}$ is the $\dot{\theta}$ tracking term (6.6) that substitutes the velocity tracking term in [133], q_1, q_2, q_3, q_4 are weights normalized with respect to the weight of $J_{\dot{\theta}_i}$, $\varepsilon_{i,\text{lag}}$ is the so called lag error and is a specific term of the MPCC formulation (refer to [133] for more details), $\varepsilon_{i,\text{lat}}$ is the lateral distance to the path, δ_i and τ_i are respectively the steering angle and the throttle, $\delta_{\min}$, $\delta_{\max}$, $\tau_{\min}$, $\tau_{\max}$ are the minimum and maximum values of the respective inputs. We decided not to include explicit collision avoidance constraints, since we want to focus on the results provided by the time-inverted Kuramoto model. X_i is the predicted state of the i–th robot and f represents the robot dynamics. The state X_i is defined as:

$$X_i = \begin{bmatrix} x_i & y_i & \eta_i & v_i & s_i \end{bmatrix}^\top,$$

where x_i, y_i, η_i, v_i, s_i are the position of the rear axle, orientation, longitudinal velocity, and the natural parameter of the closest point to the i–th robot on the reference path, respectively. The dynamics of each robot are considered as a kinematic bicycle model [134]. The longitudinal acceleration is $\dot{v}_i = -cv_i + \alpha\tau_i - \beta$, where c is the damping coefficient, and α and β are the motor force coefficients (we evaluated them experimentally through step response tests for the target application). The dynamic constraint (6.2b) also includes the dynamics of the natural parameter s_i, necessary to evaluate $J_{\dot{\theta}_i}(k)$. In summary, the dynamic constraint (6.2b) (in continuous time) is given by:

$$\dot{X}_i = \begin{bmatrix} v_i \cos\eta_i \\ v_i \sin\eta_i \\ \frac{v_i \tan(\delta_i)}{l} \\ -cv_i + \alpha\tau_i - \beta \\ v_i \end{bmatrix}, \tag{6.3}$$

where l is the length between the front and rear axle of each robot.

Concerning the term $J_{\dot{\theta}_i}$, the time-inverted Kuramoto model (6.1) provides a desired $\dot{\theta}_i^D$, that can be tracked by the MPCC. This, however, is expressed as a feedback law depending on the neighboring agents' positions. Thus, to implement this policy directly, the MPCC should be designed in a fully distributed way. This would imply a large amount of network traffic and would ultimately hinder the real-time applicability of the overall control strategy.

For this reason we opted to keep the desired $\dot{\theta}_i^D$ constant during the whole prediction horizon. In this way the θ_j values need to be exchanged only once per sampling time. Our choice is further legitimized by noticing that at equilibrium the $\dot{\theta}_i^D$ remains constant.

The cost term $J_{\dot{\theta}_i}$ is defined as

$$J_{\dot{\theta}_i} = (\dot{\theta}_i^D - \dot{\theta}_i)^2. \tag{6.4}$$

The state of the vehicle provides s and $\dot{s}$ (since $\dot{s} = v$ as described in (6.3)), thus we must rewrite (6.4) as a function of these quantities. The expression of the arc-length s for a certain value of γ is as follows:

$$s(\gamma(t)) = \int_{\gamma_0}^{\gamma(t)} \|r'(\tilde{\gamma})\|_2 d\tilde{\gamma}, \tag{6.5}$$

where we set $\gamma_0 = 0$ as defined in [131], $r'(\tilde{\gamma})$ is the partial derivative of the Lissajous curve with respect to $\tilde{\gamma}$, and the $\|\cdot\|_2$ operator is the Euclidean norm. By computing the time derivative of (6.5) we obtain $\dot{s} = \dot{\gamma}\|r'(\gamma)\|_2$. Note that the MPCC formulation can only predict future values of s, thus the term $\|r'(\gamma)\|_2$ needs to be expressed as a function of the latter, that is, an expression of $\gamma(s)$ is needed. This however is not trivial since it requires the inversion of (6.5) that is not possible to formalize analytically. For this reason we have evaluated $\gamma(s)$ numerically, and used a suitable polynomial approximation of it during run-time.

The velocity tracking term in the MPCC cost function (6.4) can be re-written as:

$$J_{\dot{\theta}_i}(s, \dot{s}) = (\dot{\theta}_i^D - \dot{s}/\|r'(\gamma(s))\|_2)^2. \tag{6.6}$$

6.7 Experimental results

Due to hardware availability reasons, we are going to consider 3 scaled-down car-like robots and 2 simulated agents. To control the robots a dedicated ROS network has been set up and we rely on an OptiTrack motion capture system to receive the positions of the robots in real-time. The ring topology $\mathcal{R}$ in [131] adopted in the experiments reads as follows, $\mathcal{R}_1 = \{2, 4\}, \mathcal{R}_2 = \{1, 5\}, \mathcal{R}_3 = \{4, 5\}, \mathcal{R}_4 = \{1, 3\}, \mathcal{R}_5 = \{2, 3\}$. The radius of encumbrance of each vehicle is $r_i = 0.125$ (m). We consider the following Lissajous curve for the experiments:

$$\mathcal{L} : \begin{cases} x(\gamma) &= 1.8\cos(3\gamma) \\ y(\gamma) &= 0.8\sin(2\gamma). \end{cases} \tag{6.7}$$

We satisfy collision avoidance at the equilibrium, that is, $r_i < r_{i,\max}$. At run-time each agent computes its desired $\dot{\theta}_i^D$ according to (6.1). Note that to perform this step the i–th agent requires the θ_j values, with $j \in \mathcal{R}_i$. Once $\dot{\theta}_i^D$ is computed the value is passed to the MPCC, which solves the optimization problem (6.2), (6.3) and computes the control inputs τ_i, δ_i. The video of the experiments can be found at [135].

The parameters used for the experiments are the following: $q_1 = 0.01$, $q_2 = 0.025$, $q_3 = 0.01$, $q_4 = 0.01$, $\delta_{\min} = -\frac{\pi}{9}$ (rad), $\delta_{\max} = \frac{\pi}{9}$ (rad), $\tau_{\min} = 0.1$, $\tau_{\max} = 0.25$, $M = 15$ in (6.2), $c = 0.94$, $\alpha = 36.8$, $\beta = 0.94$, $l = 0.175$ (m) in (6.3), $\omega = 0.3$ (rad/s).

The first experiment shows the importance of the feedback term in (6.1). We run the experiment neglecting it and imposing the dynamics $\dot{\theta}_i = \omega, \forall i = 1, \dots, N$. The obtained results can be found at [135]. The initial configuration of the system is picked around an equilibrium point. After few seconds the system diverges from the equilibrium due to modelling errors and disturbances, until a physical collision occurs.

The second experiment shows the results obtained by imposing the time-inverted Kuramoto dynamics (6.1). The initial configuration of the system (Figure 6.2-(a)) is set around an equilibrium point. Thanks to the feedback action, the system is able to remain in that equilibrium configuration, see Figures 6.2-(b,c,d). At the time instant of (d), to experimentally show the resiliency of our algorithm, we simulate a temporary failure. In particular, we modify the dynamics of the simulated agents in $\dot{\theta}_a = 0$, where $a = \{1, 5\}$. The networked system converges to another equilibrium configuration (Figure 6.2-(f)), where all the agents are stationary. After few seconds we reset $\dot{\theta}_a$ as in (6.1) and the system, after a brief transient, comes back to its previous equilibrium configuration (Figures 6.2-(f,g,h)). Figure 6.3 reports the quantitative data for the same experiment. The obtained results are in strong agreement with the results obtained in the simulations, see [131].

In the last experiment we impose the time-inverted Kuramoto dynamics (6.1). We run the experiment for a sufficient amount of time in order to investigate the effects of disturbances and model mismatch on target detection, collision avoidance and complete coverage. In Figure 6.4 we focus on the agent 3; we report the evolution in time of the Euclidean distance between agents 3 and 4 (d_{34}), and between agents 3 and 5 (d_{35}), and the lateral error from the Lissajous path ($d_{3l} \equiv \varepsilon_{3,\text{lat}}$). The maximum value for the tracking error d_{3l} is the increase in the sensing range, sufficient to guarantee complete coverage. The maximum distance between two neighboring agents ($d_{34}(t^\star)$) is twice the sensing radius sufficient to ensure target detection. In this experiment the minimum sensing radius to ensure target detection in the ideal case has to be multiplied by a factor $\eta_1 = 1.2429$, in

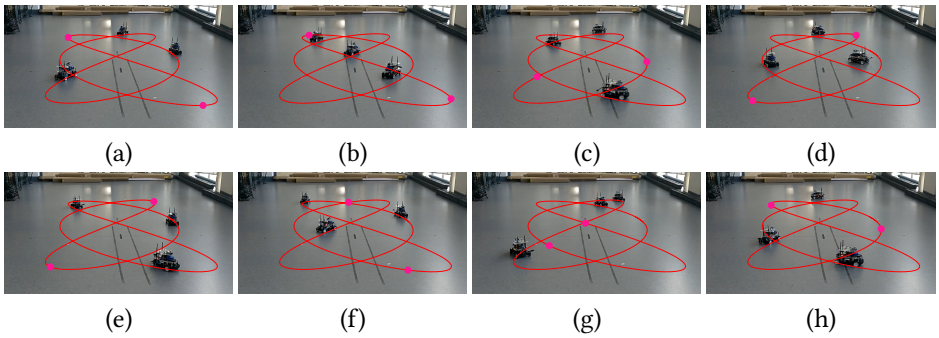


Figure 6.2: Experimental results with $N = 5$ robots (3 physical and 2 simulated). The robots are constrained to move on the Lissajous path described in (6.7). Each robot follows the time-inverted Kuramoto dynamics (6.1). The system starts at the equilibrium configuration and preserves it (a)-(d). At the time instant associated with (d) the agents 1 and 5 suddenly change their dynamics in $\dot{\theta}_{1,5} = 0$. The system reaches an alternative equilibrium configuration (e). After few seconds agents 1 and 5 resume their original dynamic (6.1) and the system comes back to the previous equilibrium configuration.

order to compensate the effects of the lateral error d_{3l}. At the same time instant $t^\star = 20.8$ (s) the large amount of lateral error d_{3l} leads to a collision between agent 3 and the simulated agent 5, since $d_{35}(t^\star) = 0.192$ (m).

6

Regarding the risk of collision an effective solution is to include explicit constraints in the MPCC's optimization problem (6.2). Considering the tracking error we have observed how it has negative effects on the properties of the system, yet at the price of an enlarged sensing radius and a smaller encumbrance radius the system is able to recover the desired features. Notice that a proper choice of the Lissajous path can also be beneficial. The tracking error d_{3l} may be induced by several factors: (i) the limited values for the steering angle $\delta_{\min}, \delta_{\max}$, with respect to the minimum radius of curvature in the Lissajous curve; (ii) the discrepancies between the model and the real dynamics of the robots; and (iii) the assumption in the MPCC formulation of keeping constant the $\dot{\theta}_i^D$ values for the whole prediction horizon. To reduce the tracking error, we should focus on these aspects.

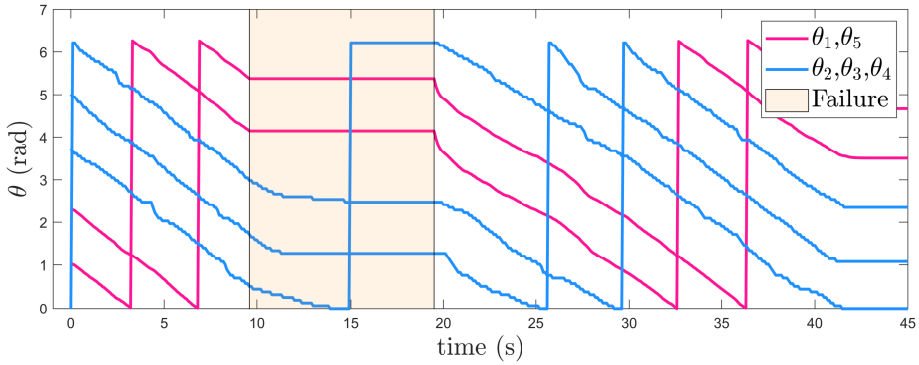


Figure 6.3: Evolution in time of the states of the agents in the experiment of Figure 6.2. In pink the states of the simulated agents, while in blue the states of the physical agents. The area in orange denotes the failure interval, where the simulated agents change their dynamics.

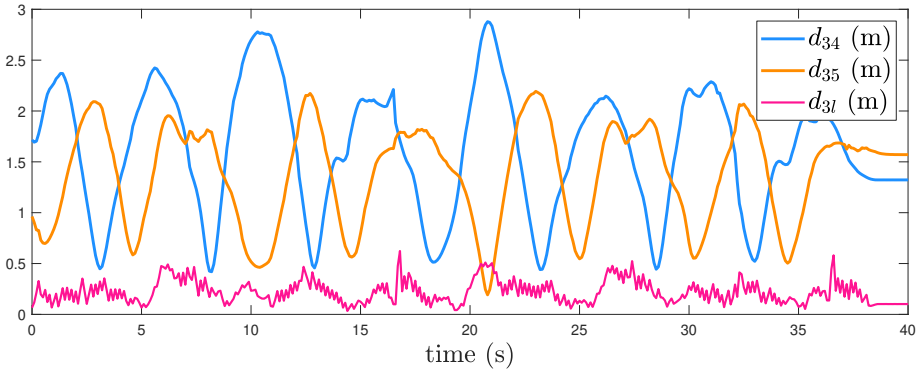


Figure 6.4: Evolution in time of the distance between robots 3 and 4 (d_{34}), the distance between robots 3 and 5 (d_{35}), and the tracking error for robot 3, i.e., the distance between the robot's position and the reference point on the Lissajous curve.

7

Conclusions and Future Work

This chapter brings together the main results of the thesis, reflecting on the contributions made across modelling, control, coordination and experimental validation. It also highlights several promising directions for continued investigation, outlining how the ideas developed here may evolve into future advances in autonomous driving research.

7.1 Conclusions

This thesis advanced the state of the art in autonomous driving by addressing three research objectives, Objective (I) to enable physically grounded, low-cost, reproducible experimentation with car-like robots that can rely on a complete identification pipeline, Objective (II) to develop cooperative driving methods that preserve safety by explicitly considering actuator limits and accounting for potentially compromised communication channels, and Objective (III) to design MPCC formulations that remain reliable on highly curved trajectories and in high-performance settings without dependence on offline solutions. These objectives were pursued through a combination of platform design, system identification, robust and attack-resilient cooperative control, and model predictive control, validated through simulation and extensive hardware experiments.

The work began in Chapter 2, fulfilling Objective (I), with the design and characterization of the Delft Autonomous-driving Robotic Testbed (DART), a 1:20 scale Ackermann-steered vehicle that preserves essential features of full-scale driving, including steering geometry, suspension effects, an electric motor, and onboard sensing and compute, while enabling fast and safe laboratory experimentation. Compared to other similar platforms such as [13, 14], our platforms has similar characteristics in terms of driving capabilities and on-board computing resources, yet is significantly cheaper and more reproducible, owing to the reduced number of custom components. Additionally, we provide a system-identification pipeline tailored to small-scale vehicles that yields a kinematic bicycle model and a dynamic bicycle model suitable for model predictive control.

Chapter 3 addressed Objective (II) by introducing a distributed, attack-resilient platooning framework that preserves collision avoidance and string stability under compromised communication. The main contributions of the method are that, for the low level switching ACC-CACC controller, it relies on reachability-based analysis that guarantees safety even if an adversarial attack remains undetected indefinitely, differently from observer-based and machine learning based approaches that aim at reconstructing the original signal. Compared to other reachability-based approaches our method also considers scenarios where the platoon leader is forced to perform an emergency brake, resulting in broader safety guarantees. At the coordination level, this thesis contributes a distributed topology-management scheme that supports platoon-level operations such as merge, split, and rearrange, while leveraging the predecessor-follower topology to detect inconsistencies in shared information and isolate a compromised vehicle by moving it to the tail of the platoon, a feature that is unique to our methodology.

In Chapter 4, corresponding to Objective (III), we developed Curvature-Aware Model Predictive Contouring Control (CA-MPCC) for urban driving. The core methodological contribution is a reformulation of MPCC that explicitly accounts for path curvature in the geometric error representation and removes the lag-error term used in conventional MPCC. Relying on a more precise path-progres sestimate, CA-MPCC reduces tuning complexity and mitigates failure modes that occur on tight curves when progress approximation drifts

under large deviations. The resulting optimal control problem improves robustness to curvature variations, reduces parameter sensitivity, and remains real-time feasible, as demonstrated in simulation and through hardware experiments on DART on highly curved trajectories.

Chapter 5 extended Objective (III) to high-performance autonomous racing with the rCA-MPCC formulation. This chapter contributes a curvature-informed terminal cost that improves closed-loop behavior during aggressive maneuvers and does not depend on offline solutions such as precomputed racelines, improving transfer to scenarios where only local geometry is available. In addition, it introduces a compact and lightweight reference-path representation that improves the controller reliabilty and overall closed-loop racing trajectories. To further strengthen model-based prediction at high speed, the thesis augments the physics-based vehicle model from Chapter 2 with identified actuator dynamics and a residual dynamics model learned using Gaussian Processes, capturing unmodeled effects. Together, these contributions improve model fidelity and control robustness, enabling faster and more consistent lap times in experiments on small-scale cars. We additionally transfer the formulation to aerial drone racing in simulation, further demonstrating the robustness of the method.

Finally, Chapter 6 demonstrated how Objectives (I) and (III) support broader cooperative robotics applications beyond classical platooning. By combining DART with CA-MPCC, the thesis implements a distributed persistent monitoring and target detection system in which multiple robots track coordinated Lissajous trajectories generated by time-inverted Kuramoto dynamics. The resulting system achieves collision avoidance and complete area coverage using only local communication, and hardware experiments confirm that the high-level distributed coordination integrates reliably with low-level MPC tracking in the presence of disturbances and temporary agent failures.

Overall, this thesis contributes, first, a reproducible experimental platform and identification toolchain that lowers the barrier to credible hardware validation, second, a distributed and attack-resilient cooperative driving framework with safety and string stability guarantees under compromised communication, and third, curvature-aware MPCC formulations for both urban driving and high-performance racing that improve robustness and real-time reliability on highly curved trajectories. These contributions jointly accelerate iteration cycles for autonomous driving research and provide a foundation for future advances toward safer, more robust, and more efficient autonomous transportation systems.

7.2 Future Work

While this thesis advances the experimental and theoretical foundations of autonomous driving, several aspects remain open for future research.

From small to full-scale autonomous driving. The small-scale experiments demonstrated that the DART platform effectively captures the key behaviors of full-scale vehicles. However, a more systematic investigation of this similarity would allow engineers and researchers in the field to rely even more on small-scale testing, resulting in faster and more efficient deployment. In particular, future work should assess the correspondence between small-scale and full-scale vehicles in two main aspects: vehicle dynamics and

sensing capabilities.

Vehicle dynamics. Concerning the vehicle dynamics, we indicate two main areas that would require further research. The first is related to the physical similarity between vehicles of different scales. In the field of aerodynamics, this is a well-investigated topic, for example it is a well known fact that wind speeds need to be higher for small scale testing in a wind tunnel compared to the full-scale model. In vehicle dynamics however, the literature is far more scarce. A noticeable example is [136], where the π theorem is applied to show that vehicle dynamics can be represented by a set of dimensionless parameter groups, such as the ratio of cornering stiffness to vehicle weight. By ensuring these Π groups are consistent between scales, promising results have shown that small-scale platforms can indeed sit within the statistical distribution of typical full-scale vehicles, validating their use as representative testbeds. An interesting future direction would be perform the same statistical analysis on the DART testbed, and map the limit testing conditions of the small-scale vehicle to the full-scale equivalent. This would show the range of full-scale equivalents that the platform can be considered a reliable surrogate for, adding relevance to the experimental findings reported in this thesis.

The second concerns how differences in macro-component complexity, e.g., the brushed electric motor and simplified driveline used on DART compared with multi-stage automotive powertrains, propagate to the overall vehicle dynamics. Simplified actuators expose distinct bandwidth limits and saturation patterns, leading to qualitative changes in the input-output behaviour beyond the quantitative scaling effects discussed above. As an illustrative example of these two effects, consider the tyres. The difference in scale alters the magnitude of the slip-angle-to-lateral-force characteristic curve, while the absence of pressurisation in small-scale tyres changes its qualitative shape.

Sensing capabilities. In this thesis we relied on on-board sensors for the platooning experiments presented in Chapter 3. However, due to constraints on weight, size, and power consumption, the sensing hardware available on the DART platform differs substantially from full-scale automotive-grade sensors. In particular, small-scale Lidars and cameras exhibit significantly larger measurement uncertainty, shorter effective range, and lower signal-to-noise ratios. These limitations restrict the operational speeds at which safe and reliable control can be demonstrated. As a consequence, the platooning experiments were carried out at relatively low speeds, where sensing uncertainty does not compromise collision-avoidance guarantees.

For more aggressive manoeuvres, such as high-speed autonomous racing, we relied on an external motion-capture system, which provides millimetre accuracy and almost negligible latency. This sensing setup enabled us to focus on the control and dynamic modelling aspects of the problem without being limited by the onboard sensor quality. However, it also means that the experiments do not fully replicate the sensing conditions encountered in real-world autonomous vehicles, where perception uncertainty must be explicitly accounted for.

Future work should therefore aim at understanding how sensing uncertainty in the small-scale platform corresponds to that of full-scale systems. This relates to the broader question of physical similarity mentioned in the previous paragraph. For example, full-scale automotive LiDARs have an uncertainty in the order of centimetres at a look-ahead distance of thirty meters while the vehicle travels at 60 km/h. Translating such conditions

to a 1:20 scale platform requires jointly considering the scaled operating speed, stopping distance, and the time available for collision avoidance. In other words, the equivalent sensing uncertainty at small scale should be determined according to the ratio between reaction times, braking dynamics, and overall different vehicle dynamic capabilities.

Developing such a scaling law for sensing uncertainty, would strengthen the connection between small-scale experimentation and full-scale autonomous driving, enabling a more principled use of small-scale platforms for evaluating perception-driven algorithms.

Platooning in mixed traffic. The proposed platooning framework assumes that a malicious entity may only affect the communication network, while all vehicles remain otherwise collaborative. Inserting a human-driven vehicle in the platoon effectively breaks this assumption, since its actions are potentially unpredictable. Extending this framework to mixed traffic scenarios with human-driven or non-cooperative agents imposes more stringent requirements both at the platoon controller and platoon coordinator level.

Concerning safety aspects pertaining to the former, this requires handling unexpected cut-ins, unknown vehicle dynamics, and non-uniform response behaviours. The current reachability-based safety verification would need to be extended to include a wider range of possible worst-case initial conditions. However, since the actions of the non-cooperative vehicle are unpredictable, safety in an absolute sense is no longer achievable, and safety verification would need to be relaxed, for example leaning into responsibility-aware frameworks.

Concerning the platoon coordinator, our current strategy relies on line-of-sight to check for consistency between communicated and observed accelerations of the preceding vehicle. A human-driven vehicle that merges into the platoon may block this visual link, preventing the Kalman-filter-based attack detection module from comparing measurements and communicated signals. In practice, the presence of such a vehicle is equivalent to a Denial-of-Service attack on its predecessor, since it provides no acceleration data. Under the existing coordination logic, this situation would trigger a platoon reconfiguration in which downstream vehicles attempt to overtake the obstructing agent to restore communication integrity. However, executing such a manoeuvre safely is non-trivial, since the non-cooperative vehicle may behave unpredictably. Moreover, from a broader traffic perspective, repeated or aggressive overtaking of human-driven vehicles may be undesirable or socially unacceptable. Extending the coordinator to handle these scenarios therefore requires new strategies for maintaining platoon integrity without assuming full observability or cooperative behaviour. These may include alternative sensing modalities, more flexible platoon-splitting and merging policies, socially aware decision-making rules that balance safety, efficiency, and interaction with human drivers.

Emergency manoeuvres in urban driving and Gaussian Processes for adaptive dynamics modelling. High-speed racing was used as a proxy to study vehicle dynamics at the limits of handling, yet urban emergencies may differ significantly from racing conditions owing to the presence of obstacles, occlusions, and more stringent requirements on reaction time. Before extending rCA-MPCC to these safety-critical contexts, it is essential to clearly identify the types of emergency situations that commonly occur in urban driving. This could be achieved by leveraging large-scale real-world datasets to characterize representative events such as sudden cut-ins, pedestrians emerging from behind occlusions, or temporary

sensor faults. A systematic understanding of these scenarios would enable the design of realistic test cases and provide the basis for an experimental study applying rCA-MPCC to urban emergency manoeuvres, bridging the gap between performance-oriented racing control and safety-critical urban navigation.

The use of Gaussian Process residual models in rCA-MPCC opens promising avenues for adaptation to changing driving conditions, such as variations in road surface condition. To fully leverage this potential, future research should investigate online learning and co-adaptation of both the GP and the underlying physics-based model, enabling the controller to respond to rapidly evolving dynamics. Achieving this goal will require new data collection strategies and mechanisms for automatic controller retuning to guarantee safety and stability under uncertainty, ultimately moving toward adaptive model predictive control frameworks. Following the same philosophy used when reasoning about physical similarity, the experimental study of environment-dependent dynamics would also benefit from replicating relevant atmospheric conditions—such as rain, humidity, or dust in a controlled laboratory setting. Understanding how these conditions scale between small and full-scale vehicles would make it possible to design small-scale experiments whose outcomes transfer meaningfully to real-world driving scenarios, further strengthening the role of miniature robotic platforms in the development of robust, weather-aware autonomous driving systems.

Model Reliability. While the system identification process in Chapter **??** successfully captured the primary dynamics of the DART platform, the selection of specific model structures—such as the friction and motor curves—could be further refined. Future research should include a formal sensitivity analysis to quantify how uncertainties or errors in the identified parameters (e.g., the fitting parameters a, b, c for friction) propagate to the control performance. Such an analysis would provide a more principled basis for model selection and the definition of parameter bounds. An interesting addition to chapter 5 would be to perform an experimental campaign by purposefully changing the values of the model parameters and showing the correlation to the lap time performance. This would show what parameters are the most critical to identify correctly in terms of measurable overall racing performance decrease.

References

References

[1] pxhere.com. busy highway. https://pxhere.com/en/photo/622467.

[2] Iain Thomson. Self-driving cars a reality for 'ordinary people' within 5 years, says google's sergey brin, 2012. Speech at California AV bill signing; news coverage.

[3] Dante d'Orazio. Elon musk predicts a tesla will be able to drive itself across the country in 2018, 2016. Press call / media reports in Oct 2016.

[4] Sam Abuelsamid. Waymo ceo on timeline for ubiquitous autonomy, 2018. Interview / conference remarks; news coverage.

[5] Waymo LLC. Waymo one robotaxi service (u.s.), 2020. Company blog or reputable news article describing public service.

[6] Baidu. Apollo go driverless ride-hailing (china), 2022. Company announcement or reputable news article.

[7] Gamal Elghazaly, Raphaël Frank, Scott Harvey, and Stefan Safko. High-definition maps: Comprehensive survey, challenges, and future perspectives. *IEEE Open Journal of Intelligent Transportation Systems*, 4:527–550, 2023.

[8] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

[9] Yueyuan Li, Wei Yuan, Songan Zhang, Weihao Yan, Qiyuan Shen, Chunxiang Wang, and Ming Yang. Choose your simulator wisely: A review on open-source simulators for autonomous driving. *IEEE Transactions on Intelligent Vehicles*, pages 1–19, 2024.

[10] United Nations Economic Commission for Europe (UNECE). UN Regulation No. 157 (R157): Automated Lane Keeping Systems (ALKS). `https://unece.org/sites/default/files/2025-06/R157r1e.pdf`, 2025. Accessed: 2025-12-13.

[11] Brian Goldfain, Paul Drews, Changxi You, Matthew Barulic, Orlin Velev, Panagiotis Tsiotras, and James M. Rehg. Autorally: An open platform for aggressive autonomous driving. *Control Systems Magazine*, 39:26–55, 2019.

[12] MIT RACECAR. `https://racecar.mit.edu/`.

[13] BARC. `https://closestnum20.com/barc_v4-0/`.

[14] Siddhartha S. Srinivasa, Patrick E. Lancaster, Johan Michalove, Matt Schmittle, Colin Summers, Matthew Rockett, Joshua R. Smith, Sanjiban Choudhury, Christoforos Mavrogiannis, and Fereshteh Sadeghi. Mushr: A low-cost, open-source robotic racecar for education and research. *ArXiv*, abs/1908.08031, 2019.

[15] Donckey Car. `https://http://docs.donkeycar.com/`.

[16] Marc Green. " how long does it take to stop?" methodological analysis of driver perception-brake times. *Transportation human factors*, 2(3):195–216, 2000.

[17] Rens Van der Heijden, Thomas Lukaseder, and Frank Kargl. Analyzing attacks on cooperative adaptive cruise control (cacc). In *2017 IEEE Vehicular Networking Conference (VNC)*, pages 45–52. IEEE, 2017.

[18] Niloofar Jahanshahi and Riccardo MG Ferrari. Attack detection and estimation in cooperative vehicles platoons: A sliding mode observer approach. *IFAC-PapersOnLine*, 51(23):212–217, 2018.

[19] Yang Zheng, Shengbo Eben Li, Keqiang Li, and Wei Ren. Platooning of connected vehicles with undirected topologies: Robustness analysis and distributed h-infinity controller synthesis. *IEEE Transactions on Intelligent Transportation Systems*, 19(5):1353–1364, 2017.

[20] Xu Jin, Wassim M Haddad, Zhong-Ping Jiang, Aris Kanellopoulos, and Kyriakos G Vamvoudakis. An adaptive learning and control architecture for mitigating sensor and actuator attacks in connected autonomous vehicle platoons. *International Journal of Adaptive Control and Signal Processing*, 33(12):1788–1802, 2019.

[21] Simone Baldi, Di Liu, Vishrut Jain, and Wenwu Yu. Establishing platoons of bidirectional cooperative vehicles with engine limits and uncertain dynamics. *IEEE Transactions on Intelligent Transportation Systems*, 22(5):2679–2691, 2020.

[22] Anye Zhou, Jian Wang, and Srinivas Peeta. Robust control strategy for platoon of connected and autonomous vehicles considering falsified information injected through communication links. *Journal of Intelligent Transportation Systems*, pages 1–17, 2022.

[23] Ben Groelke, Christian Earnhardt, John Borek, and Chris Vermillion. A predictive command governor-based adaptive cruise controller with collision avoidance for non-connected vehicle following. *IEEE Transactions on Intelligent Transportation Systems*, 23(8):12276–12286, 2021.

[24] Randolph Hall and Chinan Chin. Vehicle sorting for platoon formation: Impacts on highway entry and throughput. *Transportation Research Part C: Emerging Technologies*, 13(5-6):405–420, 2005.

[25] Sebastian Van De Hoef, Karl H Johansson, and Dimos V Dimarogonas. Fuel-optimal centralized coordination of truck platooning based on shortest paths. In *2015 American control conference (ACC)*, pages 3740–3745. IEEE, 2015.

[26] Aseem Borkar, Arpita Sinha, Leena Vachhani, and Hemendra Arya. Collision-free trajectory planning on lissajous curves for repeated multi-agent coverage and target detection. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1417–1422. IEEE, 2016.

[27] Rajesh Rajamani. *Vehicle dynamics and control*. Springer Science & Business Media, 2011.

[28] Yash Raj Khusro, Yanggu Zheng, Marco Grottoli, and Barys Shyrokau. MPC-based motion-cueing algorithm for a 6-DOF driving simulator with actuator constraints. *Vehicles*, 2(4):625–647, 2020.

[29] Laura Ferranti, Lorenzo Lyons, Rudy R Negenborn, Tamás Keviczky, and Javier Alonso-Mora. Distributed nonlinear trajectory optimization for multi-robot motion planning. *IEEE Transactions on Control Systems Technology*, 31(2):809–824, 2022.

[30] Benjamin David Evans, Raphael Trumpp, Marco Caccamo, Felix Jahncke, Johannes Betz, Hendrik Willem Jordaan, and Herman Arnold Engelbrecht. Unifying f1tenth autonomous racing: Survey, methods and benchmarks. arXiv, 2024. arXiv:2402.18558.

[31] Denise Lam, Chris Manzie, and Malcolm Good. Application of model predictive contouring control to an XY table. *IFAC Proceedings Volumes*, 44(1):10325–10330, 2011.

[32] Alessandro Rucco, Giuseppe Notarstefano, and John Hauser. An efficient minimum-time trajectory generation strategy for two-track car vehicles. *IEEE Transactions on Control Systems Technology*, 23(4):1505–1519, 7 2015.

[33] Maria Krinner, Angel Romero, Leonard Bauersfeld, Melanie Zeilinger, Andrea Carron, and Davide Scaramuzza. Mpcc++: Model predictive contouring control for time-optimal flight with safety constraints. *arXiv preprint arXiv:2403.17551*, 2024.

[34] Waveshare. Jetracer pro ai kit – high speed ai racing robot powered by jetson nano. `https://www.waveshare.com/jetracer-pro-ai-kit.htm`. Accessed: 2025-12-07.

[35] Lyons Lorenzo. DART. `https://github.com/Lorenzo-Lyons/DART`, 2024.

[36] Lorenzo Lyons, Thijs Niesten, and Laura Ferranti. Dart: A compact platform for autonomous driving research. In *2024 IEEE Intelligent Vehicles Symposium (IV)*, pages 129–136. IEEE, 2024.

[37] Egil Juliussen. Robotaxis: What Is Going On?, 8 2023.

[38] Yash Rajan. Top 5 truck platooning brands syncing convoy trucks via connective technology, 1 2022.

[39] Neil Winton. Computer Driven Autos Still Years Away Despite Massive Investment, 2 2022.

[40] ROBOTIS. TurtleBot. https://www.turtlebot.com/.

[41] Duckietown. Duckiebot. https://get.duckietown.com/products/duckiebot-db21?variant=40700056895663.

[42] JetRacerProAI. https://www.waveshare.com/wiki/JetRacer_Pro_AI_Kit.

[43] Lennart Ljung. Perspectives on system identification. *Annual Reviews in Control*, 34(1):1–12, 2010.

[44] Hans B Pacejka and Egbert Bakker. The magic formula tyre model. *Vehicle system dynamics*, 21(S1):1–18, 1992.

[45] Guillaume Baffet, Ali Charara, and Daniel Lechner. Estimation of vehicle sideslip, tire force and wheel cornering stiffness. *Control Engineering Practice*, 17(11):1255–1264, 2009.

[46] Jason Kong, Mark Pfeiffer, Georg Schildbach, and Francesco Borrelli. Kinematic and dynamic vehicle models for autonomous driving control design. In *2015 IEEE intelligent vehicles symposium (IV)*, pages 1094–1099. IEEE, 2015.

[47] Characteristics of Brushed DC Motors. https://techweb.rohm.com/product/motor/brushed-motor/brushed-motor-basic/209/.

[48] William Menke. Chapter 9 - detecting and understanding correlations among data. In William Menke, editor, *Environmental Data Analysis with MatLab® or Python (Third Edition)*, pages 277–317. Academic Press, third edition edition, 2022.

[49] Manuel Boldrer, Lorenzo Lyons, Luigi Palopoli, Daniele Fontanelli, and Laura Ferranti. Time-inverted kuramoto model meets lissajous curves: Multi-robot persistent monitoring and target detection. *IEEE Robotics and Automation Letters*, 8(1):240–247, 2022.

[50] Lorenzo Lyons and Laura Ferranti. Curvature-aware model predictive contouring control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3204–3210. IEEE, 2023.

[51] Reliable Robot Control lab. Lidar-based lane following. https://youtu.be/QgXHbPtGkdc.

[52] Global truck driver shortage to double by 2028, says new iru report, 2023. Accessed: 29-10-2024.

[53] Farhan Ahmad, Asma Adnane, Virginia NL Franqueira, Fatih Kurugollu, and Lu Liu. Man-in-the-middle attacks in vehicular ad-hoc networks: Evaluating the impact of attackers' strategies. *Sensors*, 18(11):4040, 2018.

[54] Lorenzo Lyons, Thijs Niesten, and Laura Ferranti. DART: A Compact Platform for Autonomous Driving Research. In *2024 IEEE Intelligent Vehicles Symposium (IV)*, pages 129–136, 2024.

[55] Veronika Lesch, Martin Breitbach, Michele Segata, Christian Becker, Samuel Kounev, and Christian Krupitzer. An overview on approaches for coordination of platoons. *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[56] C Canudas De Wit and Bernard Brogliato. Stability issues for vehicle platooning in automated highway systems. In *Proceedings of the 1999 IEEE International Conference on Control Applications (cat. no. 99ch36328)*, volume 2, pages 1377–1382. IEEE, 1999.

[57] Yang Zheng, Shengbo Eben Li, Keqiang Li, Francesco Borrelli, and J Karl Hedrick. Distributed model predictive control for heterogeneous vehicle platoons under unidirectional topologies. *IEEE Transactions on Control Systems Technology*, 25(3):899–910, 2016.

[58] Sebastian Thormann, Alexander Schirrer, and Stefan Jakubek. Safe and efficient cooperative platooning. *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[59] Mohammad Hossein Basiri, Benyamin Ghojogh, Nasser L Azad, Sebastian Fischmeister, Fakhri Karray, and Mark Crowley. Distributed nonlinear model predictive control and metric learning for heterogeneous vehicle platooning with cut-in/cut-out maneuvers. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 2849–2856. IEEE, 2020.

[60] Alexander L Gratzer, Sebastian Thormann, Alexander Schirrer, and Stefan Jakubek. String stable and collision-safe model predictive platoon control. *IEEE Transactions on Intelligent Transportation Systems*, 23(10):19358–19373, 2022.

[61] Elham Semsar-Kazerooni, Jan Verhaegh, Jeroen Ploeg, and Mohsen Alirezaei. Cooperative adaptive cruise control: An artificial potential field approach. In *2016 IEEE Intelligent Vehicles Symposium (IV)*, pages 361–367. IEEE, 2016.

[62] Stefania Santini, Alessandro Salvi, Antonio Saverio Valente, Antonio Pescapé, Michele Segata, and Renato Lo Cigno. A consensus-based approach for platooning with intervehicular communications and its validation in realistic scenarios. *IEEE Transactions on Vehicular Technology*, 66(3):1985–1999, 2016.

[63] Aidin Ferdowsi, Ursula Challita, Walid Saad, and Narayan B Mandayam. Robust deep reinforcement learning for security and safety in autonomous vehicle systems. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 307–312. IEEE, 2018.

[64] Kang Yang, Di Liu, Simone Baldi, Wenwu Yu, and Chen Lv. Decoupling-based resilient control of vehicular platoons under injection of false wireless data. *IEEE Transactions on Intelligent Transportation Systems*, 2024.

[65] Twan Keijzer, Paula Chanfreut, José María Maestre, and Riccardo Maria Giorgio Ferrari. Collaborative vehicle platoons with guaranteed safety against cyber-attacks. *IEEE Transactions on Intelligent Transportation Systems*, 2024.

[66] Ge Guo, Jian Kang, Hongbo Lei, and Dandan Li. Finite-time stabilization of a collection of connected vehicles subject to communication interruptions. *IEEE Transactions on Intelligent Transportation Systems*, 23(8):10627–10635, 2021.

[67] Feng Zhongwei, Keyun Qin, Xiaohang Jiao, Feifei Du, and Dongshen Li. Cooperative adaptive cruise control for vehicles under false data injection attacks. In *2023 IEEE 6th International Conference on Industrial Cyber-Physical Systems (ICPS)*, pages 1–5. IEEE, 2023.

[68] Alexander Johansson, Ehsan Nekouei, Karl Henrik Johansson, and Jonas Mårtensson. Multi-fleet platoon matching: A game-theoretic approach. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2980–2985. IEEE, 2018.

[69] Floriano De Rango, Mauro Tropea, Pierfrancesco Raimondo, Amilcare Francesco Santamaria, and Peppino Fazio. Bio inspired strategy for improving platoon management in the future autonomous electrical vanet environment. In *2019 28th International Conference on Computer Communication and Networks (ICCCN)*, pages 1–7. IEEE, 2019.

[70] Timo Sturm, Christian Krupitzer, Michele Segata, and Christian Becker. A taxonomy of optimization factors for platooning. *IEEE Transactions on Intelligent Transportation Systems*, 22(10):6097–6114, 2020.

[71] Christian Krupitzer, Michele Segata, Martin Breitbach, Samy El-Tawab, Sven Tomforde, and Christian Becker. Towards infrastructure-aided self-organized hybrid platooning. In *2018 IEEE Global Conference on Internet of Things (GCIoT)*, pages 1–6. IEEE, 2018.

[72] Simona Sacone, Cecilia Pasquale, Silvia Siri, and Antonella Ferrara. Centralized and decentralized schemes for platoon control in freeway traffic systems. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 2665–2670. IEEE, 2021.

[73] Prasad Vegendla, Tanju Sofu, Rohit Saha, Mahesh Madurai Kumar, and Long-Kung Hwang. Investigation of aerodynamic influence on truck platooning. Technical report, SAE Technical Paper, 2015.

[74] Distributed Attack-Resilient Platooning Against False Data Injection. `https://github.com/Lorenzo-Lyons/Distributed-Attack-Resilient-Platooning-Against-False-Data-I` Accessed: 30-05-2024.

[75] Distributed Attack-Resilient Platooning Against False Data Injection. `https://youtu.be/M0ukDXu6Mk0`. Accessed: 30-05-2024.

[76] Sahand Hadizadeh Kafash, Jairo Giraldo, Carlos Murguia, Alvaro A Cardenas, and Justin Ruths. Constraining attacker capabilities through actuator saturation. In *2018 Annual American Control Conference (ACC)*, pages 986–991. IEEE, 2018.

[77] Yulong Cao, S Hrushikesh Bhupathiraju, Pirouz Naghavi, Takeshi Sugawara, Z Morley Mao, and Sara Rampazzi. You can't see me: Physical removal attacks on {LiDAR-based} autonomous vehicles driving frameworks. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 2993–3010, 2023.

[78] Michael H Shaham, Risha Ranjan, Engin Kırda, and Taşkın Padır. Design and realization of a benchmarking testbed for evaluating autonomous platooning algorithms. In *International Symposium on Experimental Robotics*, pages 582–594. Springer, 2023.

[79] Andrew J Kerns, Daniel P Shepard, Jahshan A Bhatti, and Todd E Humphreys. Unmanned aircraft capture and control via gps spoofing. *Journal of field robotics*, 31(4):617–636, 2014.

[80] Mani Amoozadeh, Arun Raghuramu, Chen-Nee Chuah, Dipak Ghosal, H Michael Zhang, Jeff Rowe, and Karl Levitt. Security vulnerabilities of connected vehicle streams and their impact on cooperative driving. *IEEE Communications Magazine*, 53(6):126–132, 2015.

[81] Sergey V Drakunov and Vadim I Utkin. Sliding mode control in dynamic systems. *International Journal of Control*, 55(4):1029–1037, 1992.

[82] Norbert Bißmeyer, Sebastian Mauthofer, Kpatcha M Bayarou, and Frank Kargl. Assessment of node trustworthiness in vanets using data plausibility checks with particle filters. In *2012 IEEE Vehicular Networking Conference (VNC)*, pages 78–85. IEEE, 2012.

[83] Eman Mousavinejad, Fuwen Yang, Qing-Long Han, Quanwei Qiu, and Ljubo Vlacic. Cyber attack detection in platoon-based vehicular networked control systems. In *2018 IEEE 27th International Symposium on Industrial Electronics (ISIE)*, pages 603–608. IEEE, 2018.

[84] Cheng-Zong Bai, Vijay Gupta, and Fabio Pasqualetti. On kalman filtering with compromised sensors: Attack stealthiness and performance bounds. *IEEE Transactions on Automatic Control*, 62(12):6641–6648, 2017.

[85] Ozan Tonguz, Nawapom Wisitpongphan, Fan Bai, Priyantha Mudalige, and Varsha Sadekar. Broadcasting in vanet. In *2007 mobile networking for vehicular environments*, pages 7–12. IEEE, 2007.

[86] R Craig Coulter et al. *Implementation of the pure pursuit path tracking algorithm*. Carnegie Mellon University, The Robotics Institute, 1992.

[87] Lorenzo Lyons and Laura Ferranti. Curvature-aware model predictive contouring control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3204–3210. IEEE, 2023.

[88] Nikolaus Lang, Andreas Herrmann, Markus Hagenmaier, and Maximilian Richter. Can Self-Driving Cars Stop the Urban Mobility Meltdown?, 8 2020.

[89] The 6 Levels of Vehicle Autonomy Explained.

[90] Yan Ding. Three Methods of Vehicle Lateral Control: Pure Pursuit, Stanley and MPC, 3 2020.

[91] Akhil Nagariya and Srikanth Saripalli. An iterative lqr controller for off-road and on-road vehicles using a neural network dynamics model. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 1740–1745, 2020.

[92] Max Schwenzer, Muzaffer Ay, Thomas Bergs, and Dirk Abel. Review on model predictive control: an engineering perspective, 11 2021.

[93] Suiyi He, Jun Zeng, and Koushil Sreenath. Autonomous racing with multiple vehicles using a parallelized optimization with safety guarantee using control barrier functions. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 3444–3451, 2022.

[94] Nishant Chowdhri, Laura Ferranti, Felipe Santafé Iribarren, and Barys Shyrokau. Integrated nonlinear model predictive control for automated driving. *Control Engineering Practice*, 106:104654, 2021.

[95] Denise Lam, Chris Manzie, and Malcolm Good. Model predictive contouring control. In *Proceedings of the IEEE Conference on Decision and Control*, pages 6137–6142. Institute of Electrical and Electronics Engineers Inc., 2010.

[96] Alisa Rupenyan, Mohammad Khosravi, and John Lygeros. Performance-based trajectory optimization for path following control using bayesian optimization. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 2116–2121, 2021.

[97] Roger Skjetne, Thor I. Fossen, and Petar V. Kokotović. Robust output maneuvering for a class of nonlinear systems. *Automatica*, 40(3):373–383, 3 2004.

[98] Wilko Schwarting, Javier Alonso-Mora, Liam Pauli, Sertac Karaman, and Daniela Rus. Parallel autonomy in automated vehicles: Safe motion generation with minimal intervention. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 1928–1935. Institute of Electrical and Electronics Engineers Inc., 7 2017.

[99] Oscar De Groot, Bruno Brito, Laura Ferranti, Dariu Gavrila, and Javier Alonso-Mora. Scenario-Based Trajectory Optimization in Uncertain Dynamic Environments. *IEEE Robotics and Automation Letters*, 6(3):5389–5396, 7 2021.

[100] Alexander Liniger, Alexander Domahidi, and Manfred Morari. Optimization-based autonomous racing of 1: 43 scale RC cars. *Optimal Control Applications and Methods*, 36(5):628–647, 2015.

[101] Jose L. Vazquez, Marius Bruhlmeier, Alexander Liniger, Alisa Rupenyan, and John Lygeros. Optimization-based hierarchical motion planning for autonomous racing. In *IEEE International Conference on Intelligent Robots and Systems*, pages 2397–2403. Institute of Electrical and Electronics Engineers Inc., 10 2020.

[102] Sirish Srinivasan, Sebastian Nicolas Giles Nicolas Giles, and Alexander Liniger. A Holistic Motion Planning and Control Solution to Challenge a Professional Racecar Driver. *IEEE Robotics and Automation Letters*, 6(4):7854–7860, 10 2021.

[103] Alain Micaelli and Claude Samson. *Trajectory tracking for unicycle-type and two-steering-wheels mobile robots*. PhD thesis, INRIA, 1993.

[104] Laura Ferranti, Bruno Brito, Ewoud Pool, Yanggu Zheng, Ronald M Ensing, Riender Happee, Barys Shyrokau, Julian F P Kooij, Javier Alonso-Mora, and Dariu M Gavrila. SafeVRU: A research platform for the interaction of self-driving vehicles with vulnerable road users. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 1660–1666, 2019.

[105] John C Mason and David C Handscomb. *Chebyshev polynomials*. Chapman and Hall/CRC, 2002.

[106] David Q Mayne, James B Rawlings, Christopher V Rao, and Pierre O M Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, 2000.

[107] A. Zanelli, A. Domahidi, J. Jerez, and M. Morari. Forces nlp: an efficient implementation of interior-point... methods for multistage nonlinear nonconvex programs. *International Journal of Control*, pages 1–17, 2017.

[108] Lorenzo Lyons and Laura Ferranti. Curvature-Aware Model Predictive Contouring Control, 2023.

[109] Dr Xiaoxi He, John Li. Autonomous vehicles market 2025-2045: Robotaxis, autonomous cars, sensors, 2024. Accessed: 2025-09-29.

[110] Carsten Heer. Sales of service robots up 30 Accessed: 2025-09-30.

[111] Fortune Business Insights. Warehouse robotics market size, share and industry analysis, 2025. Accessed: 2025-09-30.

[112] IFPRI . The global drone revolution in agriculture, 2025. Accessed: 2025-09-30.

[113] FACT.MR . Surveillance drone market, 2025. Accessed: 2025-09-30.

[114] Stephen Council. Cruise, waymo win watershed votes to expand driverless car ride-hailing in san francisco, 2023. Accessed: 2025-09-30.

[115] Zoox. Introducing zoox, 2024. Accessed: 2025-09-30.

[116] Jouav. How fast can a drone fly? top speeds of various types, 2025. Accessed: 2025-09-30.

[117] Manuel Boldrer, Alvaro Serra-Gomez, Lorenzo Lyons, Javier Alonso-Mora, and Laura Ferranti. Rule-based lloyd algorithm for multi-robot motion planning and control with safety and convergence guarantees. *arXiv preprint arXiv:2310.19511*, 2023.

[118] Johannes Betz, Hongrui Zheng, Alexander Liniger, Ugo Rosolia, Phillip Karle, Madhur Behl, Venkat Krovi, and Rahul Mangharam. Autonomous vehicles on the edge: A survey on autonomous vehicle racing. *IEEE Open Journal of Intelligent Transportation Systems*, 3:458–488, 2022.

[119] Drew Hanover, Antonio Loquercio, Leonard Bauersfeld, Angel Romero, Robert Penicka, Yunlong Song, Giovanni Cioffi, Elia Kaufmann, and Davide Scaramuzza. Autonomous drone racing: A survey. *IEEE Transactions on Robotics*, 40:3044–3067, 2024.

[120] Ben Tearle, Kim P Wabersich, Andrea Carron, and Melanie N Zeilinger. A predictive safety filter for learning-based racing control. *IEEE Robotics and Automation Letters*, 6(4):7635–7642, 2021.

[121] Federico Pizarro Bejarano, Lukas Brunke, and Angela P Schoellig. Safety filtering while training: Improving the performance and sample efficiency of reinforcement learning agents. *IEEE Robotics and Automation Letters*, 2024.

[122] Robin Verschueren, Stijn De Bruyne, Mario Zanon, Janick V Frasch, and Moritz Diehl. Towards time-optimal race car driving using nonlinear mpc in real-time. In *53rd IEEE conference on decision and control*, pages 2505–2510. IEEE, 2014.

[123] José L Vázquez, Marius Brühlmeier, Alexander Liniger, Alisa Rupenyan, and John Lygeros. Optimization-based hierarchical motion planning for autonomous racing. In *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 2397–2403. IEEE, 2020.

[124] Denise Lam, Chris Manzie, and Malcolm Good. Model predictive contouring control. In *49th IEEE Conference on Decision and Control (CDC)*, pages 6137–6142. IEEE, 2010.

[125] Lorenzo Lyons. curvature-aware-mpcc-pkg: Curvature-aware model predictive contouring control. `https://github.com/Lorenzo-Lyons/curvature-aware-mpcc-pkg`, 2025. GitHub repository, last accessed 2025-11-15.

[126] Arindam Chaudhuri. B-splines. In *Encyclopedia of Computer Graphics and Games*, pages 242–252. Springer, 2024.

[127] Natanael Karjanto. Properties of chebyshev polynomials. *arXiv preprint arXiv:2002.01342*, 2020.

[128] Jie Wang. An intuitive tutorial to gaussian process regression. *Computing in Science & Engineering*, 25(4):4–11, 2023.

[129] James R. Gardner, Andreas Klein, et al. Gpytorch: Blackbox Gaussian Processes on Pytorch, 2023.

[130] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[131] Manuel Boldrer, Lorenzo Lyons, Luigi Palopoli, Daniele Fontanelli, and Laura Ferranti. Time-inverted kuramoto model meets lissajous curves: Multi-robot persistent monitoring and target detection. *IEEE Robotics and Automation Letters*, 8(1):240–247, 2022.

[132] Manuel Boldrer, Francesco Riz, Fabio Pasqualetti, Luigi Palopoli, and Daniele Fontanelli. Time-inverted kuramoto dynamics for κ-clustered circle coverage. In *60th Conf. on Dec. and Control (CDC)*, pages 1205–1211. IEEE, 2021.

[133] Bruno Brito, Boaz Floor, Laura Ferranti, and Javier Alonso-Mora. Model predictive contouring control for collision avoidance in unstructured dynamic environments. *IEEE Robotics and Automation Letters*, 4(4):4459–4466, 2019.

[134] Jason Kong, Mark Pfeiffer, Georg Schildbach, and Francesco Borrelli. Kinematic and dynamic vehicle models for autonomous driving control design. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, pages 1094–1099, 2015.

[135] Manuel Boldrer, Lorenzo Lyons, Luigi Palopoli, Daniele Fontanelli, and Laura Ferranti. Video associated with the submission.

[136] Using a scale testbed: Controller design and evaluation. *IEEE Control Systems Magazine*, 21(3):15–26, 2002.

Acknowledgments

In this last section, I would like to extend my sincere gratitude to everyone who was there for me throughout this journey. This achievement carries within it a bit of your support and guidance, I hope you may also feel a little proud. I will try to tell story of how this academic milestone has come to fruition, presenting the characters as they appear.

My adventure started entirely thanks to my dear friend Lorenzo (Bernardo), it is indeed thanks to him that I came across the vacancy that in the end brought me here, who knows where I would be without him.

As many of you will know, moving to the Netherlands comes with the arduous task of finding a house. It was the Von Gus who came with me all the way from Italy to help me look. Curiously, although she had always pushed me to study and work abroad, she was quite surprised when I asked her if she would come with me. Rebellious as I was, I think she never imagined I would actually listen to her. As I was preparing to leave the country, my father's top concern was making sure I had a decent tool box. I indulged him at the time, but as it turned out, he was right. Building a house worth of IKEA furniture is not a trivial task.

Then came the actual moving in, buying all the furniture for the new empty house, and my first day of work. Arguably, it's not a brilliant idea to have all three events on the same day, but this is how the story went. I was very lucky to have Valentina's support in those first few weeks, and for the following few years.

Laura the first person to trust I could be useful. From the very beginning she relied on me to work with the new robots we had just unboxed, and to look after my very first master student, Thijs. Although, to be fair, we both know it was more the other way around, I hope I was able to return the favor at least a little bit.

I started my PhD just as COVID-19 was ending, and there were indeed quite a few of us that started around the same time for this reason. The first person I met in my new office (It even had my name on the door!) was Gustavo. An expert in both coding and reading people's emotions, although not so good at keeping track of where he's shoes were. They would indeed mysteriously disappear from time to time. Luckily, Pablo always seemed to know their exact location, a fact he claims is pure coincidence. The most elusive member of the office was by far Ashwin, who's schedule remains a mystery to this day. Italo joined the department a few weeks after me, I remember getting the "hello everyone" email from the secretariaat and rushing to be the first to answer. We have been good friends ever since. Later came Mossi, thanks to whom I overcame my fear of integral terms in PID controllers, and Jelle, my favorite source of dutch humor. The last person to join was Lasse, a brilliant researcher and a great human being. I still have much to learn from him in both fields.

I like to refer to the middle of my PhD as the "golden age", it was for a good few years that my generation of PhDs, the group that started as COVID-19 was ending, overlapped with the people that started a few years before us. I am truly thankful for all the countless dinners we all spent laughing and drinking together.

From the "older" generation I would like to thank Giovanni, for the unofficial mentoring and for conveying his enthusiasm about GPs (that are much better than NNs). Alvaro for the interest and the ability to always ask the right questions, and for teaching me a few cool salsa moves. Mnauel, the man with many names, I wish to thank for all the hours we spent together scribbling math away on various pieces of paper, for supervising me in secret (as we would later discover) and for setting the example for how chill someone can be. I extend a special thank you also to Rodrigo, who had his office in the fancy second floor, and would often come down stairs in the afternoon to pop into my office and share a word of wisdom or two. A habit that I took up myself when I also moved to the second floor during my last year.

It's a good moment to introduce the younger generation, Mariano and Beatrice. To Mariano I owe my mental health during the last year, specially thanks to the morning coffee gossip sessions and the mandatory 3pm fruit brake (death upon you if you used work as an excuse not to join). Beatrice I thank for always being ready to lend a supporting ear.

I now wish to thank all the people that I couldn't weave smoothly into the story, I apologize for my lack of writing prowess. Maia for her energy and good heart. Her Bobyness for her unshakeable determination. Anton for his insightful boxing tips. My sister Camilla, from whom I am still trying to learn the art of not stressing too much. Edoardo, Gianmarco and Damiano for coming to visit way more often than what curtsy would require, showing me that good friendships endure no matter the physical distance. Adriano, with whom I have shared so much.

Lastly I would like to thank Cristina, for putting up with me during the final stretch. Your level head and outstanding resilience are a never ending source of admiration.

Lorenzo
Delft, June 2026

List of Publications

JOURNAL ARTICLES

1. **L. Lyons**, M. Boldrer, L. Ferranti, Distributed attack-resilient platooning against false data injection, *IEEE Transactions on Vehicular Technology*, early access, 2025, doi: 10.1109/TVT.2025.3614452.

2. M. Boldrer, **L. Lyons**, L. Palopoli, D. Fontanelli, L. Ferranti, Time-inverted Kuramoto model meets Lissajous curves: Multi-robot persistent monitoring and target detection, *IEEE Robotics and Automation Letters*, vol. 8, no. 1, pp. 240–247, 2022.

3. L. Ferranti, **L. Lyons**, R. R. Negenborn, T. Keviczky, J. Alonso-Mora, Distributed nonlinear trajectory optimization for multi-robot motion planning, *IEEE Transactions on Control Systems Technology*, vol. 31, no. 2, pp. 809–824, 2022.

4. M. Shokri, **L. Lyons**, S. Pequito, L. Ferranti, Battery Identification With Cubic Spline and Moving Horizon Estimation for Mobile Robots, *IEEE Transactions on Control Systems Technology*, vol. 32, no. 5, pp. 1944–1951, 2024.

PRE-PRINTS

5. M. Boldrer, Á. Serra-Gomez, **L. Lyons**, J. Alonso-Mora, L. Ferranti, Rule-based Lloyd algorithm for multi-robot motion planning and control with safety and convergence guarantees, *arXiv preprint arXiv:2310.19511*, 2023.

CONFERENCE PAPERS

6. **L. Lyons**, T. Niesten, L. Ferranti, DART: A Compact Platform for Autonomous Driving Research, in *2024 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2024, pp. 129–136.

7. **L. Lyons**, L. Ferranti, Curvature-aware model predictive contouring control, in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2023, pp. 3204–3210.

Included in this thesis.

Curriculum Vitæ

Lorenzo Lyons

Personal Data
Place and Date of Birth: Rome, Italy | 15 October 1995
Email: l.lyons@tudelft.nl

Education

2021–2025	Ph.D. candidate, Delft University of Technology, The Netherlands. Topic: Modelling, control, and coordination for autonomous driving on a reproducible robotic platform.
2018–2021	M.Sc. Mechanical Engineering (Robotics and Mechatronics track), Politecnico di Milano, Italy (110 *cum laude*). Thesis: Optimal control of partial differential equations for drone swarm applications.
2014–2018	B.Sc. Mechanical Engineering, University of Rome Tor Vergata, Italy (110 *cum laude*).

TU Delft