Degree thesis

Bachelor's degree in Industrial Technology Engineering (GETI)

Non-linear Model Predictive Control for a Formula Student Racing Autonomous Vehicle

June 30, 2022

Author: Antoni Salom Llabrés

Directors: Vicenç Puig Cayuela

Convocatòria: 07/2022

Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona

Preface

Acknowledgements

First of all I would like to acknowledge and thank my fellow teammates of the BCNeMotorsport team, who make possible that all the work and hours that take to develop a work like this can be reflected on an amazing newly fabricated race car as the CAT14x. Also I thank the most to the colleagues of the Control, Electronics and Perception departments who compose the driverless team with whom I've been working side by side in the last year. Lastly, I would also like to thank to my mentors and also colleagues of my first year in the team as member of the Autonomous Systems Department, in the season 2020-2021.

I would also like to thank Vicenç Puig Cayuela, my advisor for this work for his help, advises and his full time disposal for any question or meeting.

Finally, I would also like to thank my family for making possible for me to be living abroad study and work in projects such as the Formula Student.

Abstract

The aim of this work is to design, implement and test a full control stack for a Formula Student racing vehicle and the subsequent analysis and study of the parameters and results. The control stack is composed by a Non-Linear Model Predictive Controller using the Curvilinear Dynamic Bicycle Model of the vehicle and a 2^{nd} derivative continuous Online Planning Algorithm.

The final objective is obtain a full control stack for the BCNeMotorsport car for season 2021-2022, the CAT14x, to compete in the Formula Student summer competitions and excel in all of them against the top teams and to approach as much as possible the performance of the driverless car to a human driver.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Formula Student

To understand the context in which this work has been done, the first matter to explain is what is Formula Student (FS). It is one of the top international student competitions that includes student teams from all the best tech universities in the world. The competition consists in the design, manufacturing and testing an F1 style racing car from scratch to compete against rival universities in the summer competitions. The first FS competition was carried out in the year 1981 in the US. From that point it started to spread all over the world and to Europe. In 2006, the first edition of Formula Student Germany (FSG) took place. Nowadays FSG is the most prestigious and reference FS competition in Europe. Every year, more than 70 FS teams from all over the world compete in the Hockenheim Ring in three categories: FSE (Formula Student Electric), FSC (Formula Student Combustion) and FSD (Formula Student Driverless). In the year 2017, the driverless competition was included in FSG, as pioneers of the innovation and ambition present in the Formula Student competition. This year's competition, the 2022 edition, has had a major rules change, the FSD competition is erased, and the driverless events are now added to the overall FSE competition, this way, every car must have an operational autonomous system to be able to win the competition. Moreover, a new competition/trophy is added, the Driverless Cup, aside of the driverless events present in the FSE competition, the best teams with the best autonomous system compete to obtain the maximum amount of points in the Driverless Cup.

Source: Formula Student Germany webpage [13]
Figure 1: Formula Student Germany 2021.

The Formula Student competition points are divided in dynamic and static events. The FS encourages not only building the fastest car, but also the design and its justification is evaluated in the static events.

	CV & EV	DC
Static Events:		
Business Plan Presentation	75 points	-
Cost and Manufacturing	100 points	-
Engineering Design	150 points	150 points
Dynamic Events:		
Skid Pad	50 points	-
DV Skid Pad	75 points	75 points
Acceleration	50 points	-
DV Acceleration	75 points	75 points
Autocross	100 points	-
DV Autocross	-	100 points
Endurance	250 points	-
Efficiency	75 points	-
Trackdrive	-	200 points
Overall	1000 points	600 points

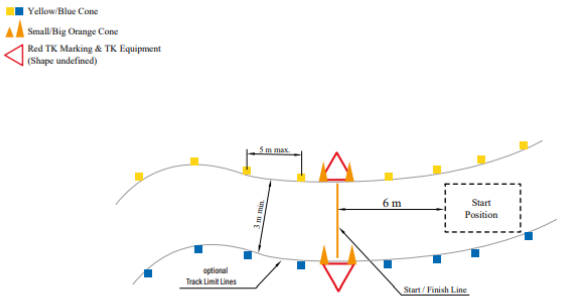
Source: FSG Rules 2022 [1]
Table 1: FSG point distribution chart.

The two implementations stated in this work are destined to the driverless dynamic events of the FSE competition and the DC. The controller is implemented to control the throttle and steering of the car in all the dynamic events, and the planner is destined to the autocross event. In the following subsections, all these dynamic events will be explained and later the specifications and requirements extracted from these events.

1.1.1 DV Autocross

In the first place, the autocross is the main event where the whole autonomous system is tested. It consists of a single fast lap to an unknown track delimited by blue cones on the left and yellow cones on the right. Each team has three opportunities to try to score the best laptime. The fact that the track is unknown, determines the need of a perception system to detect the cones on run time, to later use the control algorithms to plan the trajectory and finally control the car.

The layout consists of a track no longer than 500m and not shorter than 200m, with straights up to 80m, turns up to 50m diameter and hairpin turns up to 9m of outside diameter.



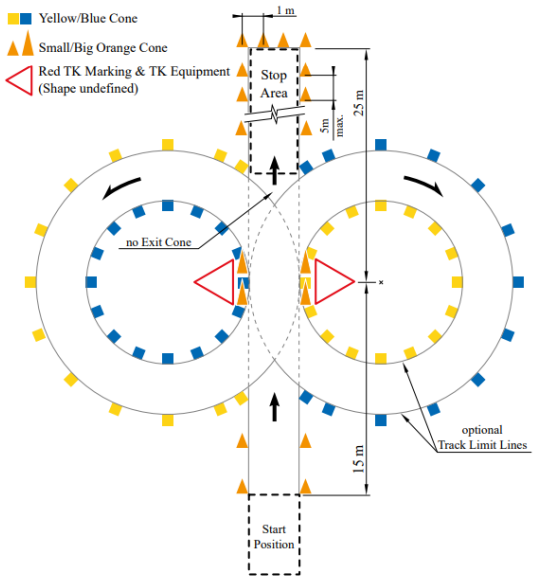
Source: FSG Hanbook 2022 [2]
Figure 2: Autocross layout.

1.1.2 DV Trackdrive

The second main event in the FS competition is the trackdrive. It consists on an endurance event of ten laps to the same track as the autocross event. This fact makes possible to save the autocross map, and compute the optimal trajectory offline. Then, at the time of the event, in-map localization is needed to localize the car in the track and the loaded map. In this way, with the pre-computed optimal trajectory, the controller is able to achieve much higher velocities.

1.1.3 DV Skidpad

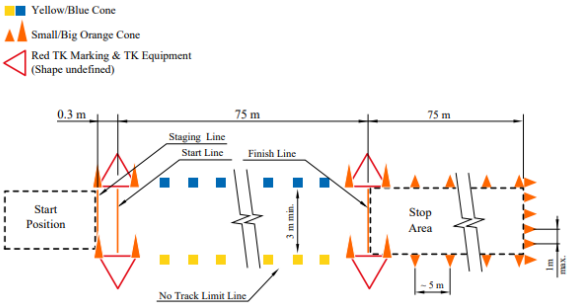
The skidpad consists of a track which is always the same in all the competitions. It is an eight shaped track, with one left and one right circle. The car must take two right turns first, two left turns, and finally brake and stop safely in the designated area. Therefore, the track is known and the trajectory can be pre-computed, but the car needs a perception algorithm to detect the cones and re-localize the theoretical map with the actual map, since the initial position can not be perfectly established.



Source: FSG Hanbook 2022 [2]
Figure 3: Skidpad layout.

1.1.4 DV Acceleration

Finally, the acceleration event consists of an straight line of 150m, also with blue cones at the left and yellow on the right. The time is measured from the beginning to the ending line at 75m. This event is destined to test the limits of the vehicle and show the potential of FS cars, very lightweight powerful cars built for acceleration rather than peak velocity.



Source: FSG Hanbook 2022 [2]
Figure 4: Acceleration layout.

1.2 BCN eMotorsport

To keep going with the work context, a main part is the FS team BCN eMotorsport. BCN eMotorsport is the FS team of the ETSEIB and ETSETB associated to the Universitat Politècnica de Catalunya, in Barcelona. The team was funded in the year 2007 as ETSEIB motorsport with its first combustion car, the CAT01. In 2011, the team took the great step of the electrification, building the first electric car of the team, the CAT05e. Finally, in 2018-2019 with the fusion between ETSEIB Motorsport and Driverless UPC, birthed BCN eMotorsport. The goal of the current

team is to build and to prove that the new car, the CAT14x is the fastest driven and driverless car in Spain.

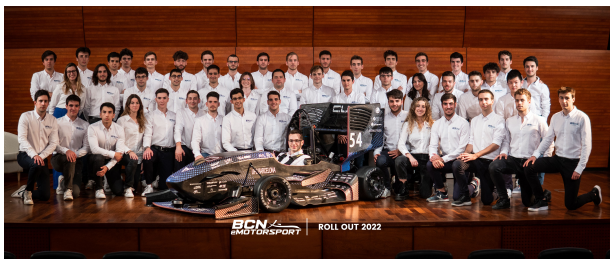


Figure 5: BCN eMotorsport team.

Figure 6: CAT14x.

1.3 State of the art

It is important to study and research the actual state-of-the-art in the control and planning algorithms nowadays before starting the implementation of any algorithm addressing these problems. This makes up the basis of any work or implementation, and not only in the FS environment, but in any other application.

1.3.1 Formula Student

In the specific case of the FS, three different categories of teams can be differentiated. First, the top teams, most of them German teams. They are the oldest and historical teams who have been in the FS for more years and count with the legacy of the knowledge of the previous generations. Tipically, they compete for the first places in the competitions and normally share they knowledge for the FS community to advance and evolve their own teams generation after generation. On the other hand, there are the mid-teams, which are the ones which have enough budget to manufacture a good car, and advance thanks to the top teams. Finally, the lowest budget teams that normally are starting up. It is from this top teams that the state-of-the-art FS comes from.

Driverless control The main approach from top teams such as *AMZ racing*, with references such as [9][10][12] from Zürich, Switzerland, or *Formula Student team Delft* from Delft, Netherlands, is to develop a model predictive controller to obtain steering and acceleration controls given a reference path. Then, the variations come in the MPC formulation and vehicle model used, that differ depending on the teams' means. Other approches such as the *KA RaceIng* team, with references such as [11], from Germany, are to develop an MPC for the steering commands and an heuristic-armed PID controller with a velocity profile reference for torque control.

Driverless planning From the online trajectory planning point of view, some teams opt to the track limits plus path implementations via space-discretization algorithms such as the Voronoi or the Delaunay algorithm. Other teams, such as *AMZ racing*, optimize the midline curvature to obtain an optimum smooth path.

Figure 7: AMZ racing driverless car.

Figure 8: TU Delft driverless car.

1.3.2 Other applications

It is also important to widen the scope and also research and study other applications, not only the ones like the one that is being addressed. Model Predictive Controller have been widely used in multiple applications in the industry world, and have proven to have outstanding results in lots of different applications. For example, a similar application to the one being addressed but outside the FS, is the one proposed in [3]. A MPC designed for air navigation of a four-motor drone capable of controlling the drone under harsh environments. In this paper, an MPC taking into account a very different model with different physical characteristics, also having amazing results.

1.4 FS control specifications and requeriments

Regarding the requirements and specifications of the controller, they are defined by the FS competition and precisely the dynamic events in which the vehicle has to take part on. The controller is destined to control the car in all the dynamic events previously described, obviously with a different set of parameters for optimum performance in all of the different events.

1.4.1 Requeriments

- **Processing Unit** able to compute and execute the algorithms in a C++ and ROS framework.
- **Planning** algorithm to use the cones from the track to calculate the reference trajectory for the controller to follow.
- **Actuators** with the capability of controlling the car in the track. Corresponding to an autonomous steering system, an autonomous system brake and access to the inverter for output torque.
- **Sensors** for controller data feedback from the real performance of the car. Specifically, the localization of the car in the track, the actual steering angle, velocity and acceleration.
- **Communication** network between the PU and the ECUs controlling the actuators and also to receiving the data from the sensors.

1.4.2 Specifications

- *Reliability* and safety even at high speeds up to **100** $\frac{km}{h}$.
- Sufficient *Rate* for the output controls to be able to control the car without abrupt changes. Minimum of **20 Hz** and as high as possible to obtain better performance.
- *Fast* and racing driving style with average velocities of **15** $\frac{m}{s}$.

1.5 FS planning specifications and requirements

As well as the controller specifications, the planning ones are determined by the FS competition. Specifically, the planner is destined to the autocross event, where the car must finish one lap as fast as possible to an unknown track described before.

1.5.1 Requirements

- **Processing Unit** able to compute and execute the algorithms in a C++ and ROS framework.
- **Perception** system to detect the blue and yellow cones and output the limits of the track.
- Actual **State** of the car (position) to be able to publish the correct part of the trajectory to the MPC.

1.5.2 Specifications

- *Continuous* **curvature** therefore 2_{nd} derivative continuous trajectory.
- Take advantage of *all data* (**cones**) from the track limits, not limited to some of them for computation time.
- *Fast solving* for any number of new cones. Minimum of **20 Hz**.

1.6 Objectives

The first objective of this work is to obtain an MPC algorithm and an online planning algorithm that fit with all the system's and environment's specifications and requirements for the FS competition. Secondly, to obtain a controller that is capable of driving a FS racing car at high speeds in an smooth and controlled way, as close as possible to a human driver. In the third place, to justify a new implementation of both algorithms, it has to be assured that they are an improvement of the previous season's algorithms.

The final objective is to obtain competitive laptimes and scorings in the dynamic events, **Trackdrive** and **Autocross**, compared to the top teams in the FS summer competitions.

1.7 Workplan

The workplan is subject to a four-month period in the FS season corresponding to the academic period. The time span of the work invested in the two implementation is 20 weeks and IS divided in the two main implementations with their corresponding subtasks.

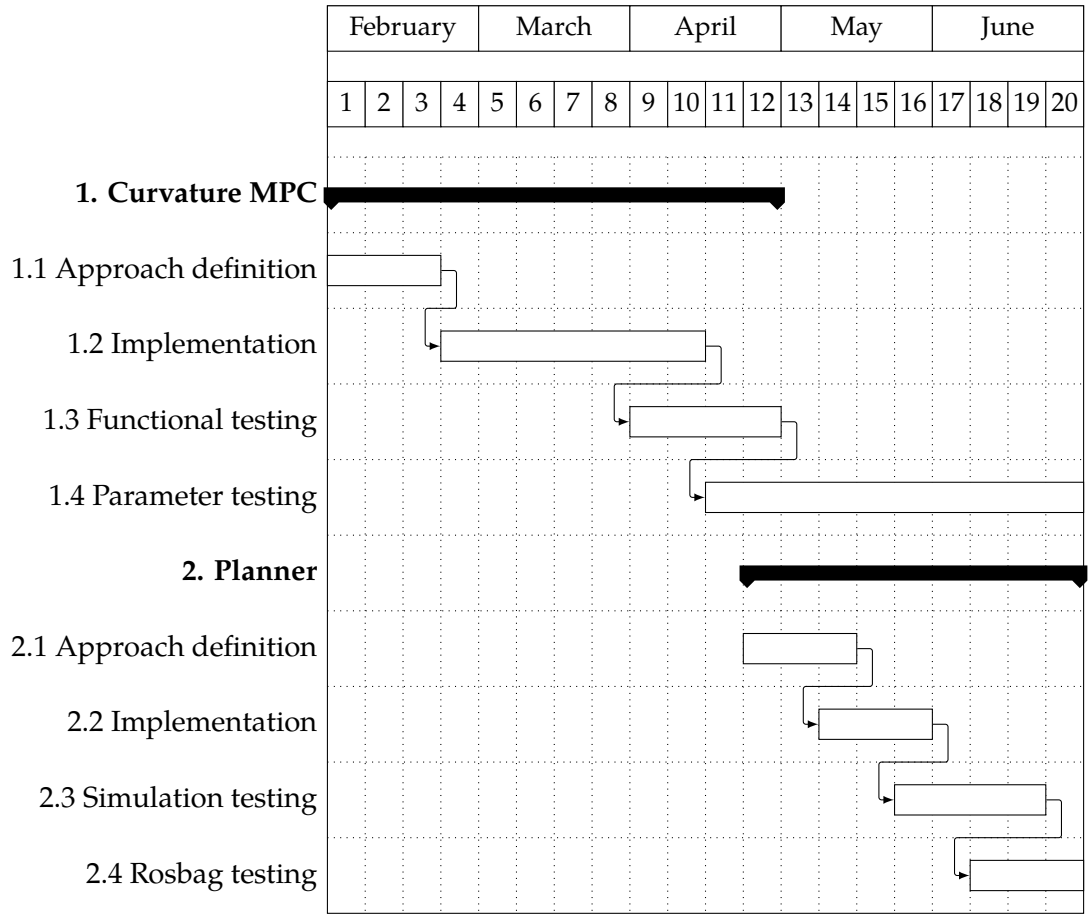


Table 2: Workflow Gantt chart.

As shown in the previous Gantt Chart 2, the first two months were only destined to the Curvature MPC implementation, since it was the main goal of the season. But at half implementation, it was seen that a new planning algorithm was needed, since the curvature output from the previous one did not fit the new controller. Therefore, the second phase of the work was started which was the new implementation. Nevertheless, the parameter tuning and controller testing in simulation work did not stop.

The work time is estimated as 5h of work per day, with some extras for extra tasks, and 5 days of work per week. With this estimated values the time dedicated to each task is distributed in the following way:

Task	h/day	day/week	Weeks	Amount(h)
Approach definition	5	5	3	75
Implementation	5	5	10	250
Functional testing	3	5	4	60
Parameter testing	1	5	10	50
Total hours				435

Table 3: MPC hours distribution chart.

Task	h/day	day/week	Weeks	Amount(h)
Approach definition	3	5	3	45
Implementation	5	5	3	75
Simulation testing	1	5	4	20
Rosbag testing	1	5	10	50
Total hours				190

Table 4: Planner hours distribution chart.

2 Model Predictive Control

2.1 Concepts

Model predictive control is a control technique based on iterative optimization to obtain optimum controls for a plant model, which can be linear on non-linear, satisfying a series of restrictions in a prediction window, addressed as prediction horizon. The predictive control techniques have a set of principles which are discussed in the following subsection. All the theory and knowledge is extracted from references such as e.g. [8].

2.1.1 Predictive model

A key concept in predictive control is the model of the plant which is aimed to control. Depending on the system, the model that describes its behaviour can be assumed linear or highly non-linear. In this thesis, the non-linear model predictive control will be addressed, since the vehicle model is a highly non-linear system.

2.1.2 Optimization

To obtain the optimal controls as stated, an optimization problem is formulated. In the cases as the addressed in this work, multivariable optimization problems are used. Moreover, a cost function over the horizon is stated, as one of the most important elements in any optimization. Finally, a set of constraints, consisting in a number of equalities or inequalities that define the limits of the system is considered.

An example of the usual formulation of an MPC optimization problem is the stated in the following equations:

$$\begin{aligned} \min_{y,u} \quad & \sum_{k=0}^{N-1} \left|W^y(y_{k+1}(t) - r(t))\right|^2 + \left|W^{\Delta u}\Delta u_k\right|^2 \\ \text{s.t.} \quad & y_k \in \chi \\ & u_k \in \Upsilon \\ & y_{k+1} = f(y_k, u_k) \\ & k = 0, .., N \end{aligned} \tag{1}$$

W^y, W^u : diagonal state and control variable slew rate weight matrix.

y_k, u_k : state and control variables at time k.

$r(t)$: reference on time t.

2.2 Vehicle model

As mentioned in Subsection 2.1 , the model of the plant plays a great role in predictive control. In the case addressed, the system that is aimed to control is a FS racing vehicle, therefore the vehicle model that describes the behaviour of the car in the track is needed.

Vehicle modeling is a problem in itself and has been widely studied. There are many possible ways to describe the vehicle model for control. Each one of them with its own advantages and disadvantages. In [4], dynamic bicycle and four-wheel approaches are stated. In this work,

only the kinematic and dynamic bicycle models will be addressed, since the four-wheel model integration will be left for future work.

2.2.1 Kinematic bicycle model

As a first approach to vehicle modeling the option of the kinematic bicycle model is discussed. This kind of modeling, consists on simplifying the four-wheel model to a two-wheel model referred to as bicycle model. Moreover, as a first approach, the dynamic behaviour of the vehicle is not addressed, therefore, only the kinematic behaviour is described, neglecting the forces in the tyres and only taking into account the position and velocity terms. This model is widely used in robotics and some vehicle control applications. It yields a very simple formulation that is suitable for low speeds and basic applications.

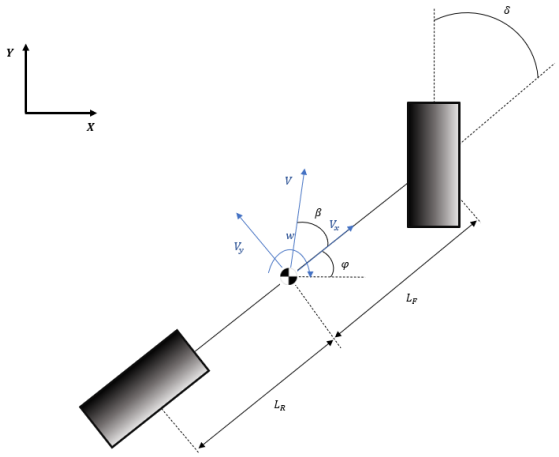


Figure 9: Kinematic bicycle model.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\varphi} \\ \dot{v_x} \\ \dot{v_y} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} v_x \cos(\varphi+\beta) - v_y \sin(\varphi+\beta) \\ v_x \sin(\varphi+\beta) + v_y \cos(\varphi+\beta) \\ w \\ a_x \\ a_y \\ \ddot{\varphi} \end{bmatrix} \quad (2)$$

The β angle is the real displacement angle,

$$\beta = \tan^{-1}\left(\frac{L_R}{L_F + L_R}\tan(\delta)\right) \quad (3)$$

2.2.2 Dynamic bicycle model

Following the same path as in the previous model, the dynamic bicycle model is the evolution to the kinematic but including the inertial behaviour of the car. Thus, the dynamic model takes into account the forces on the front and rear tyres and the momentum caused by them. In this way, the model describes how the evolution of the car state in time produce a series of deformations on the tyre which generate the forces that move the system. Thus, not only the model of the car must be included, but the model of the tyre as well. This matter has also been widely studied. The most important reference is [5] where the Pacejka tire model and magic formula which will be the ones used in this work is introduced.

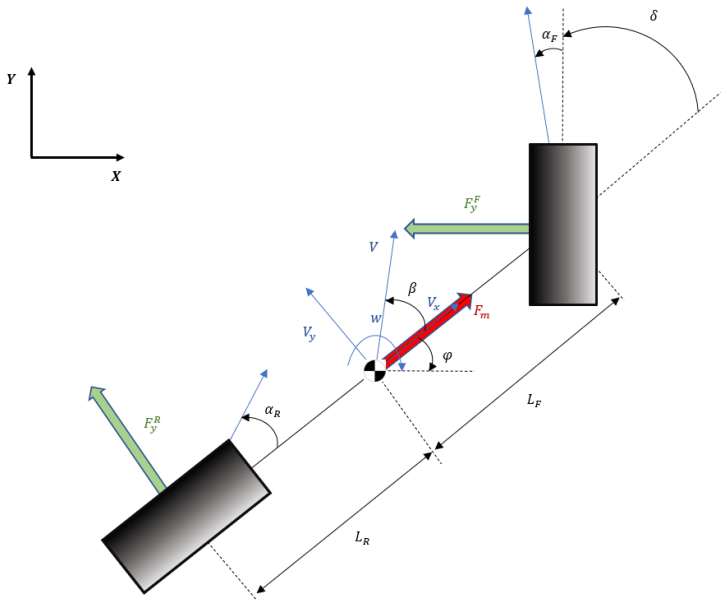


Figure 10: Dynamic bicycle model.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\varphi} \\ \dot{v_x} \\ \dot{v_y} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} v_x\cos(\varphi) - v_y\sin(\varphi) \\ v_x\sin(\varphi) + v_y\cos(\varphi) \\ w \\ \frac{1}{m}(F_x - F_y^F\sin\delta + mv_yw) \\ \frac{1}{m}(F_y^R + F_y^F\cos\delta - mv_xw) \\ \frac{1}{I_z}(F_y^F\cos(\delta)L_f - F_y^RL_r) \end{bmatrix} \tag{4}$$

In this model, the longitudinal or motor force is assumed to be in the COG, therefore the sum of longitudinal forces F_x, including the rolling resistance and the aerodynamic drag is,

$$F_x = F_m - C_{r0} - C_dv_x^2 \tag{5}$$

The lateral, rear and front (R/F), forces $F_y^{R/F}$, are calculated using the simplified magic Pacejka formula [5], which depend from the slip angles $\alpha_{R/F}$,

$$\alpha_F = \arctan(\frac{vy - L_fw}{v_x}) - \delta \tag{6}$$

is the front slip angle and

$$\alpha_R = \arctan(\frac{vy + L_r w}{v_x}) \tag{7}$$

is the rear slip angle. Finally, the lateral, rear and front (R/F), forces $F_y^{R/F}$, are calculated as follows

$$F_y^{R/F} = D_{R/F} \sin(C_{R/F} \arctan(B_{R/F} \alpha_{R/F})) \tag{8}$$

The Pacejka constants, $D_{R/F}, C_{R/F}, B_{R/F}$, are inertial parameters that depend on the tyre and the mass of the vehicle, which will be studied later in this work.

2.2.3 Criterion and final approach

After introducing the vehicle modeling and mostly of the bicycle model, the final approach for the model used in the design and implementation of the controller is a dynamic bicycle model formulated in curvilinear coordinates. The previous model described in the Subsection 2.2.2 is formulated in ordinary bi-dimensional Cartesian coordinates (X, Y). Alternatively, the following model is formulated in curvilinear coordinates (n, μ). In addition, since the vehicle is equipped with a motor on each wheel, for more precise description of the dynamics, the motor force is assumed to be divided equally on each tyre and not in the COG. Finally, since the low-level control of the vehicle controls department is destined to translate a percentage of pedal travel, the motor force F_m is described as

$$F_m = C_m D, \quad -1 < D < 1 \tag{9}$$

Then, a new parameter is added, C_m, in the motor model that depends on the motor, low-level control and car specifications that given a $D = 1$ the maximum possible longitudinal force is provided and a $D = -1$ maximum braking force is achieved. Finally, since the low-level control provides the possibility of torque vectoring, since only two wheels are presented in the bicycle model, to evaluate the torque vectoring effect a new variable is added M_{tv}, whose value is the extra momentum provided by the difference in wheel torques. In addition, the possibility of including it as a control, allows the controller to use the torque vectoring beyond the rule-based methods designed for drivers.

Moreover, the state and controls for the optimization problem formulation are also included and defined.

The curvilinear coordinates are used to simplify the path formulation and the model formulation. The kinematic states that define the curvilinear coordinates are (n, μ, s). The state s defines the progress along the reference curve and it also determines the curvature along the path $k(s)$. The n state refers to the orthogonal deviation to the path and, finally, the μ state defines the heading relative to the tangent line to the path. Therefore, given these three states the curve and the path following formulation is completely defined given the $k(s)$ at progress s from the reference trajectory.

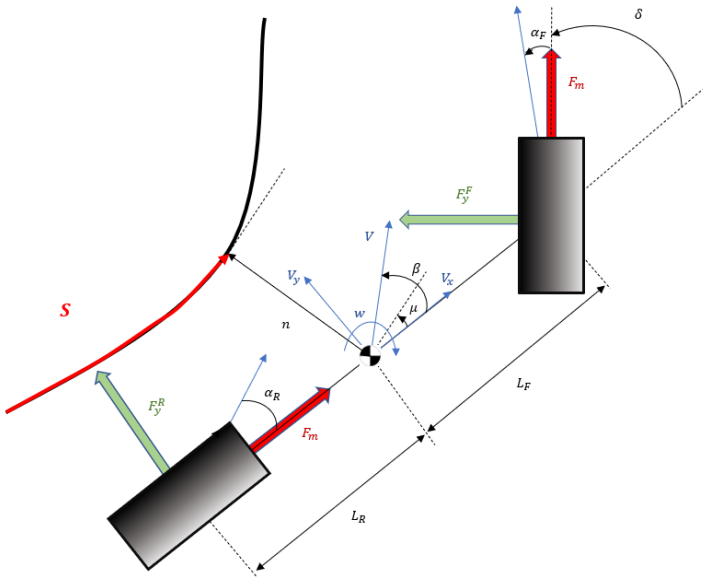


Figure 11: Curvilinear dynamic bicycle model.

Given all the variables as described in the previous Figure 11, the vehicle state is described by $x = [D, \delta, s, n, \mu, v_x, v_y, w]$ and the controls are defined as $u = [\Delta D, \Delta\delta, M_{tv}]$

$$\begin{bmatrix} \dot{D} \\ \dot{\delta} \\ \dot{s} \\ \dot{n} \\ \dot{\mu} \\ \dot{v_x} \\ \dot{v_y} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} \Delta D \\ \Delta\delta \\ \frac{v_x \cos(\mu) - v_y \sin(\mu)}{1 - nk(s)} \\ v_x \sin(\mu) + v_y \cos(\mu) \\ w - k(s)\dot{s} \\ \frac{1}{m}(F_x - F_y^F \sin(\delta) + m v_y w) \\ \frac{1}{m}(F_y^R + F_m \sin(\delta)) + F_y^F \cos(\delta) - m v_x w \\ \frac{1}{I}(F_y^F \cos(\delta) + F_m \sin(\delta)) L_F - F_y^R L_R + M_{tv} \end{bmatrix} \tag{10}$$

The model in curvilinear coordinates is formulated as stated. Forces F_x and $F_y^{R/F}$ are defined as in equations (5) and (8), and the terms corresponding to the momentum generated for changing the F_m point of application and M_{tv} are added.

The derivative of s, defined as the progress rate $\dot{s}$, describes a crucial state for the MPC optimization formulation, since it is developed to race as fast as possible, therefore maximizing the progress rate forces the controller to advance along the reference path as fast as possible. In this way, the velocity is not a target but it is maximized. As stated in system (10), $\dot{s}$ depends on the velocities and on $k(s)$. This anticipates that the controller will be very curvature dependant, since it is the only information beside s that it is getting from the reference path. This behaviour will be later addressed in this work.

2.3 Optimization problem formulation

Following the Subsection 2.1, once the predictive model is defined, for the controller design, the next step is to formulate the optimization problem. The problem is formulated with the objective of achieving, firstly, the less laptime as possible (velocity maximizing), secondly, soft controls to not damage the hardware, and finally, to keep the car into the dynamic bounds defined by the model and the track. To simplify and to reduce the computational cost of the optimization, the state vector x is redefined as $x = [D, \delta, n, \mu, v_x, v_y, w]$, since the s-state can be predicted using the other optimized variables and $k(s)$ is not calculated into the MPC but it is extracted from the reference track. Nonetheless, the progress rate will be calculated to add it to the cost function. The vehicle model (10) is stated as $f(x, u)$ and N is the prediction horizon used. The formulation of the problem is as follows

$$\begin{aligned} \min_{x,u} \quad & \sum_{k=0}^{N} J_{MPC} \\ \text{s.t.} \quad & x_{k+1} = f(x_k, u_k), \;\; x_k \in \chi, \\ & u_k \in \Upsilon, \\ & x_k \in \Lambda_{Track}, \;\; x_k \in \Lambda_{Ellipse}, \\ & k = 0, 1, ..., N, \end{aligned} \tag{11}$$

In the next subsections, every part of this formulation will be further detailed

2.3.1 Cost function

The cost function defined as J_{MPC} is the most important factor of the optimization and is composed by the following terms

$$J_{MPC} = -q_{sd}\dot{s}_k + q_n n_k^2 + R_D(\Delta D_k)^2 + R_\delta(\Delta\delta_k)^2 + R_{M_{tv}}(M_{tvk})^2 + L(x_k) \tag{12}$$

The $-q_{sd}$ weight is negative since the objective is to maximize the progress rate $\dot{s}$. The weight q_n aims at making the car to follow the reference track, penalizing n-state high values. Moreover, the control softening weights $R_D, R_\delta, R_{M_{tv}}$ are positive weights included to penalize abrupt control changes. Finally, the $L(x_k)$ function is included to penalize the difference between the kinematic and dynamic slip angles to achieve a more swift driving,

$$\begin{aligned} & L(x_k) = q_\beta(\beta_{dyn,k} - \beta_{kin,k}), \\ & q_\beta > 0, \\ & \beta_{kin,k} = \arctan(\frac{\delta_k l_R}{l_F + l_R}), \\ & \beta_{dyn,k} = \arctan(\frac{v_{y,k}}{v_{x,k}}), \end{aligned} \tag{13}$$

2.3.2 Constraints

Finally, the optimization problem constraints represent the mechanic and dynamic limits of the car. In the first place, the state has bounds destined to keep the car in the track, χ is defined as the feasible set of states. Also the controls are bounded by hardware limits i.e. the maximum the car can steer δ_{max}, Υ is defined as the feasible set of controls.

With the intention of restricting the car to the inside of the track, track restrictions are added,

$$\begin{aligned} n + L_1 \sin(\lfloor \mu \rfloor) + L_2 \cos \mu \leq FREE_L(s) \\ -n + L_1 \sin(\lfloor \mu \rfloor) + L_2 \cos \mu \leq FREE_R(s) \end{aligned} \tag{14}$$

The expressions L_1, L_2 are the distances from de CoG to the left and right furthest corners of the car and $FREE_L(s), FREE_R(s)$ determine the free distance from the reference track to the track limit. The feasible space is defined as Λ_{Track}. Finally, the equations are reformulated to obtain a 0 on the right side, for solving optimization and commodity reasons. Moreover, slack variables are introduced because if the car exits the track we want it to re-enter, not leading to an infeasibility,

$$\begin{aligned} n + L_1 \sin(\lfloor \mu \rfloor) + L_2 \cos \mu - FREE_L(s) - s_{t,L} \leq 0 \\ -n + L_1 \sin(\lfloor \mu \rfloor) + L_2 \cos \mu - FREE_R(s) - s_{t,R} \leq 0 \end{aligned} \tag{15}$$

The last restriction corresponds to the friction ellipse. The car forces must be within the ellipse to keep the model being valid, since outside of the bounds of the ellipse the non-linear behaviour of the forces is not described by the model,

$$(\rho_{long} F_M)^2 + (F_y^{R/F})^2 \leq (\lambda D_{R/F})^2 \tag{16}$$

Where ρ_{long} is the tire ratio and λ is the parameter associated to the maximum combined force. The $\Lambda_{Ellipse}$ defines the feasible space delimited by the ellipse that is included as a constraint in the optimization problem. Since this constraint will be used to control the racing style, later in this work the constraint will be analyzed. Moreover, quadratic non linear constraints are easily violated, so slack variables (s_R, s_L) are included to avoid infeasibilities,

$$\left(\frac{\rho_{long} F_M}{\lambda D_{R/F}} \right)^2 + \left(\frac{F_y^{R/F}}{\lambda D_{R/F}} \right)^2 - s_{e,R/F} - 1 \leq 0 \tag{17}$$

2.3.3 FORCESPro solver

The chosen solver to solve the optimization problem formulated in Subsection 2.3 is a commercial software provided by the software company *Embotech*, which is sponsor of the BCN eMotorsport team. The FORCESPro solver is implemented and commercialized destined to MPC run-time problem solving. It is optimized in size and speed to obtain best results even with highly non-linear problem such as the described in this work.

FORCESPro provides a MATLAB and Python High-Level Interface for the problem formulation and later code generation for embedded systems to speed up the solving process.

As every controller, the MPC needs feedback to be able to control the driverless car. This feedback is given through the initial state at the first stage of every iteration. The data measured from the sensors is set. The solver also provides the option to facilitate an initial solution to speed up the solving process. This initial solution used is the previous solution if the iteration was optimum or the midpoint of the bounds. In case it is not optimum.

Moreover, the solver needs the curvature at the point corresponding to each stage. This is given by the planner, but the planner does not provide directly these points. The corresponding curvature must be chosen from all the points of the planner. This is computed in two different

ways, depending if in the last iteration an optimum was found or not. In case an optimum is not found, a sampling distance is specified and the points of the planner reference are chosen equispaced. In the case, in the previous iteration an optimum is found, using the equation of the model (10) corresponding to the progress rate $\dot{s} = \frac{v_x \cos(\mu) - v_y \sin(\mu)}{1 - nk(s)}$ the next N arc lengths, s, are predicted and then chosen from the planner.

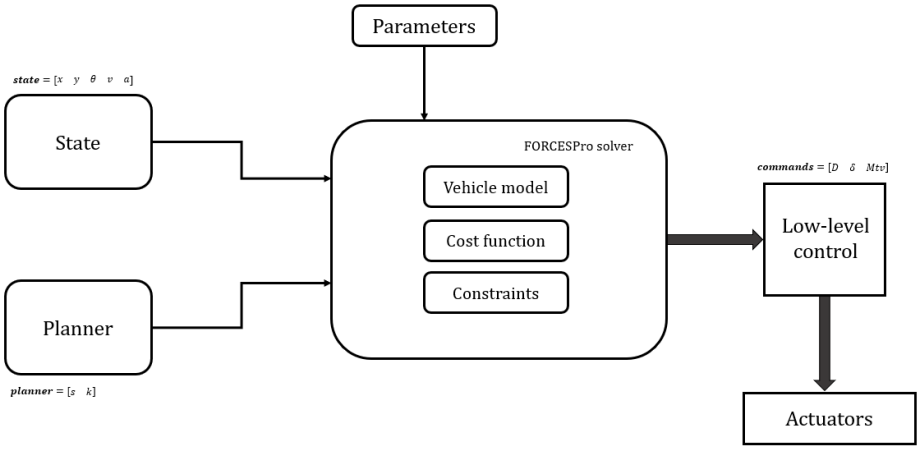


Figure 12: FORCESPro and MPC diagram.

2.4 Implementation

All the autonomous system, control and electronics algorithms used in the FS car, are implemented under a ROS 1 Noetic and C++ framework, running in a self-developed Processing Unit with a linux Gnu-x86_64 operating system. The PU provides a very controlled and flexible environment along with a 20 core processor with high computational power to run the most computationally demanding algorithms.

All the implementation of the MPC algorithm in C++ circles around the MATLAB client and the generated solver code provided by FORCESPro. Using the High-Level MATLAB interface, the formulation of the optimization problem is implemented in MATLAB and then the code is generated in C into a C++ callable library. Afterwards, the ROS nodes are implemented in C++ and use the libraries and executables generated by the MATLAB client. In the MATLAB implementation, the number of parameters, variables, bounds, etc. are specified and then the generated code requires a set of vectors corresponding to each of the specified requirements that must be provided to the solver on run-time. The C++ node manages the real-time information provided by the ROS and sensors framework and uses the solver to solve the optimization problem, therefore obtaining the optimum controls for the stages specified. Finally, the node is also in charge of publishing these commands and sending them back into the network for the low-level control algorithms which are destined to send the orders directly to the actuators, and consequently, move the driverless car.

From the algorithm implementation point of view, several points must be considered, before entering the ROS loop, the MPC object parameters and car state is initialized. Firstly, inside the loop, the parameters are updated with the *Dynamic Reconfigure Tool*, which permits online

parameter tuning. Afterwards, the state and planner are received via their callbacks. For the mpc **state feedback**, defined as $x = [D, \delta, s, n, \mu, v_x, v_y, w]$, the velocity and steering states are directly received from the sensors, the D feedback is obtained from the longitudinal acceleration data from the accelerometer, then the progress rate is obtained from the planner and finally the n and μ states are calculated with the position data from de LiDAR and the reference trajectory.

2.4.1 Algorithm pseudocode

Algorithm 1 Curvature MPC

$mpc \leftarrow mpc :: mpc()$
$p \leftarrow initial\_parameters$
{Initialize MPC and parameters}
$state \leftarrow initial\_state$
{Initialize car state}
$r \leftarrow ros :: Rate()$
{Set MPC frequency}
while $ros :: ok$ **do**
  $p \leftarrow$ **ros::callback_dynConfig()**
  {Dynamic reconfigure callback}
  $state \leftarrow$ **ros::callback_state()**
  {State callback}
  $r \leftarrow$ **ros::callback_planner()**
  {Planner reference callback}
  if $exit\_flag == 1$ **then**
    $planner \leftarrow mpc :: compute\_sPrediction()$
  else
    $planner \leftarrow s\_sampling$
  end if
  {Get N reference points from planner}
  $all\_parameters, ub, lb \leftarrow mpc :: parameter\_setup()$
  {Set solver params and bounds}
  mpc::CurvatureSolver_solve(planner)
  {Solve optimization problem with FORCESPro }
  $solution, exit\_flag \leftarrow mpc :: get\_solution()$
  {Get N solutions and exit flag from solver}
  $ros :: msg\ commands \leftarrow solution$
  {Prepare ROS message}
  if $!finished$ **then**
    $ros :: publish(commands)$
  else
    $commands \leftarrow mpc :: safeStop()$
    $ros :: publish(commands)$
  end if
  {If not finished, publish commands}
  $ros :: sleep(r)$
  $ros :: spinOnce()$
  {Spin ROS thread to publish at MPC frequency}
end while

2.5 Results and data analysis

In the following section, the results and data of the MPC implementation will be shown. The controller will be tested in a simulation environment used in the team BCN eMotorsport, which is an evolution of a public Github repository from one of the top FS teams AMZ racing. The simulation provides an environment according to the FS competition and all the physics and dynamic models and perception, are developed by the team. The simulation has proven to be valuable for algorithm and controller testing, but also afterwards a lot of testing with the car in the real world is required.

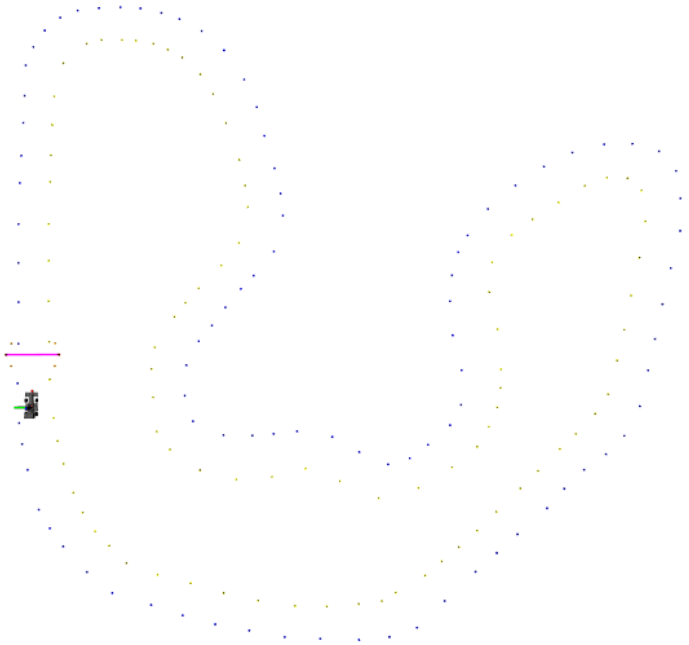


Figure 13: RVIZ visualization of Germany 2019 track.

To test the controller, the reference used is an offline computed optimal path to a known track, calculated by the Trajectory Racing Optimizer algorithm of the team BCN eMotorsport. The trajectories calculated by the TRO have proven to be correct and optimal in the tracks which are the perfect tools to test and optimize the controller. The following tests and analyses are destined to test the MPC and understand how the essential parameters affect the car behaviour on track. The objective is to obtain the lowest laptime possible whilst achieving smooth controls not to damage the hardware.

The FORCESPro solver provides an exit flag to every solver iteration, this way there are different possible exit flags. In the first place, when the optimum was found the solver exit flag output is 1. Secondly, if a solution is found but it's not an optimal solution, the exit flag is 0. Finally, if the problem is unfeasible, the solver is not capable of finding a solution, the exit flag is -7. Therefore, the objective is to obtain always exit flag 1 to achieve the required performance, sometimes, the solver can not find and optimum, in sharp turns or difficult parts of the track, this does not make the car to go off track, but the laptime and contol smoothness are compromised.

2.5.1 Cost function data analysis

The cost function terms have proven to be very important, not only in the performance of the car in the track, but also the feasibility of the problem. The most important terms in the cost function will be set to different values and velocity profile, laptime and the % of 0 exit flags will be calculated and contrasted.

Progress rate maximization, q_{sd}, is the weight associated to the progress rate, $\dot{s}$, and is added to the cost function with a negative sign to maximize it.

The laptime achieved in every one of the cases is 43.5s, 34.6s, 24.7s and 22.58s respectively. The % of optimal exit flags are 100%, 100%, 100% and 97.2% respectively. And the number of cones hit are 0, 0, 0 and 4 respectively (see Appendix A.1.1 for the data figures). As it was expected, the laptime is drastically reduced when raising the q_{sd} value, since the car maximizes the progress. But if the value is abruptly increased, the feasibility of the problem becomes more difficult and some exit flags are set to 0, this produces that if that exit flag is obtained in a crucial part of the circuit where the car goes fast, it is traduced into hitting some cones and therefore loosing more time than what is gained.

Control softening weights, $R_{D/\delta}$, are the weights associated to the controls' slew rate, and are destined to penalize the abrupt controls changes.

Regarding to R_D as the data in Appendix A.1.2 shows, a reduced value allows the controller to abruptly change the $\dot{D}$ value, therefore obtaining sharp changes in the D control. This produces a fastest laptime, 22.42s, but also more aggressive driving and some non-optimum exit flags. If the value is increased, the $\dot{D}$ value decreases and the D value is not subject to abrupt changes, achieving a more soft driving but longest laptimes. Therefore an equilibrum between the two is needed.

For the R_δ, a different conclusion in substracted from the data in Appendix A.1.3. Since the steering does not affect directly to the velocity of the car and the important factor for path following are the steering bounds, there are no changes in the laptime. Nevertheless an interesting change is seen as the R_δ value is increased, and it's that since the abrupt changes are penalized, the $\dot{\delta}$ value is softened and therefore obtaining the same curve, needed for path following, but in a much more soft fashion.

Lateral deviation weight, q_n, is the weight of the lateral deviation from the reference. This weight is destined to forcing the car to follow the track reference since the car has proven to deviate to maximize the progress rate and if the reference trajectory is optimized, the car can exit the track or hit some cones. With a low q_n value, as seen in the data in the Appendix A.1.4, the car is seen to follow the reference path but with high deviation from it, the perpendicular distance is n is increased. Since the trajectory is already optimized, the car must follow it and not deviate or cut in the corners, this results in hitting some cones in the part of the track where the reference trajectory is closest to them. The velocity profile is not affected, and the optimum flags neither.

Torque vectoring momentum weight, R_{Mtv}, is the weight of the torque vectoring control. This weight traduces in letting the controller use the additional torque vectoring moment in the track. This control and weight analysis are limited by the *FSSIM* simulation, since the low level

controllers used in the car are not implemented in it. Nevertheless, it is also studied to get to know the behaviour better and when it's time to test it on the car, have already a basis on how this control works. As it is seen in the data in Appendix A.1.5, the first values of the R_{Mtv} weight do not let the controller use the extra momentum generated by the TV, so lower values of the weight are needed. But as it is shown in the data graphs, a very low value traduces in too high momentum values and a drastic increase in non optimum exit flags with no effect on laptime.

2.5.2 Forces ellipse data analysis

The forces ellipse constraint is very important for controlling the car behaviour on track. It is destined to, when a stable and robust parameters for the MPC are found, control how the car is going to take the turn and accelerate in the straights. Depending on the track, if it is smaller and with lots of hairpins or if it is big with large straights, the car must adjust its driving style. Therefore, a thorough analysis of this constraint and its parameters must be done in order to maximize performance at the competitions. The parameters correspond to ρ_{long} and λ, the tire ratio and the maximal lateral force. The data corresponding to this contraint analysis is attached in the Appendix A.2.

As seen in the data anlysis, a reference ellipse in red is added, since both of the parameters are in both sides of the inequality, only a relation between orders is needed i.e. if $\lambda = 1$, $\rho_{long} = 0.5$ must be of the same order. A lower ρ_{long} value lets the car have more longitudinal force, achieving a maximum of 4 $\frac{m}{s^2}$ and obtaining faster laptimes and more aggressive driving. On the other hand, a ρ_{long} higher than λ, traduces in a more controlled and safe driving, but longer laptimes. Therefore an equilibrium between both must be achieved.

2.5.3 Cartesian MPC comparison

In order to quantify and justify the implementation and use of the controller stated in this work, a thorough comparison between the last season's controller and the new implementation is issued. Since the car will not be ready to test on real track, the same procedure as in the previous test in relized. Both controllers will be tested in the same conditions, since the old MPC is not able to follow the newly generated optimized trajectory generated by the TRO, the previous year's midline calculation algorithm will be used, to be in equal conditions. Both controllers will be tested in three different circuits, corresponding to three different FS competitions of the 2019 year, this tracks are: FS Germany 2019, FS Spain 2019 and FS Italy 2019.

FS Germany 2019, the data regarding to this test is attached in Appendix A.3.1. As seen in the data, this track is in which a great improvement is seen. The old MPC obtains a laptime of 34.03s and the curvature MPC obtains a laptime of 22.42s. The improvement is major, and much more than in the other tracks, this is due to the length of the track, that is much larger than in the other tracks and the new controller has an awesome performance in long tracks without lots of sharp turns, because of the progress minimization term.

FS Spain 2019, the data of this test is attached in Appendix A.3.2. In the following test, the improvement is also seen, with a decrease of almost 2 seconds in laptime. But since the track has lots of hairpins and harsh turns and it's a short track, is difficult to get a major improvement without optimizing the reference path.

FS Italy 2019, the final comparison test is attached in the Appendix A.3.3. The results are similar to the FS Spain test, with a reduction of approximately 2s in laptime.

2.5.4 Solving time analysis

Finally, the last important matter is the computing time taken by the solver, since this time will be the one in basis to which the command publish rate of the MPC will be chosen. The basis of this test is to evaluate the time taken by the solver for every solution in a lap, calculate the arithmetic average and peaks, and check whereas the actual rate of the controller, which is set to 20 Hz can be raised.

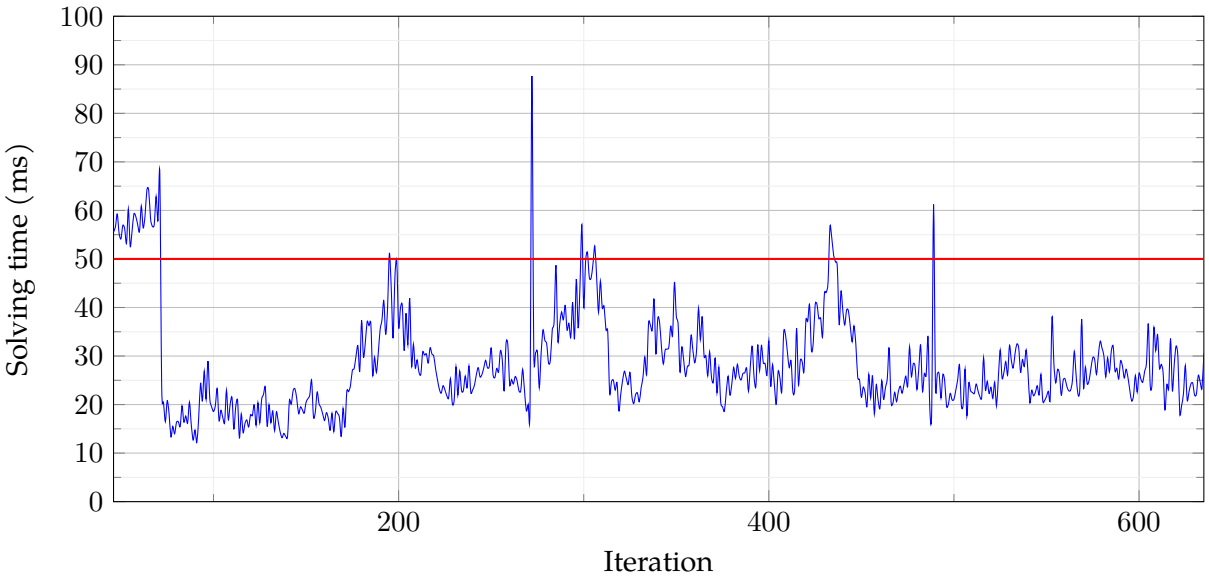


Figure 14: MPC solving time.

As seen in the previous Figure 14, the computing time average of the solver is is of 28.32 ms. The first iterations show a bigger solving time because at the beginning of the track, the car is far from the optimum trajectory and is harder for the controller to find an optimum solution, but once the car joins the trajectory, the exit flags are set to 1 and the computing time decreases. Nevertheless, this fact at the beggining has not a big effect in the average computing time, without the first step the computing time average is 27.15 ms.

With $f = 20$Hz as the controller's rate, the solving time must not exceed 50 ms $T = \frac{1}{f}$, because it would yield repeated solutions, this is not a critic effect, but is not ideal, the rate must be as high as possible but without repeated outputs. The final conclusion is that, seen the computing time has various peaks of 50 ms the rate of the controller cannot be increased.

3 Path planning

3.1 Concepts

For controlling a vehicle or any other system there is a necessary step which is planning. In the case discussed in this work, path planning is essential to give the controller a valid reference to follow. There is also velocity planning, which given a geometric path is possible to obtain a target velocity profile, since the MPC previously defined in Section 2 is designed to maximize velocity, velocity planning is not needed. Therefore, the only planning the controller requires is a geometric curve formulated in curvilinear coordinates. All the basics and research for the following algorithm are extracted and studied from [6] and [19].

3.1.1 Perception data

To procede to planning a set of data is needed to determine the correct path. This information comes from the perception pipeline, and it is composed by two arrays of cones defining the left and right limits of the track respectively. Following the rules of the FS competition, it is stated that the minimum track width is 3m and the maximum data between cones is 5m. This way some assumptions and approximations can be made in the planning algorithm.

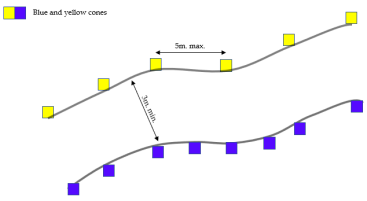


Figure 15: Perception output i.e. track limits.

As it's illustrated in the Figure 15, the two sets of blue and yellow cones define the limits of the track, also the rules' restrictions are added to the illustration.

3.1.2 Spline interpolation

There are multiple ways to solve this mathematical geometrical problem. The chosen solution is midpoint interpolation via splines. The spline interpolation provides flexibility in the formulation to achieve trajectories with n-derivative continuous. In this case, the MPC relies on the path curvature required for path following information. The curvature of an spline or any curve is function of the second derivative. Defining the curve as $r(t)$, the curvature $k(t)$ is,

$$k(t) = \int \left| r''(t) \right| dt$$

Therefore, the trajectory must have 2^{nd} continuous derivative to achieve a continuous curvature.

3.2 Algorithm

The implemented path planning algorithm aims to obtain a continuous, smooth and correct path between all the cones of the track. To obtain smooth, curvature continuous path, is necessary to build the trajectory every iteration that new cones arrive, not to recalculate at a fixed frequency, since that way it is impossible to obtain a curvature continuous path because the track limits change every iteration. Nonetheless, the algorithm must be robust to possible missdetections or perception errors.

3.2.1 Gate calculation

The first step, given the cones that define the left and right track limits, is to filter the cones and only keep the new ones, since the frequency of the track limits algorithm is 40 Hz and most of the iterations there are no new cones. The next step is to calculate the midpoints to interpolate the trajectory with splines. These are calculated via gate midpoints, a gate is defined as a pair of points each one belonging to one track limit.

The gates are calculated by iterating the cones in the track limit with most cones. The corresponding track limit is set as reference. The points A, P, B shown in the Figure 16 are the iterated cone and the previous and next cones. The points O, Q are the other track limit's closest cones to the point P. Using the bisector algorithm, the perpendicular direction $\vec{v}$ and line t to the reference track limit is calculated as follows

$$\vec{v} = \frac{\vec{PA}}{\left|\vec{PA}\right|} + \frac{\vec{PB}}{\left|\vec{PB}\right|}, \quad t : y = \frac{v_y}{v_x}x + (P_y - \frac{v_y}{v_x}) \tag{18}$$

The straight line t is intersected with the $\overline{OQ}$ line. Therefore, the point R corresponding to the non reference track limit point of the gate is obtained.

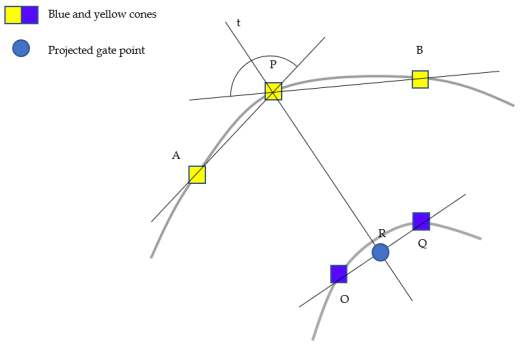


Figure 16: Algorithm graphical explanation.

Once all the gate points have been calculated (P_{left}, P_{right}), the midpoints are evaluated and the points to interpolate the trajectory (MP) are obtained,

$$MP_i = \frac{P_{left,i} + P_{right,i}}{2} \tag{19}$$

3.2.2 4^{th} and 3^{rd} order spline interpolation

The best option when interpolating spatial points are polynomials since they can achieve a desired degree of smoothness, but when the degree of the polynomial increases, the Runge's phenomenon appears i.e. oscillations are seen at the edge of the intervals. To solve this, spline interpolation defines a curve composed by different low degree polynomials, this way the desired smoothness is obtained without having oscillations. Spline interpolation has been widely studied and 3^{rd} degree splines are state-of-the-art, they are the most used and which give best

results, because with 3^{rd} degree polynomials it is possible to achieve 1^{st} and 2^{nd} derivative continuous in all the curve, therefore obtaining a smooth path.

In this work, the aim is to obtain an online calculated, curvature continuous path, so the previous part of the trajectory must be joined with a the newly calculated part. Every 3^{rd} order spline,

$$f_i(t), \;\; 0 < t < a$$

provides 4 degrees of freedom and to join a part of the trajectory with another with a continuous curvature union, the following conditions are needed,

$$\begin{cases} f_i(0) = f_{i-1}(a), \\ f_i(a) = f_{i+1}(0), \\ f_i^{'}(0) = f_{i-1}^{'}(a), \\ f_i^{'}(a) = f_{i+1}^{'}(0), \\ f_i^{''}(0) = f_{i-1}^{''}(a), \end{cases} \tag{20}$$

Therefore, five conditions with four degrees makes the problem infeasible. Consequently for the polynomial of union between trajectories it is needed to increase the degree to 4^{th}. This increases the risk of obtaining oscillations but in the results section it will be seen that the result is as good as a 3^{rd} order spline. Nevertheless 4^{th} degree is only used in the union since in the rest is not necessary.

The splines used in the algorithm are stated as follows,

$$\begin{aligned} &x_1(t) = a_{x,1} + b_{x,1}t + c_{x,1}t^2 + d_{x,1}t^3 + e_{x,1}t^4, \\ &y_1(t) = a_{y,1} + b_{y,1}t + c_{y,1}t^2 + d_{y,1}t^3 + e_{y,1}t^4, \\ &x_i(t) = a_{x,i} + b_{x,i}t + c_{x,i}t^2 + d_{x,i}t^3, \; i > 1, \\ &y_i(t) = a_{y,i} + b_{y,i}t + c_{y,i}t^2 + d_{y,i}t^3, \; i > 1, \\ &0 < t < D_i, \\ &1 < i <= N, \end{aligned} \tag{21}$$

The parametrization interval is defined from 0 to D_i, the distance between points and N is the number of interpolated points. From this point on, only the x coordinate will be studied, since the process is analog with the y coordinate.

To find the coefficients corresponding to every spline and consequently, the trajectory, a linear equation system is formulated. The formulation is later transformed to its matrix form to help the solver find the solution to the system.

The definition of the midpoints in (19) is recalled, $MP_i = (MP_{i,x}, MP_{i,y})$, the last point of the trajectory is defined as $P_{last} = (P_{last,x}, P_{last,y})$, and the last spline of the trajectory is stated as $x_{prev}(t)$ and its first and second derivatives as $x_{prev}^{'}$ and $x_{prev}^{''}$,

The 4^{th} order spline, the first of the new trajectory is formulated as follows,

$$\begin{cases} x_1(0) = P_{last,x}, & a_{x,1} = P_{last,x}, \\ x_1^{'}(0) = x_{prev}^{'}, & b_{x,1} = x_{prev}^{'}, \\ x_1^{''}(0) = x_{prev}^{''}, & 2c_{x,1} = x_{prev}^{''}, \\ x_1(D_1) = x_2(0), & a_{x,1} + b_{x,1}D_1 + c_{x,1}(D_1)^2 + d_{x,1}(D_1)^3 + e_{x,1}(D_1)^4 = a_{x,2}, \\ x_1^{'}(D_1) = x_2^{'}(0), & b_{x,1} + 2c_{x,1}D_1 + 3d_{x,1}(D_1)^2 + 4e_{x,1}(D_1)^3 = b_{x,2}, \end{cases} \tag{22}$$

The 3^{rd} order splines, corresponding to the middle of the new trajectory are formulated as follows,

$$\begin{cases} x_i^{''}(0) = x_{i-1}^{''}(D_{i-1}), & c_{x,i} = c_{x,i-1} + 3d_{x,i-1}D_{i-1} + 6e_{x,i-1}(D_{i-1})^2, \quad \textbf{if } i = 2 \\ x_i^{''}(0) = x_{i-1}^{''}(D_{i-1}), & c_{x,i} = c_{x,i-1} + 3d_{x,i-1}D_{i-1}, \quad \textbf{otherwise} \\ x_i(0) = MP_{i,x}, & a_{x,i} = MP_{i,x} \\ x_i(D_i) = x_{i+1}(0), & a_{x,i} + b_{x,i}D_i + c_{x,i}(D_i)^2 + d_{x,i}(D_i)^3 = a_{x,i+1} \\ x_i^{'}(D_i) = x_{i+1}^{'}(0), & b_{x,i} + 2c_{x,i}D_i + 3d_{x,i}(D_i)^2 = b_{x,i+1} \\ 1 < i < N, \end{cases} \tag{23}$$

The final 3^{rd} order spline has a different formulation as follows,

$$\begin{cases} x_N^{''}(0) = x_{N-1}^{''}(D_{N-1}), & c_{x,N} = c_{x,N-1} + 3d_{x,N-1}D_{N-1} \\ x_N(0) = MP_{N-1,x}, & a_{x,N} = MP_{N-1,x} \\ x_N(D_N) = MP_{N,x}, & a_{x,N} + b_{x,N}D_N + c_{x,N}(D_N)^2 + d_{x,N}(D_N)^3 = MP_{N,x} \\ x_N^{'}(D_N) = \theta_{last,x}, & b_{x,N} + 2c_{x,N}D_N + 3d_{x,N}(D_N)^2 = \theta_{last,x} \end{cases} \tag{24}$$

where θ is the perpendicular gate heading, $\theta_{last} = (\theta_{last,x}, \theta_{last,y})$ corresponds to the perpendicular parametrized heading of the last gate, so that the spline ends in perpendicular direction to the last gate,

$$\frac{dx}{dy} = \tan(\theta) \;\Rightarrow\; \theta_{last,x} = \sqrt{\frac{1}{1+\tan(\theta)^2}}, \quad \theta_{last,y} = \sqrt{\frac{1}{1+\frac{1}{\tan(\theta)^2}}}, \tag{25}$$

Once all the problem and splines equation have been formulated, the system of equations is re-formulated in the matrix form. In this way, the matrix can be edited dynamically depending on the points that are received from the perception pipeline and also the solution is much easier.

The equations are rearranged and M_{eqs} equations matrix is obtained,

$$M_{eqs} = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & & & & & & \vdots \\
0 & 0 & 2 & 0 & 0 & 0 & & & & & & \vdots \\
1 & D_1 & (D_1)^2 & (D_1)^3 & (D_1)^4 & -1 & 0 & 0 & & & & \vdots \\
0 & 1 & 2D_1 & 3(D_1)^2 & 3(D_1)^3 & 0 & -1 & 0 & & & & \vdots \\
0 & 0 & -(D_1) & -3(D_1)^2 & -6(D_1)^3 & 0 & 0 & 1 & & & & \vdots \\
\vdots & & & & & 1 & 0 & 0 & 0 & & & \vdots \\
\vdots & & & & & 1 & D_2 & (D_2)^2 & (D_2)^3 & -1 & 0 & 0 & \vdots \\
\vdots & & & & & 0 & 1 & 2D_2 & 3(D_2)^2 & 0 & -1 & 0 & \vdots \\
\vdots & & & & & 0 & 0 & -1 & -3D_2 & 0 & 0 & 1 & \vdots \\
\vdots & & & & & & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\
\vdots & & & & & & & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\
\vdots & & & & & & & & \ddots & \ddots & \ddots & 1 & 0 & 0 & 0 \\
\vdots & & & & & & & & & & \ddots & 1 & D_N & (D_N)^2 & (D_N)^3 \\
0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & 1 & 2D_N & 3(D_N)^2
\end{pmatrix} \tag{26}$$

The mathematical unknown coefficient vector C_{coefs} and the independent term vector B are defined as follows,

$$C_{coefs} = \begin{pmatrix}
a_{x,1} \\ b_{x,1} \\ c_{x,1} \\ d_{x,1} \\ e_{x,1} \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ a_{x,N} \\ b_{x,N} \\ c_{x,N} \\ d_{x,N}
\end{pmatrix}, \quad B = \begin{pmatrix}
P_{last,x} \\ x'_{prev} \\ x''_{prev} \\ 0 \\ 0 \\ 0 \\ MP_{i,x} \\ 0 \\ 0 \\ 0 \\ \vdots \\ \vdots \\ MP_{N-1,x} \\ MP_{N,x} \\ \theta_{last,x}
\end{pmatrix} \tag{27}$$

Given the matrixes that form the system, the following matix equation is solved obtaining the splines' coefficients,

$$[\mathbf{M_{eqs}}][\mathbf{C_{coefs}}] = [\mathbf{B}] \tag{28}$$

Once the splines' coefficients are obtained, C_{coefs}, they are evaluated and the new trajectory points' coordinates, $P_{traj,k}$, are obtained.

3.2.3 Curvature radius and $FREE_L, FREE_R$ calculation

Once the trajectory is obtained, since the MPC is formulated in curvilinear coordinates, it is possible to formulate the curve in the same way. The curvature is calculated as the inverse of the curvature radius,

$$R_c = \frac{1}{k(s)} \tag{29}$$

Considering that the analytic formulation of the curvature is known as a set of splines of 3^{rd} and 4^{th} degree, it is possible to calculate the curvature radius analytically, thus the curvature is given by

$$R_c = \frac{\left[\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2\right]}{\left|\frac{dx}{dt}\frac{d^2y}{dt^2} - \frac{dy}{dt}\frac{d^2x}{dt^2}\right|} \tag{30}$$

Finally, for the MPC's track constraints, the distance from the centerline to the track limits must also be computed by the planner. This distance is estimated with the following algorithm,

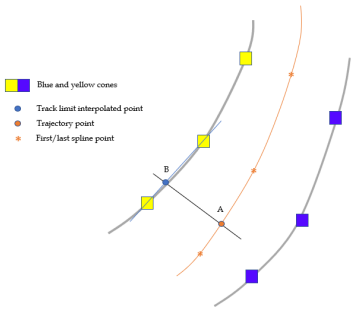


Figure 17: Graphical explanation of free distances algorithm.

Being $t_i(s) \in [0, D_i]$ an equispaced vector in the spline interval, it is used to obtain the desired discretization of the curve. Using the same vector t, the track limits are linearly interpolated and every point of the track corresponds to a point in the track limit i.e. points A, B. Then, the modulus of the vector $\overrightarrow{AB}$ is calculated, therefore obtaining the desired distance. The algorithm is analog for the other side of the track

$$\begin{aligned} &A = P_{traj,k}, \quad B = t_i(k)P_{left,i+1} + (D_i - t_i(k)P_{left,i}) \\ &\left|\overrightarrow{AB}\right| = FREE_L(k) \end{aligned} \tag{31}$$

3.2.4 Recalculation

Finally, the algorithm must be robust to change in the track limits, and if every iteration all the new cones are taken into account, it is vulnerable to perception missdetections or track limits mistakes. Consequently, a flag is published from the perception pipeline every time that the track limits are corrected. When this flag is received, the trajectory is erased and calculated from scratch with the new track limits. Then, as the car advances and new track limits are received, the new parts of the trajectory are added until a new correction flag is published.

3.3 Implementation

3.3.1 Algorithm pseudocode

Algorithm 2 Planner

$P\_t,\ r,\ fL,\ fR \leftarrow 0$ {Initialize variables}

$state \leftarrow initial\_state$ {Initialize car state}

while $ros :: ok$ **do**

  $state \leftarrow$ **ros::callback_state**() {State callback}

  $cones, flag \leftarrow$ **ros::callback_tracklimits()** {Perception callback}

    if (!flag) **then**

      c $\leftarrow planner :: get\_data(cones)$ {Get only new cones}

    else

      c $\leftarrow cones$ {Get all cones}

    end if

    P_r, **P_l** $\leftarrow planner :: get\_gates()$ {Claculate gates}

    s $\leftarrow planner :: build\_splines()$ {Splines' coefficients}

    P_t, **r**, **fL**, **fR** $\leftarrow planner :: evaluate\_points()$ {Calculate MPC requirements}

    ros::msg $\leftarrow planner :: planning\_curv$ {Prepare ros message}

    $ros :: publish(ros :: msg)$ {Publish message}

  $ros :: spin()$ {Spin ROS thread to publish at callback frequency}

end while

3.4 Results and data analysis

3.4.1 Trajectory

To study and analyze the online curvature and trajectory calculated by the planning algorithm stated before in this work, the same test has been conducted in three different tracks to evaluate its results and to have diversity in conditions of the track to check the algorithm performance. As in the MPC tests, the planner is tested in the tracks available in simulation.

In the first place, the planning algorithm is tested in the **FSG 2019 track**. The results of this test are attached in Appendix B.1. As seen in the appendix, the resulting trajectory is an smooth trajectory corresponding to the midline of the cones seen in red. The track is not seen to have lots of complications, since it is a long track with well placed cones.

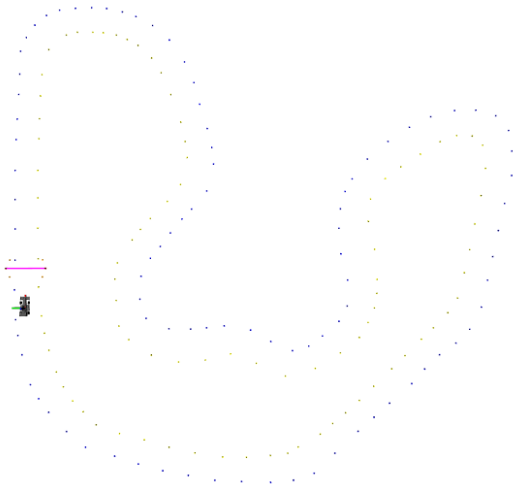


Figure 18: RVIZ visualization of FSG 2019 track.

In the second case, the **FSS 2019 track**, is known to be a much more difficult track to plan, since it is smaller and there are two harsh hairpins which make more difficult the trajectory planning. Nonetheless, the resulting trajectory is also a smooth acceptable path for the controller to follow. The results of this test are attached in Appendix B.1.

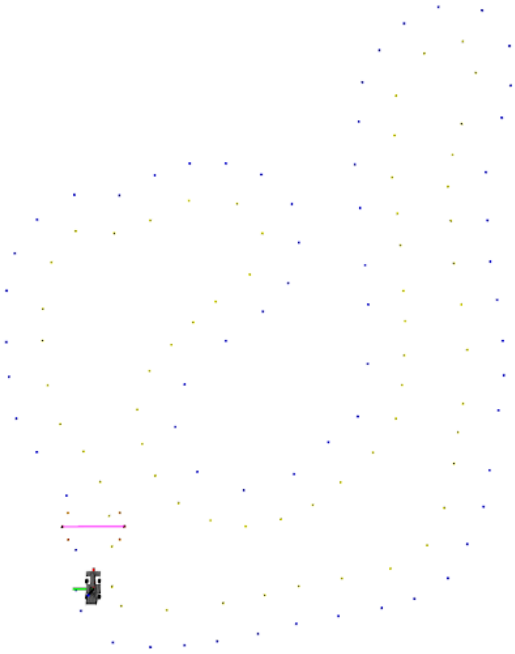


Figure 19: RVIZ visualization of FSS 2019 track.

Finally, the last test is conducted in the **FSI 2019 track**, with a very easy and short circuit to plan, since the turns have big radius and the results are also very good. The results of this test are attached in Appendix B.1.

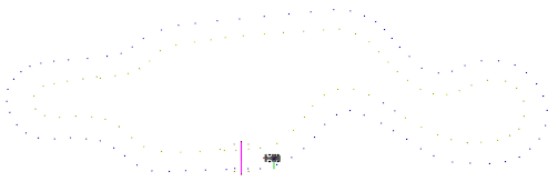


Figure 20: RVIZ visualization of FSI 2019 track.

3.4.2 Curvature

Finally, the most important factor of the new planning algorithm is analyzed. Since the reason for the need of the implementation of a new algorithm is based on the curvature calculation performed by the old planner, a comparison between the curvature calculated by both algorithms is essential. The tests are performed in the same tracks as before, and all the results for both algorithms in all three tracks are attached in Appendix B.2.

As seen in the comparison, since the previous planning was calculated via point interpolation of Bézier curves, this kind of curves are not designed to have continuous curvature with the following curves. Therefore, the result shows a non-continuous curvature with lots of steps and abrupt changes in all three tracks. Therefore, the controller is not able to follow correctly the paths and a big deviation from it is seen in the simulation. Nevertheless, the controller is able to finish the lap without major complication but at a very low pace with unstable controls.

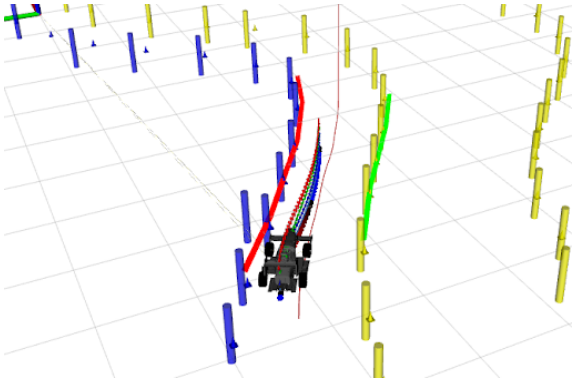


Figure 21: Curvature MPC following the old planner trajectory.

On the other hand, the curvature shown by the new planner, exposes very good results with an smooth curve, also non-continuous. This is due to perception and track limits errors which leads to recalculate the trajectory. Nevertheless, this does not affect the controller and with the output curvature of the new planning algorithm is able to follow smoothly and correctly the trajectory generated.

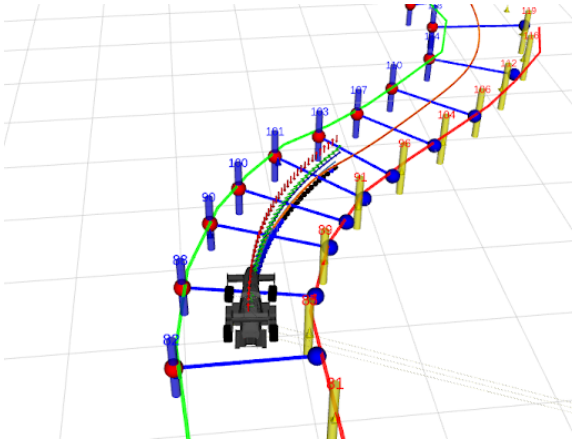


Figure 22: Curvature MPC following the new planner trajectory.

4 Final results

4.1 Autocross

As first final result, the combination of both implemented algorithms stated in this work, is evaluated via the Autocross event. In the autocross event, given the track limits from the perception pipeline, the planner calculates the trajectory through these limits and the controller is set to follow this reference path. To compare the final results with the performance of the car on track, the laptime and scoring will be compared to the results from the 2019 FSG competition that are available in their website. The following velocity profile shows the laptime and velocities achieved in the FSG 2019 track (note that the initial velocity is not 0 because the starting measuring point is 5m from where the car starts, as in the competition):

Car	City/University	Run 1(s)	DNF	Cones	Off course	DQ	Corrected time	Scoring
419	Karlsruhe KIT	28.186	0	0	0	0	28.126	100
433	Zürich ETH	31.158	0	0	0	0	31.158	90.89
478	Hamburg TU	65.356	0	3	0	0	65.356	4.5
489	Tallinn TU UAS	45.835	0	12	0	0	28.126	4.5

Source: FSG Results [13]
Table 5: FSG 2019 autocross scoring chart.

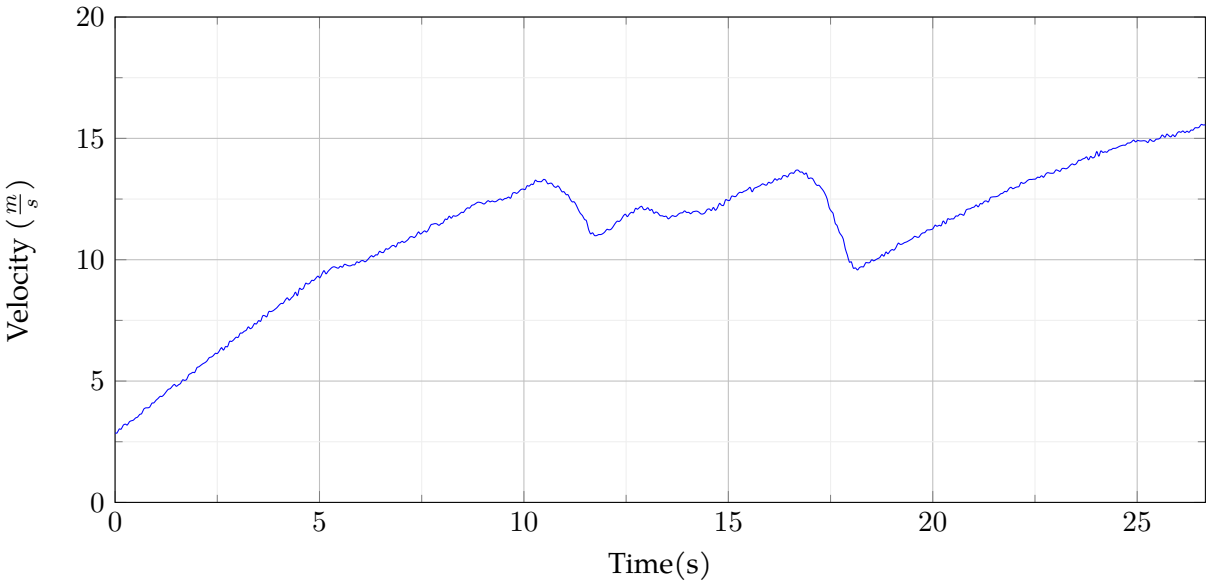


Figure 23: FSG 2019 simulation autocross velocity profile.

The laptime achieved by the control system is **26.66 s**. It is fair to say that this laptime is achieved in simulation in an ideal environment, but it is close enough to validate the controller and planner to be ready to test it on the real car.

4.2 Trackdrive

To validate and test the limits of the controller, it is validated also with the simulated Trackdrive event, consisting on 10 laps with the same track as the Autocross. The controller is tested with the TRO algorithm to optimize trajectory and obtain the best and fastest laptimes possible.

Car	City/University	Time(s)	Cones	Penalty time(s)	Completed laps	Scoring
419	Karlsruhe KIT	244.9	2	0	10	200
433	Zürich ETH	226.31	1	30	10	189.07
486	Augsburg UAS	313.96	2	0	10	134.84
485	Delft TU	706.09	2	0	9	4.5

Source: FSG Results [13]
Table 6: FSG 2019 trackdrive scoring chart.

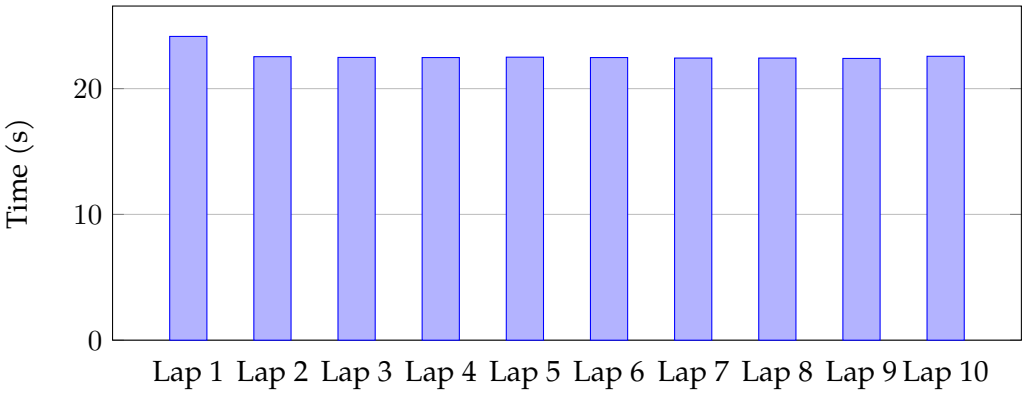


Figure 24: FSG 2019 simulation trackdrive laptimes.

The final achieved time in the simulation trackdrive is **226.44 s**. As in the autocross case, in an ideal simulation environment. But, in this case, the controller is completely validated alongside the TRO for trackdrive.

.

5 Budget

5.1 Personal cost

The personal cost is divided in two groups. The first one corresponding to the minimum wage of a junior engineer, since the is developed by a newly graduated engineer, and the cost of a degree thesis tutor to guide and assist the junior.

Details	€/h	h	Amount (€)
Minimum wage for junior engineer	10	625	6250
Degree thesis tutor	30	40	1200
Total Personal Cost			7450

Table 7: Personal cost chart.

5.2 Licensing cost

In the second place, the most important part of the cost of the project is the licensing cost. Since the solving of NLP is a very limiting matter, a very powerful and specialized solver is needed, this is the case of FORCESPro. As most of the software licenses, such as Matlab, the cost of the licensing is very high, nevertheless, since *Embotech* is an sponsor of BCNeMotorsport, the final cost of the licensing is 0.

Details	Q	€/unit	Amount (€)
FORCES PRO Engineering Node 360-day subscription	3	2947,28	8841,84
FORCES PRO SW Testing Node 360-day subscription	2	982,43	1964,86
FORCES PRO HW Testing Node 360-day subscription	3	11789,13	35367,39
Embotech-BCNeMotorsport sponsorship agreement			-46174,09
Total Licensing Cost			0

Table 8: Licensing cost chart.

5.3 Material cost

For the material cost of the project, only the essential material will be adressed, since the budget for the perception pipeline, for example, is not a matter to discuss in this work rather than in other papers. So, the budget will adress the necessary HW for developing and running both MPC and planner.

Details	Q	€/unit	Amount (€)
Vectornav VN-300	1	4000	4000
Processing Unit	1	1200	1200
GNU x86-64 workstation	1	800	800
Vectornav-BCNeMotorsport sponsorship agreement			-4000
Total Material Cost			2000

Table 9: Material cost chart.

5.4 Energetic cost

In the energetic cost subsection, the cost of work environment and devices' power consumption.

Details	€/kWh	kW	h	Amount (€)
Electricity	0.4	0.8	625	200
Natural Gas	0.13	2	125	32.5
Total Energetic Cost				232.5

Table 10: Energetic cost chart.

5.5 Total cost

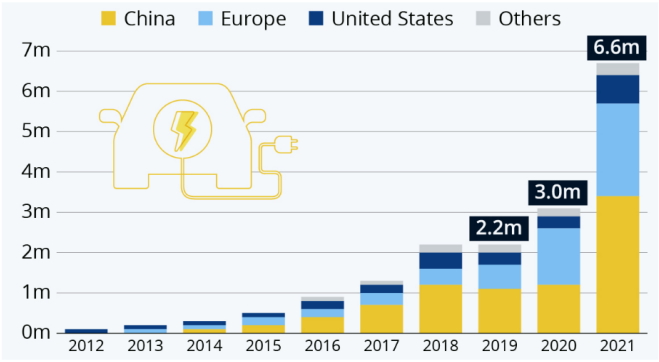
Details	Amount (€)
Personal cost	7450
Licensing cost	0
Material cost	2000
Energetic cost	232.5
Total Cost	9682.5

Table 11: Total cost chart.

6 Environmental impact

The autonomous driving concept and especially the Formula Student are deeply associated to the actual trending electrification of the cars. Most of the big car companies around the world are moving towards a sustainable way of transportation such as the electric cars. As seen in the following figure, the sales of electric cars have drastically increased in the last years, scaling to 6.6 million sales in the major countries.

Car electrification is seen as a way of reducing the negative environmental impact of automobile traffic. One of the main issues that it helps to reduce is the widely known greenhouse effect, produced by the gas emissions of combustion cars, such as stated in articles like [15].



Source: Global Electric Car Sales [14]
Figure 25: Worldwide electric car sales from 2012 to 2021.

Conclusions

To conclude with this work, the first step is to discuss the goals and objectives completion and analysis. The main objectives of the thesis were to implement both control and planning algorithms that fit with the specifications and requirements, to obtain a controller allowing an FS racing car at high speeds, to obtain competitive laptimes and scorings in the FS dynamic events and to improve the results of last season's algorithms.

For the first objective, both algorithms have proven to meet the specifications. Regarding to the computation time and the controller rate, as seen the results in Figure 14, the controller is able to operate at a minimum rate of 20 Hz without repeated data, but is seen to be possible to increment it if the algorithm is executed in more potent devices. For the planning algorithm, the results seen in Appendix B.1 have also shown a feasible and smooth trajectory between the cones for the MPC to follow at any rate, since the calculation is only done once, the publishing rate can be tuned. Moreover, as seen in the results of Appendix B.1, the 2^{nd} derivative continuous trajectory approach has proven to have great results in the curvature of the trajectory, obtaining a curvature continuous path.

As the second objective, as seen in most of the velocity profiles in the Appendix A.1, the controller is able to follow the reference path at high speeds, averaging around 14-15 $\frac{m}{s}$ and maximum velocities of 18-19 $\frac{m}{s}$ in an stable and controlled condition.

For the third objective, as observed in the results Section 4, the laptimes obtained (in simulation) correspond to winning laptimes and scorings in the FSG 2019 competition in both Trackdrive and Autocross events.

Finally, from the results in the Appendixes A.3.1, A.3.2 and B.2, the controller presents a major improvement in laptime and behaviour as the previous season's controller, the cartesian MPC. And the curvature comparison of the old planning algorithm with the one stated in this work, clearly shows the improvement and justifies the much better behaviour and path following of the controller with the new planner.

To add to the analysis, an unmet objective or requirement is the behaviour seen in the Appendix A.1.5 results section. The objective was to obtain the best of the 4-wheel drive with the additional momentum added by the torque vectoring. But, the simulation environment and the vehicle model used in the simulation does not provide the tools to correctly evaluate this behaviour and this control and the torque vectoring momentum control has not been able to study and analyze correctly.

Future work

Regarding to the future work to follow up with this thesis, there is plenty of work to do in both disciplines, planning and control.

For the control follow up, the main improvement reside in the vehicle modeling, to keep with the way that has been following the team since its start. At first a kinematic bycicle model MPC was implemented, then the dynamic model improvement was added, including the curvilinear model. The next step was to add the 4-wheel model of the vehicle, which describes much better the dynamic and behaviour of the car in the track. Nevertheless, a big model change comes

with lots of challenges, such as the rising of the solving time of the solver. On other hand, another active field of research is the parameter tuning. The controller has an important number of parameters regarding the dynamics of the car and most of them are uncertain or are not estimated correctly, a parameter estimation or a parameter learning model is a good way to attack this problem.

From the planning point of view, the way to continue is by using optimization. The planner stated in this work is destined to midline calculation for the afterwards MPC optimization, but as in the TRO approach, it is possible to online optmize the curvature of the trajectory and obtain smoother paths which are much better for the controller to follow.

References

[1] Formula Student Germany, *Formula Student Rules 2022*, FSG, (2022) (Version 1.0)
Consulted (2021-2022)

[2] Formula Student Germany, *Formula Student Handbook 2022*, FSG, (2022-01-26) (Version 1.1)
Consulted (2021-2022)

[3] Guillem Torrente, Elia Kaufmann, Philipp Fohn, Davide Scaramuzza, *Data-Driven MPC for Quadrotors*, IEEE ROBOTICS AND AUTOMATION LETTERS, february, 2021
Consulted (2022)

[4] Moustapha Doumiati, Ali Charara, Alessandro Victorino, Daniel Lechner, *Vehicle dynamics estimation using Kalman Filtering*, ISTE and John Wiley & Sons, Inc., (2013)
Consulted (2021-2022)

[5] Hans B. Pacejka, *The magic formula tyre model. Vehicle System Dynamics*, & Bakker, (1992)
Consulted (2021-2022)

[6] Ignasi Mallèn, *Developement of the Global Race Optimizer*, PBE Report, (2020)
Consulted (2021-2022)

[7] Albert Gassol Puigjaner, *Development of the Model Predictive Controller and Simultaneous-Localization-And-Mapping Algorithm of the Control System*, Degree thesis, UPC (2021)
Consulted (2021-2022)

[8] Vicenç Puig, *Model Predictive Control*, Theory slides MUAR, UPC, (2022)
Consulted (2021-2022)

[9] AMZ driverless, *AMZ Driverless: The Full Autonomous Racing System*, ETH Zurich, Switzerland, (2019)
Consulted (2021-2022)

[10] Sirish Srinivasan, Sebastian Nicolas Giles, Alexander Liniger, *A Holistic Motion Planning and Control Solution to Challenge a Professional Racecar Driver*, ETH Zurich, Switzerland, (2021)
Consulted (2021-2022)

[11] Andrés Alvarez, Nico Denner, Zhe Fang, David Fischer..., *The Software Stack That Won the Formula Student Driverless Competition*, Karlsruhe Institut of Technology, Germany, (2022)
Consulted (2022)

[12] AMZ driverless, *Accurate Mapping and Planning for Autonomous Racing*, ETH Zurich, Switzerland, (2020)
Consulted (2021-2022)

[13] *Formula Student Germany Results*, https://www.formulastudent.de/fsg/results/2019/, (2022).

[14] *Global Electric Car Sales*, https://www.statista.com/chart/26845/global-electric-car-sales/,

(2022).

[15] *How eco-friendly are electric cars?*, https://www.bmuv.de/fileadmin/Daten_BMU/Pools/Broschueren/elektroautos_en_bf.pdf, (2022).

[16] W.F. Milliken and D.L. Milliken, *Race car vehicle dynamics*, SAE International, (1994)
`Consulted (2021-2022)`

[17] Sebastian Thrun, Wolfram Burgard, Dieter Fox, *Probabilistic Robotics* (*Intelligent Robotics and Autonomous Agents*), MIT Press, (2005)
`Consulted (2021-2022)`

[18] *ROS Documentation*, https://docs.ros.org/, (2022).

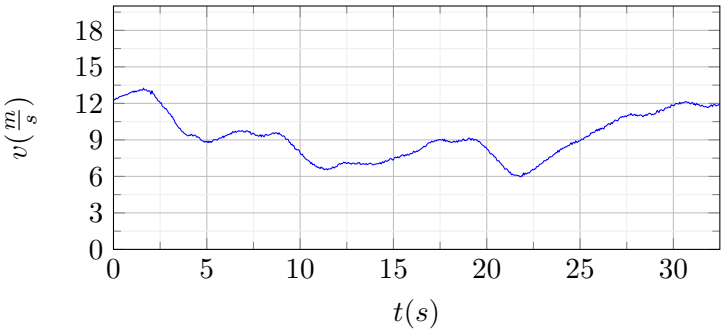
[19] F. Braghin, *Race driver model*), Computers and Structures 86.13-14, (2008)
`Consulted (2021-2022)`

A Appendix: MPC data figures and charts

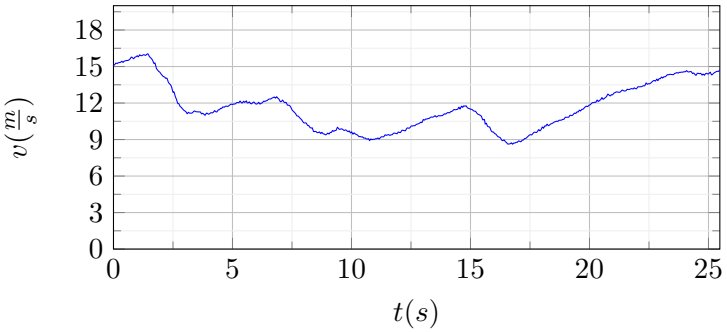
A.1 Cost function

A.1.1 q_{sd}

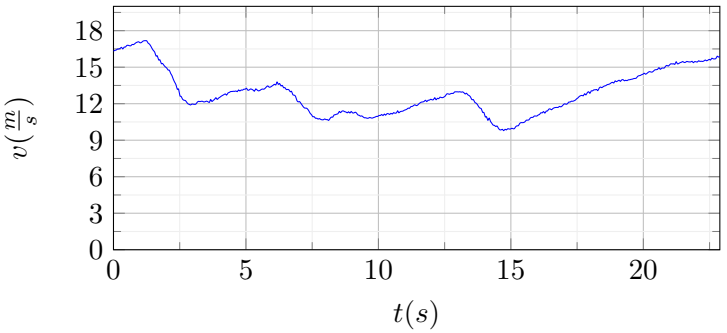
Figure 26: Velocity profiles



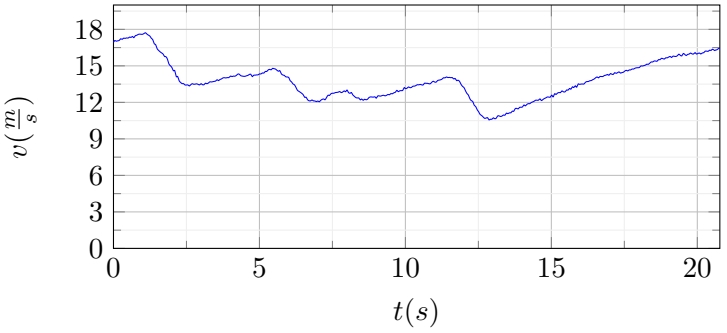
(a) $q_{sd} = 30$



(b) $q_{sd} = 100$

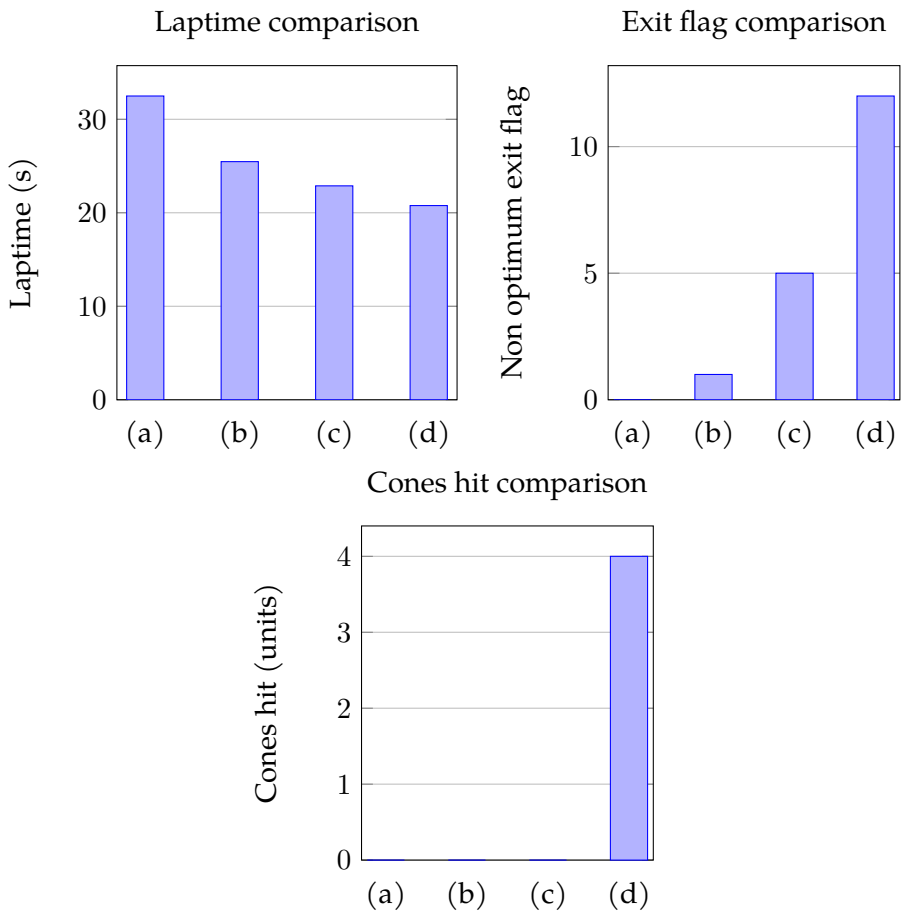


(c) $q_{sd} = 200$



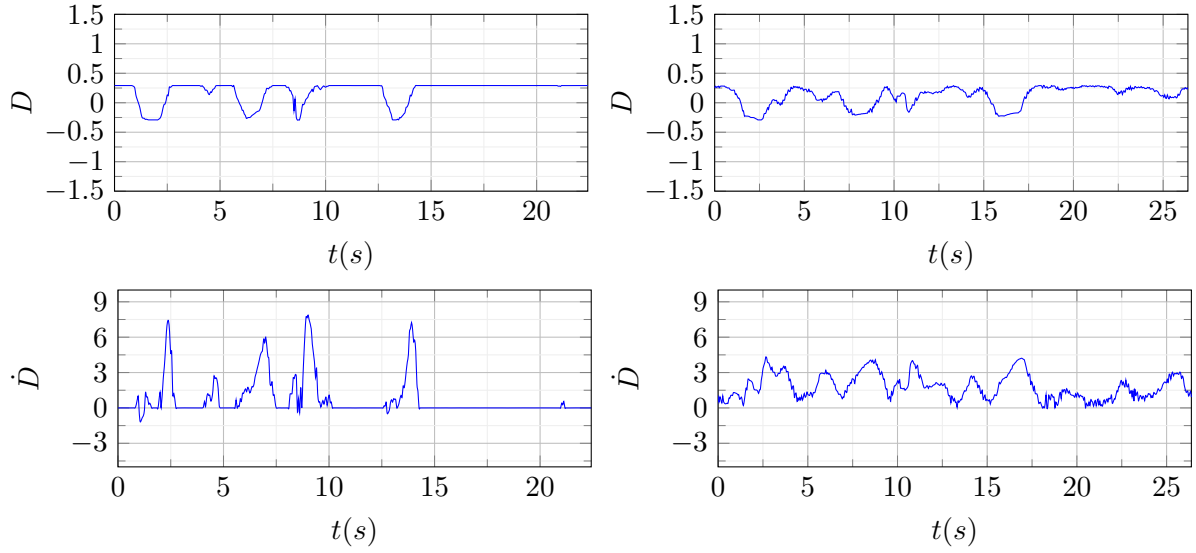
(d) $q_{sd} = 500$

Figure 27: Car behaviour comparison



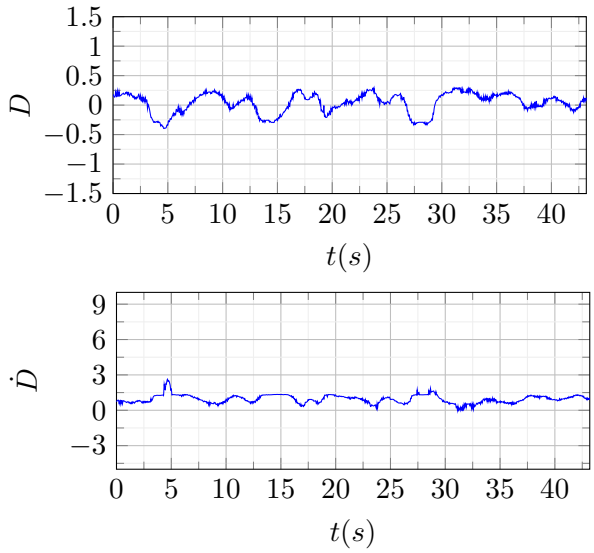
A.1.2 R_D

Figure 28: D and $\dot{D}$ comparison



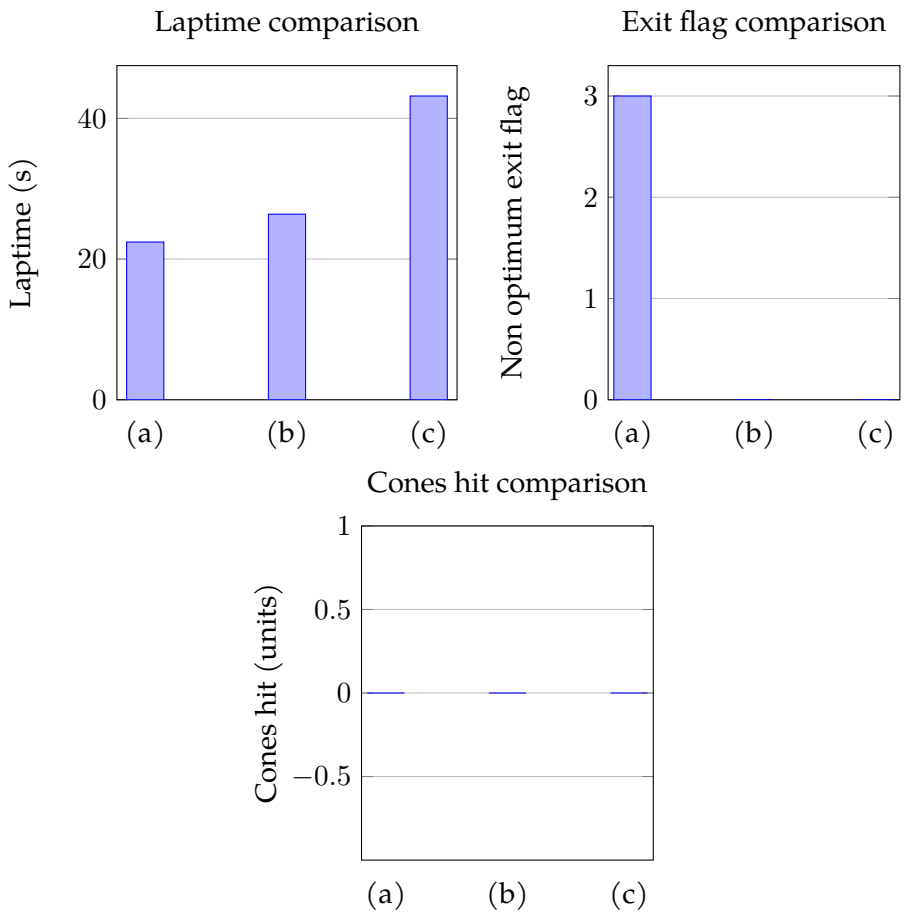
(a) $R_a = 0.1$

(b) $R_a = 1$



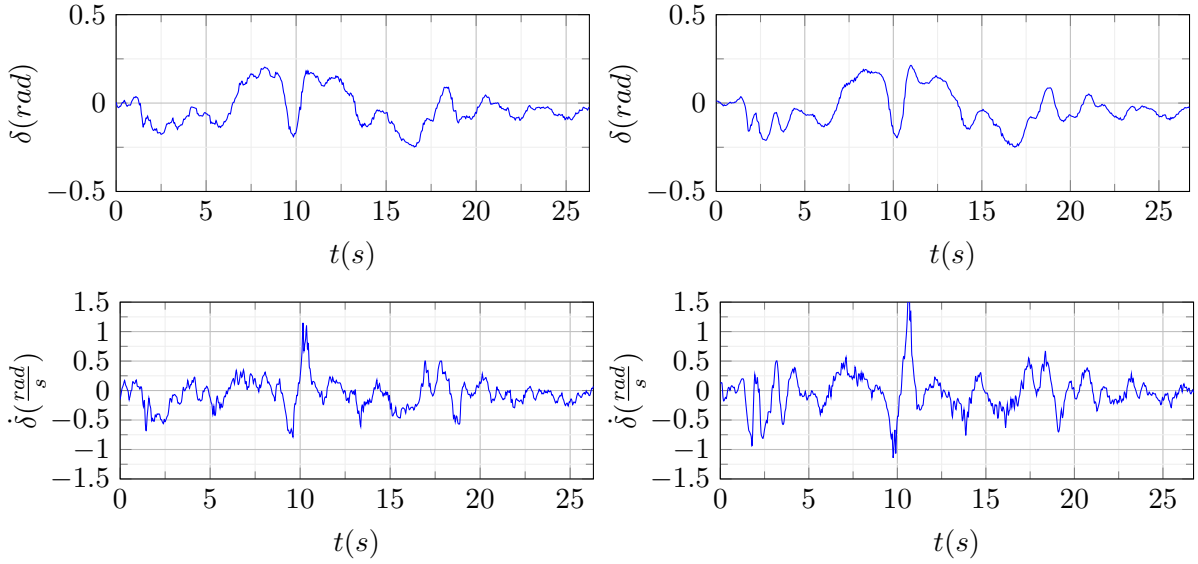
(c) $R_a = 10$

Figure 29: Car behaviour comparison



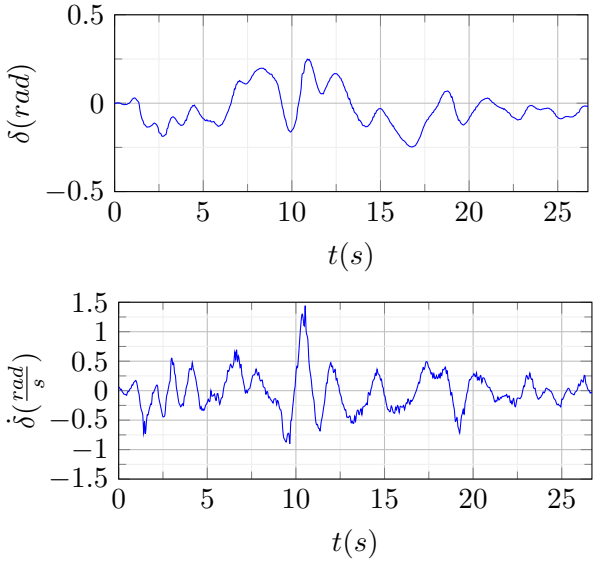
A.1.3 R_δ

Figure 30: δ and $\dot{\delta}$ comparison



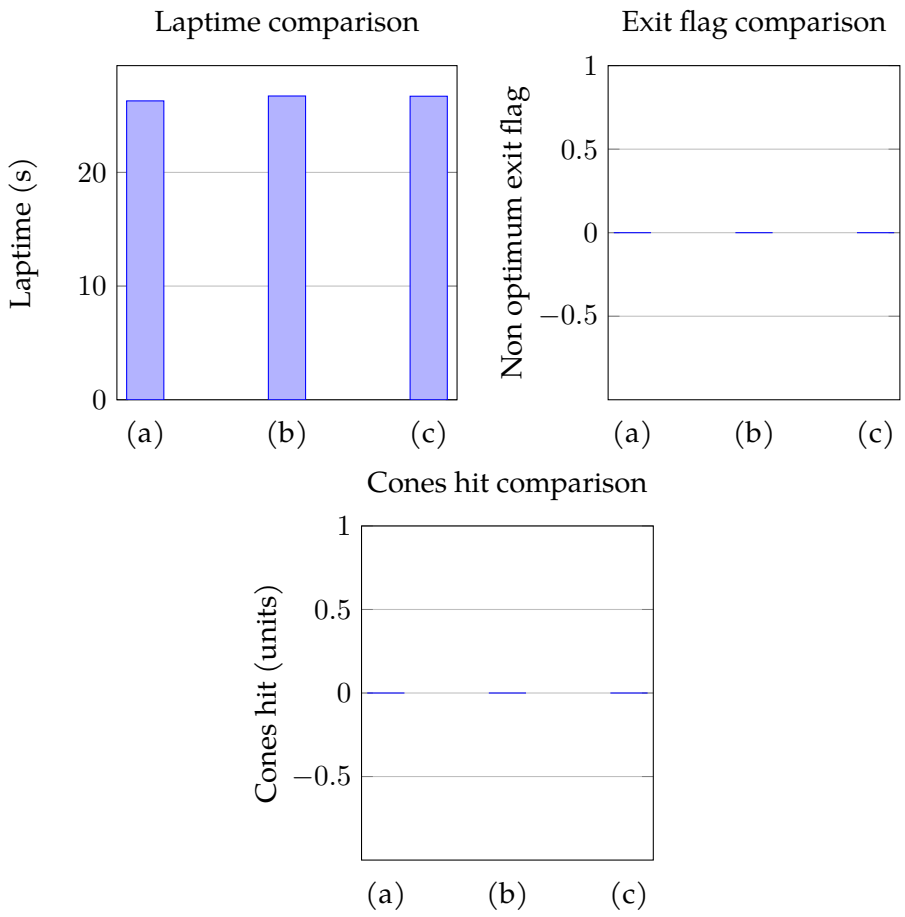
(a) $R_\delta = 0.1$

(b) $R_\delta = 1$



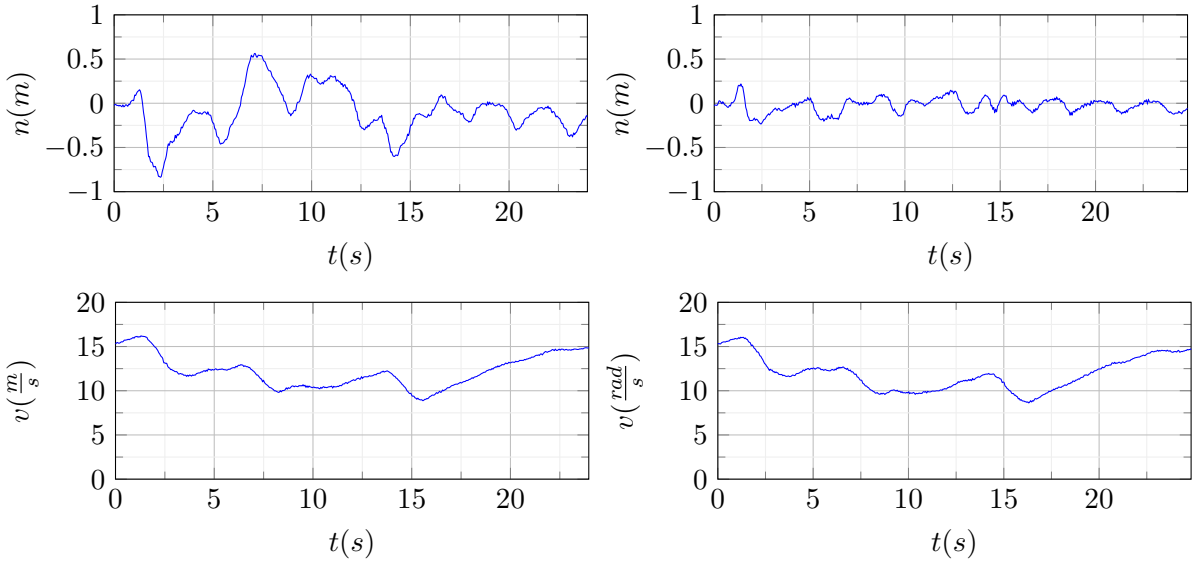
(c) $R_\delta = 10$

Figure 31: Car behaviour comparison



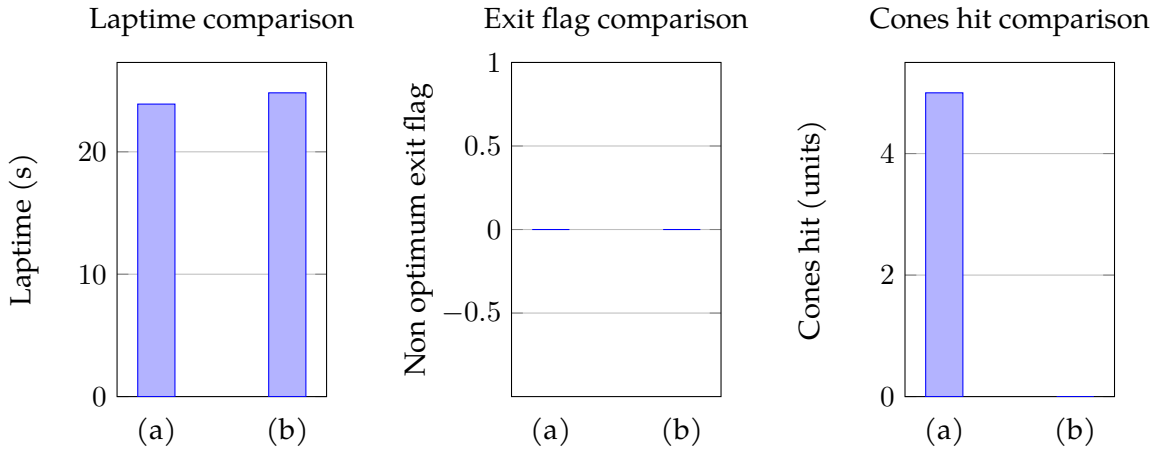
A.1.4 q_n

Figure 32: n and velocity profile comparison



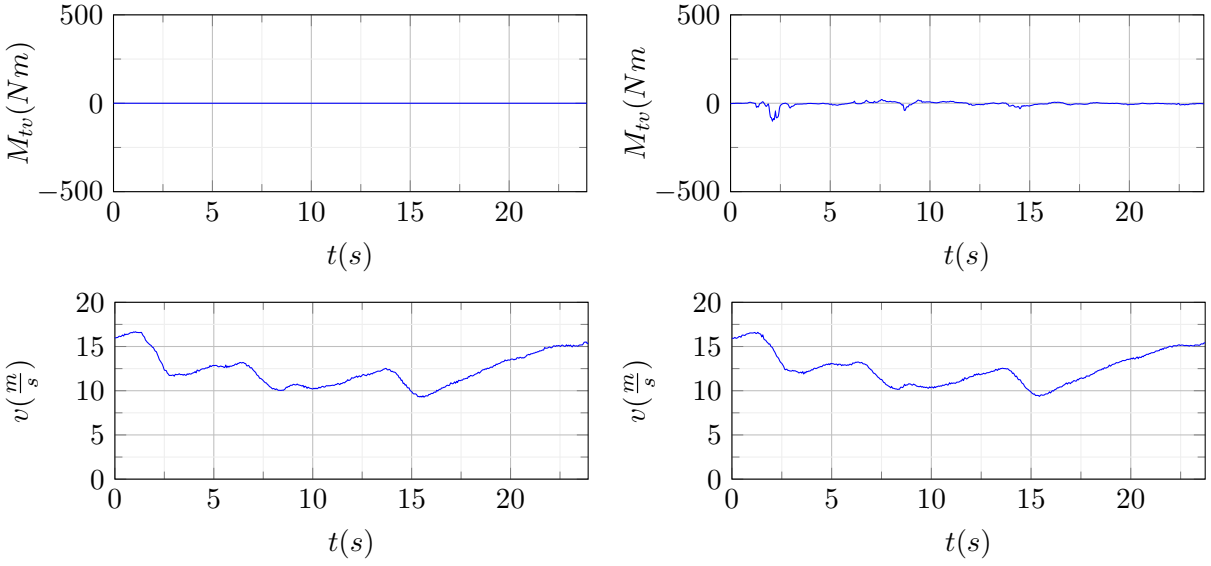
(a) $q_n = 10$ (b) $q_n = 30$

Figure 33: Car behaviour comparison



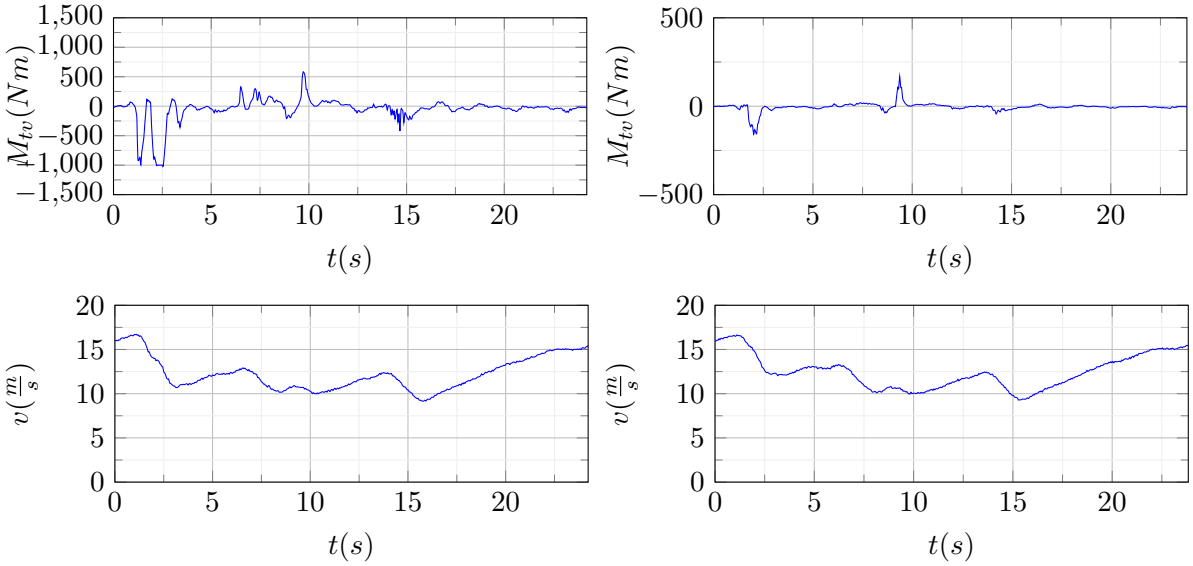
A.1.5 R_{Mtv}

Figure 34: M_{tv} and velocity profile comparison



(a) $R_{Mtv} = 1$

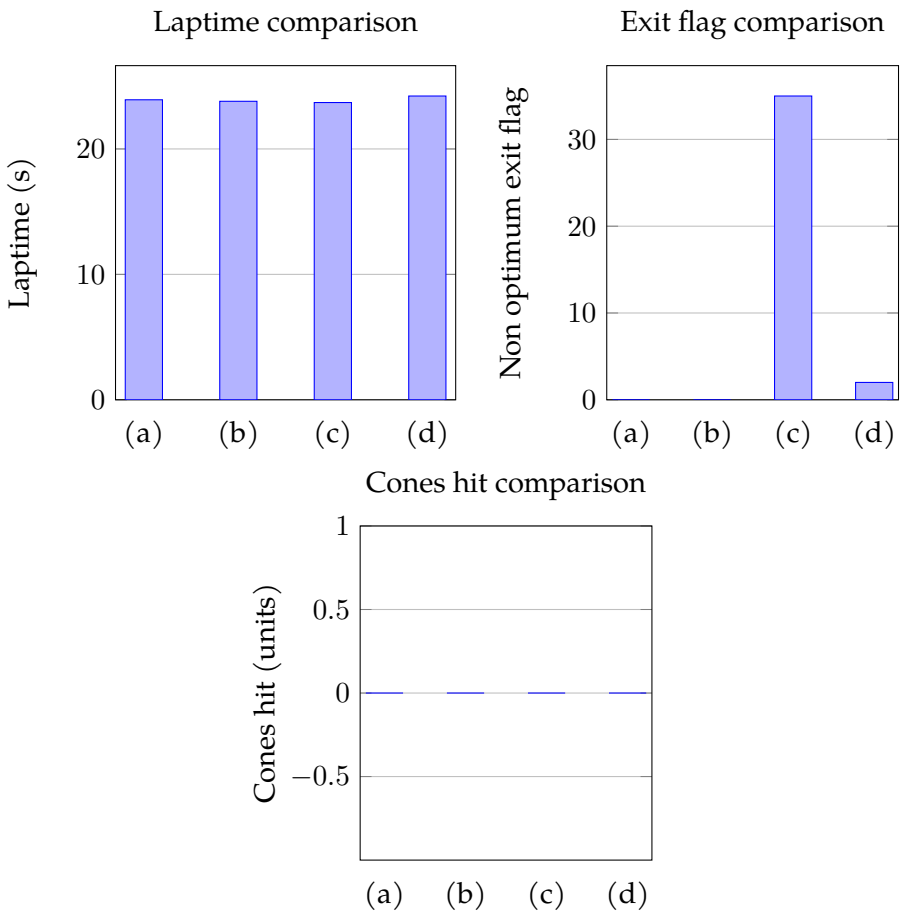
(b) $R_{Mtv} = 110^{-4}$



(c) $R_{Mtv} = 110^{-5}$

(d) $R_{Mtv} = 7.510^{-5}$

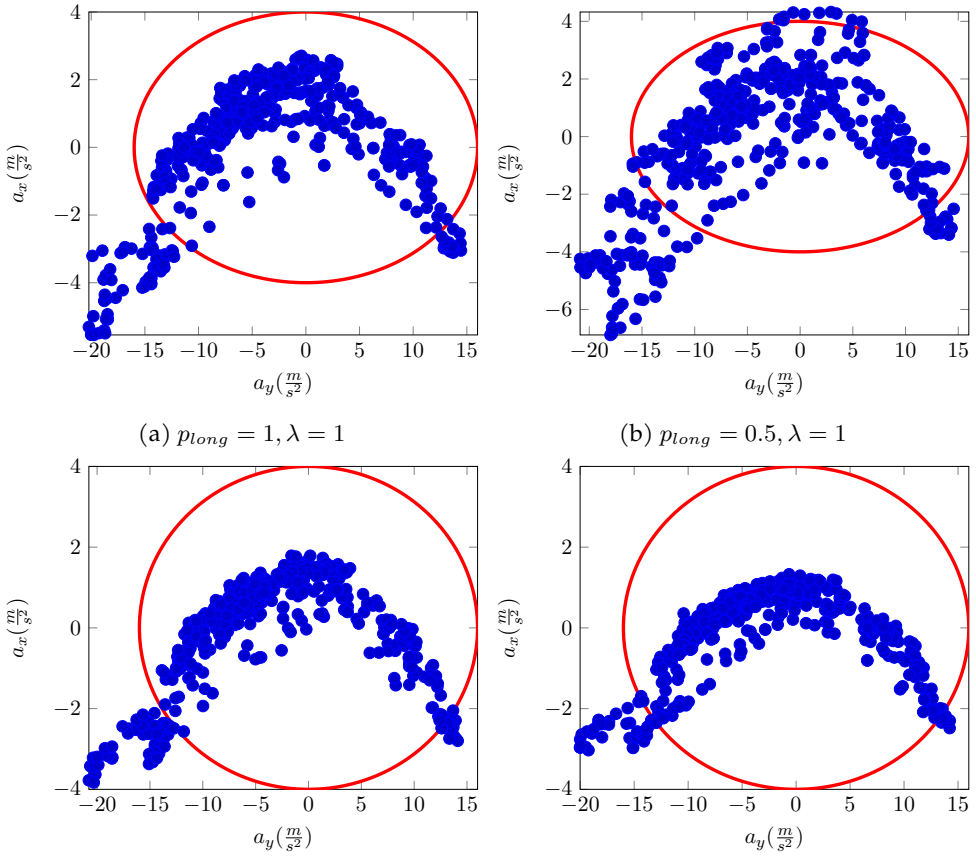
Figure 35: Car behaviour comparison



A.2 Forces ellipse

A.2.1 ρ_{long}, λ

Figure 36: Forces ellipse data



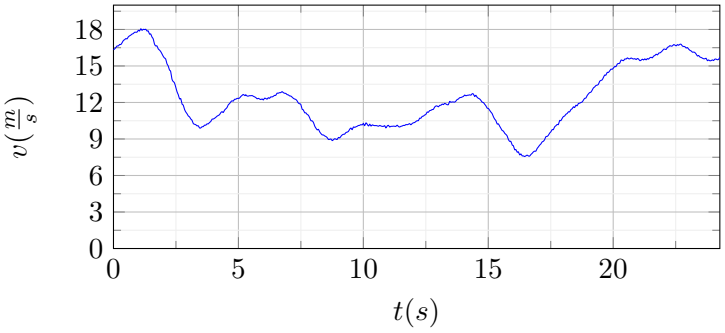
(a) $p_{long} = 1, \lambda = 1$

(b) $p_{long} = 0.5, \lambda = 1$

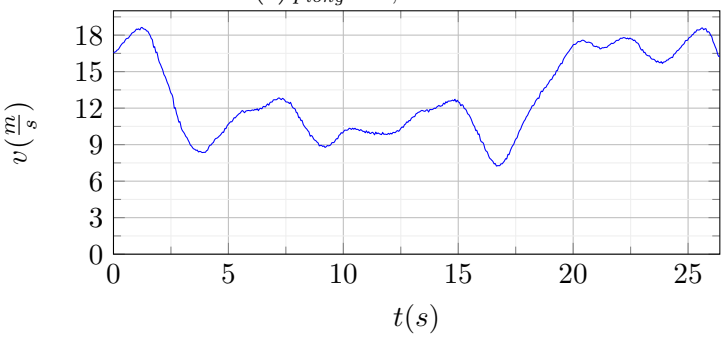
(c) $p_{long} = 1.5, \lambda = 1$

(d) $p_{long} = 2, \lambda = 1$

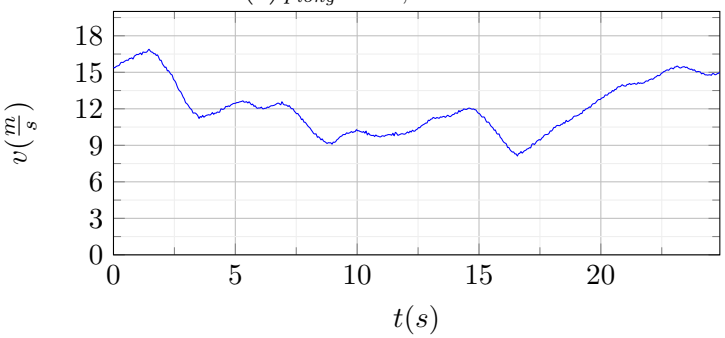
Figure 37: Velocity profiles



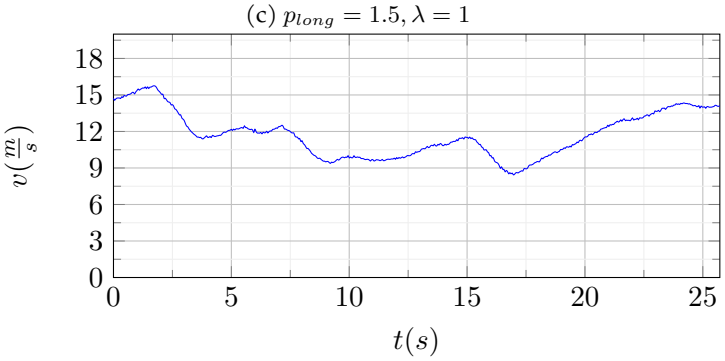
(a) $p_{long} = 1, \lambda = 1$



(b) $p_{long} = 0.5, \lambda = 1$

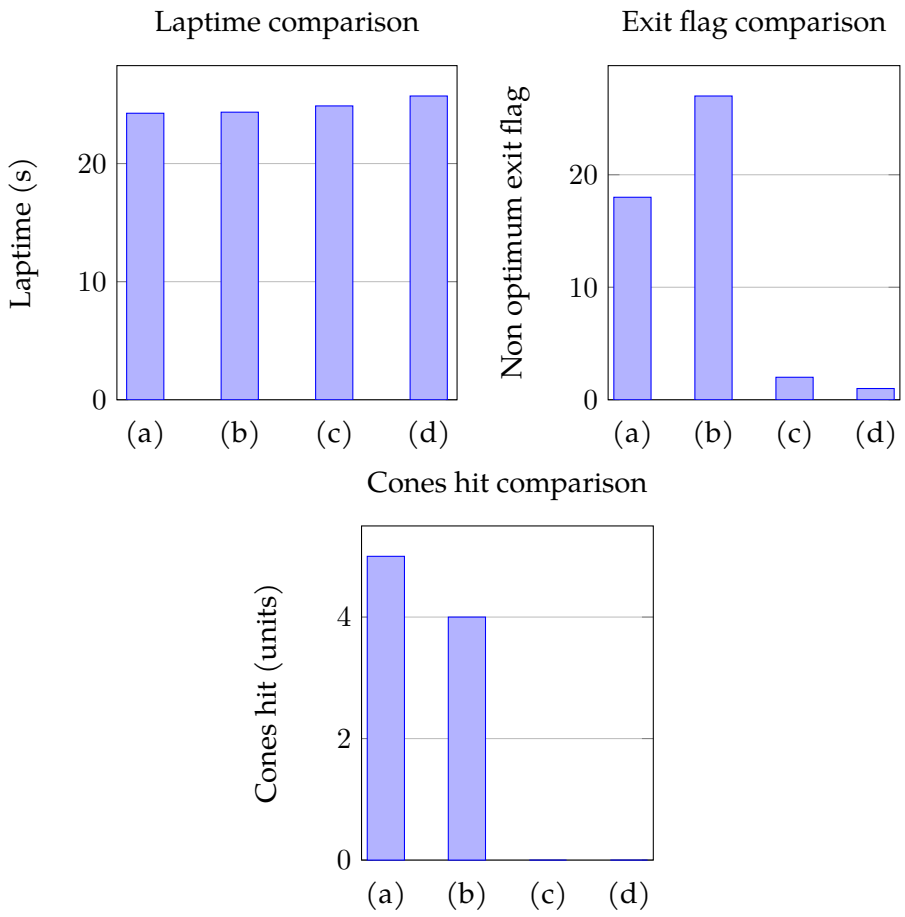


(c) $p_{long} = 1.5, \lambda = 1$



(d) $p_{long} = 2, \lambda = 1$

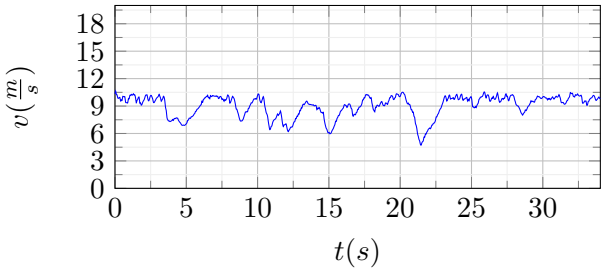
Figure 38: Car behaviour comparison



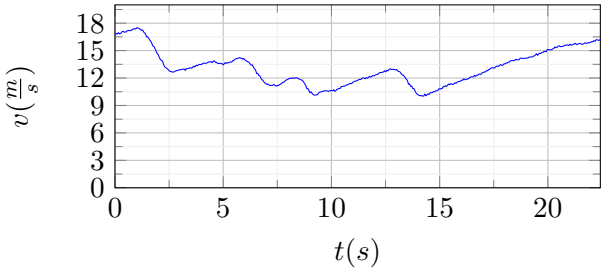
A.3 Cartesian MPC comparison

A.3.1 Germany 2019

Figure 39: MPCs velocity profiles comparison



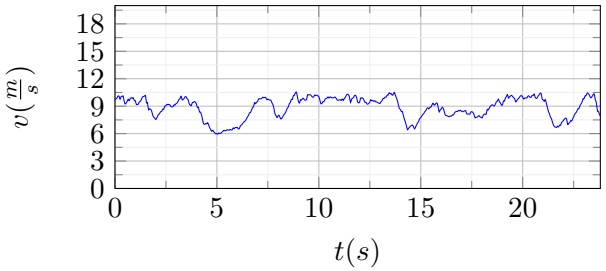
(a) Cartesian MPC season 2020-2021



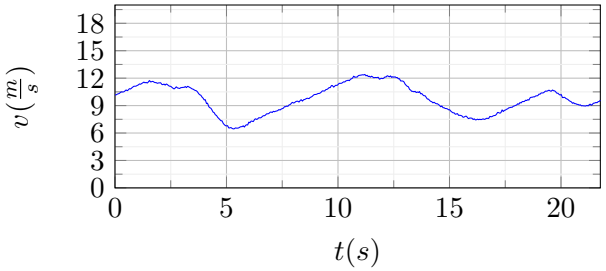
(b) Curvature MPC season 2021-2022

A.3.2 Italy 2019

Figure 40: MPCs velocity profiles comparison



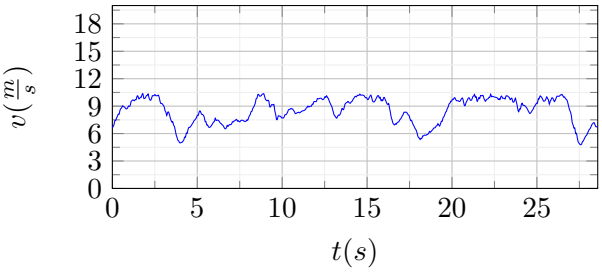
(a) Cartesian MPC season 2020-2021



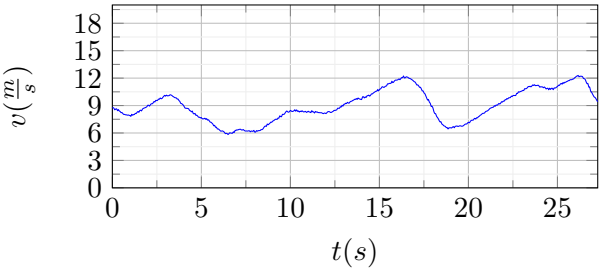
(b) Curvature MPC season 2021-2022

A.3.3 Spain 2019

Figure 41: MPCs velocity profiles comparison



(a) Cartesian MPC season 2020-2021



(b) Curvature MPC season 2021-2022

B Appendix: Planner data figures and charts

B.1 Trajectory

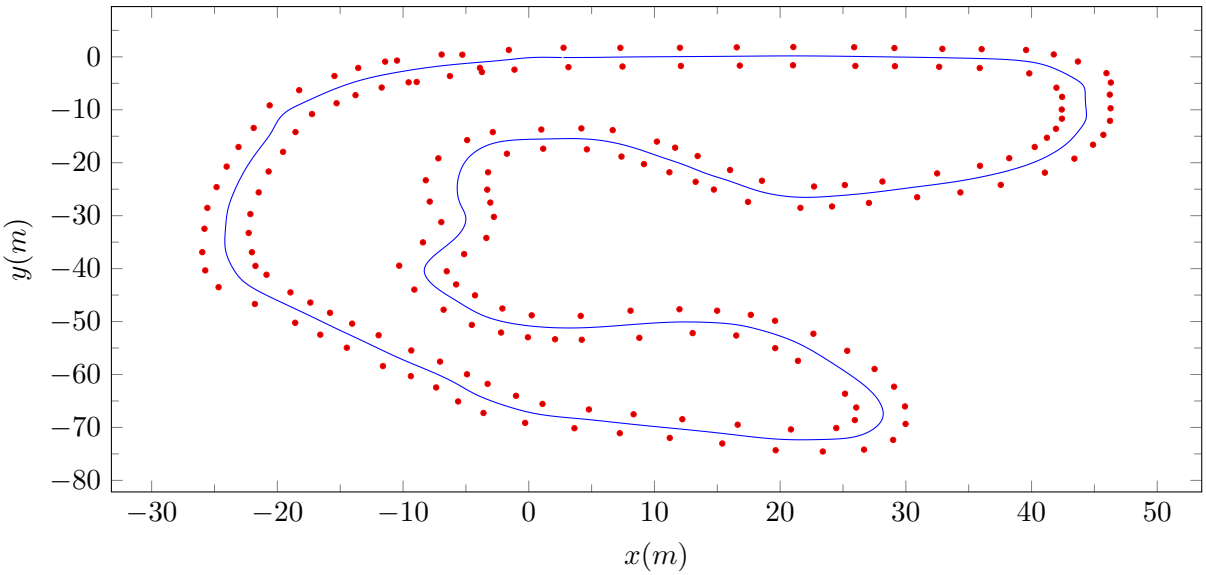


Figure 42: FSG 2019 calculated trajectory.

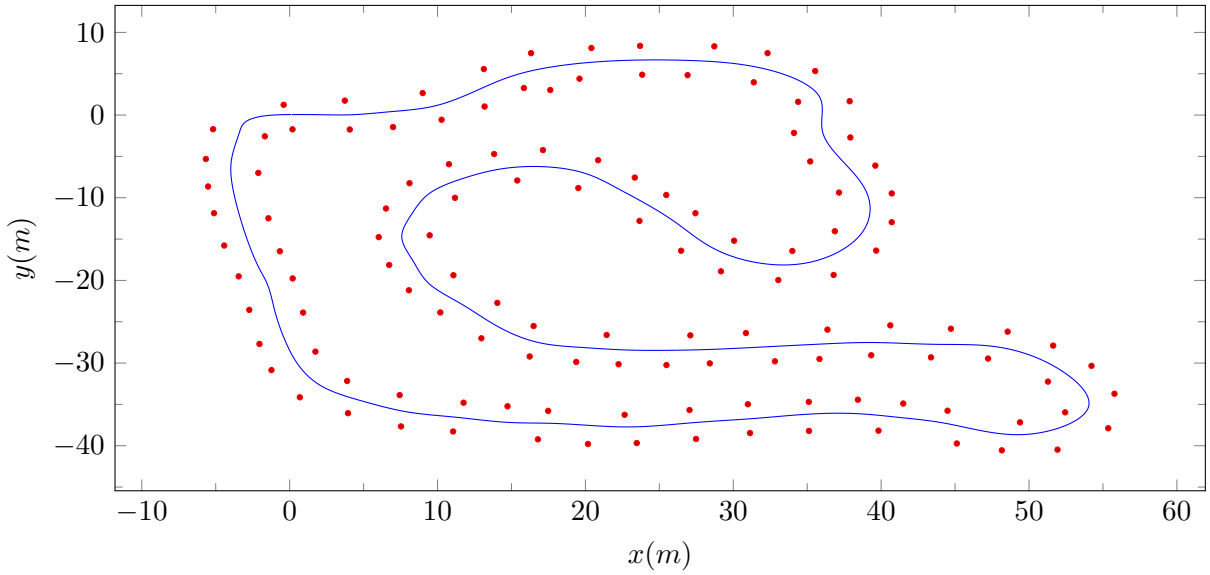


Figure 43: FSS 2019 calculated trajectory.

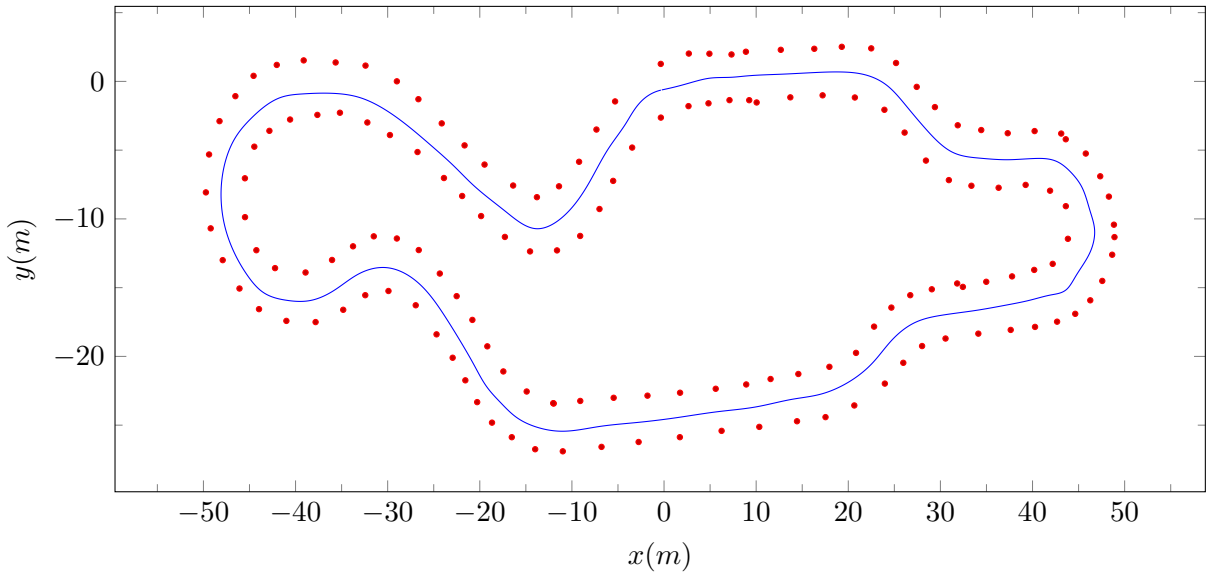


Figure 44: FSI 2019 calculated trajectory.

B.2 Curvature

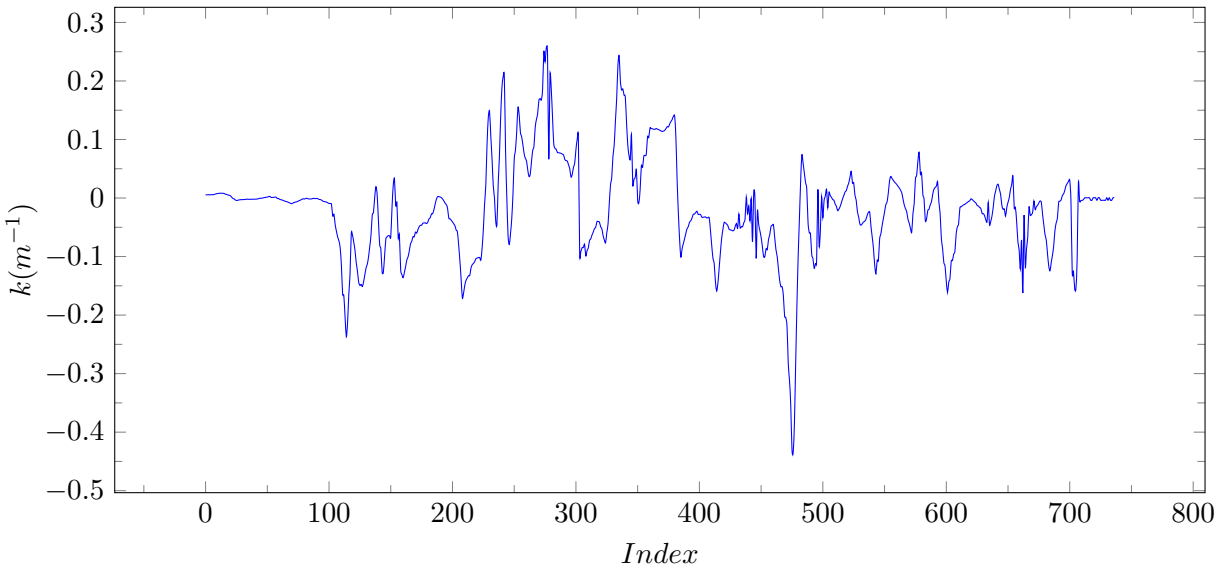


Figure 45: FSG 2019 new planner calculated curvature.

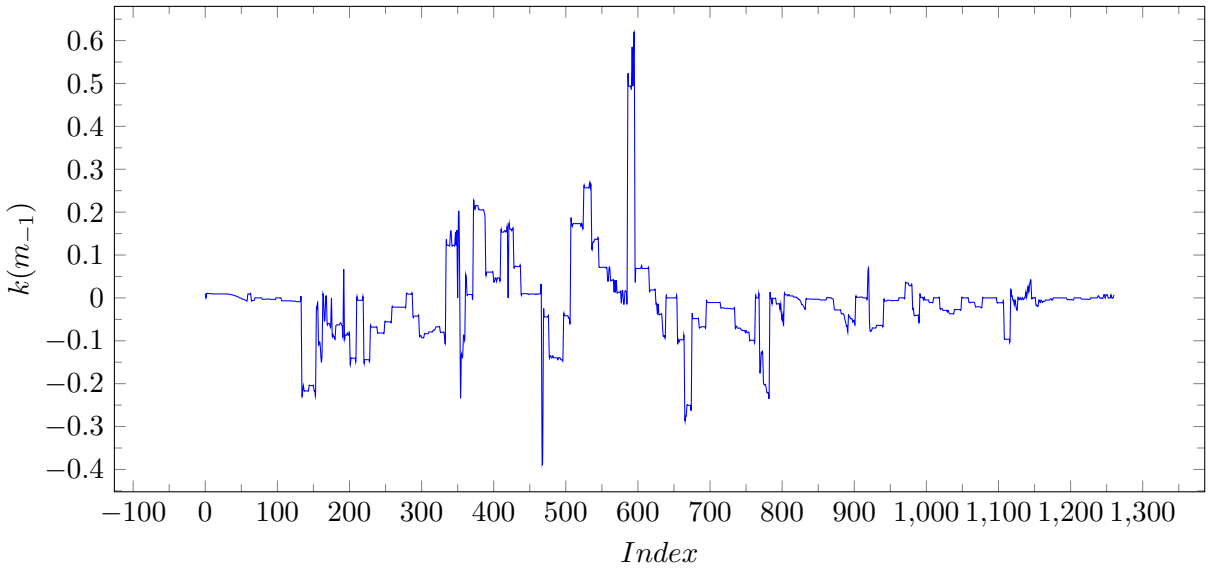


Figure 46: FSG 2019 old planner calculated curvature.

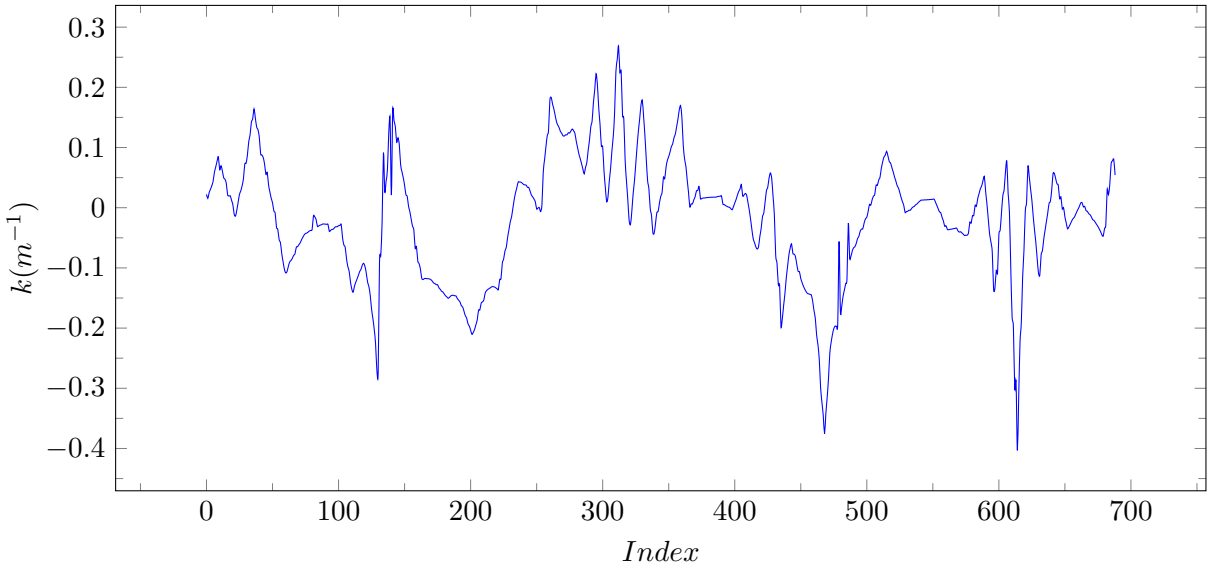


Figure 47: FSS 2019 new planner calculated curvature.

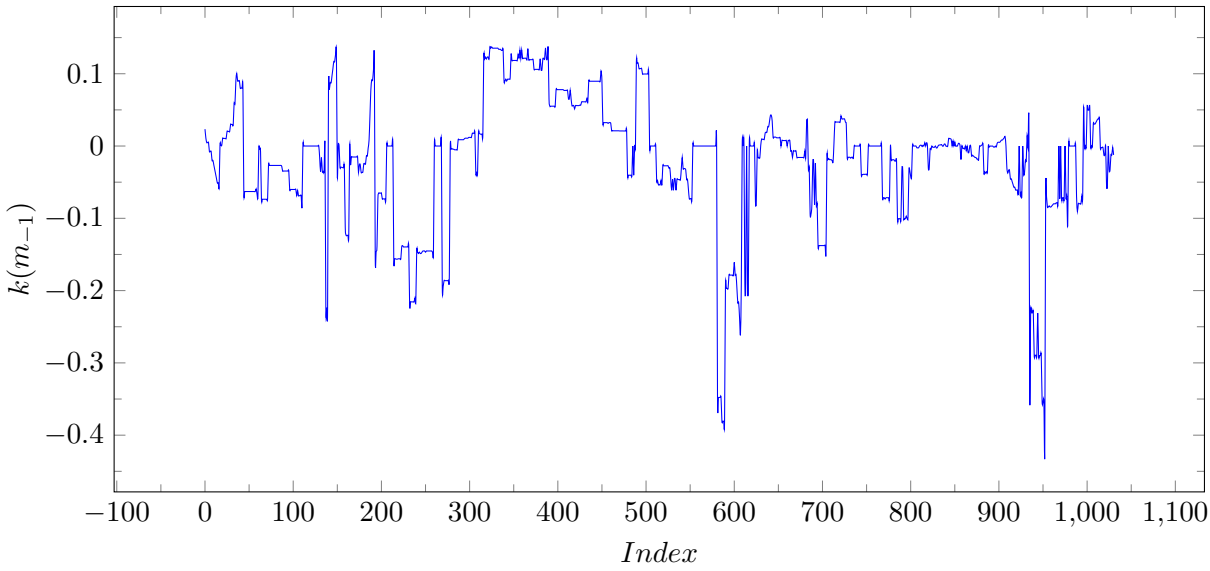


Figure 48: FSS 2019 old planner calculated curvature.

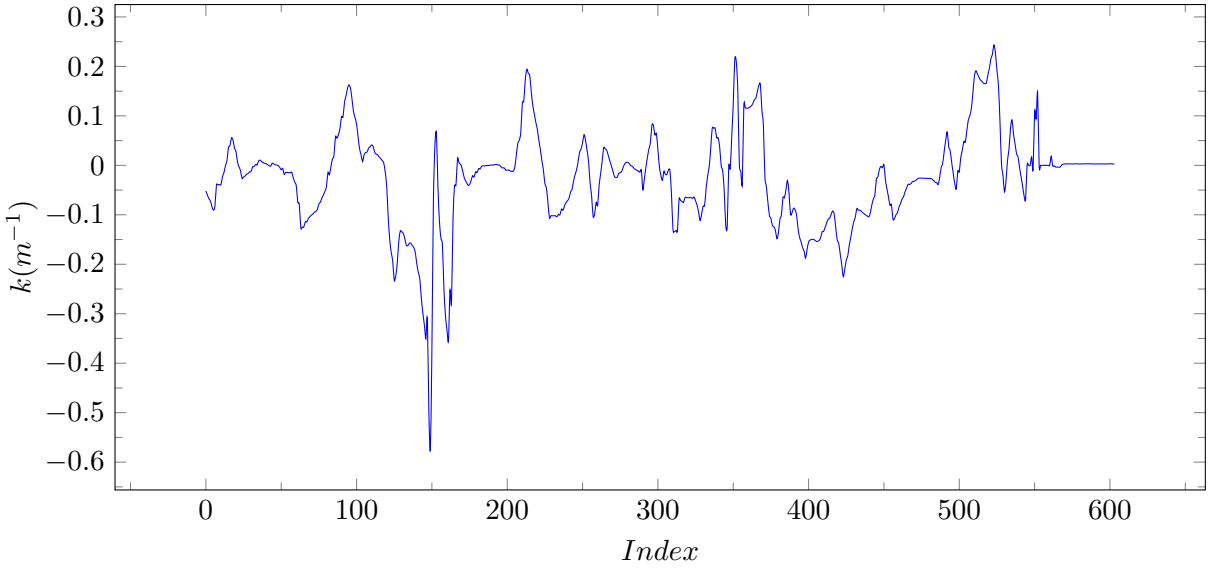


Figure 49: FSI 2019 new planner calculated curvature.

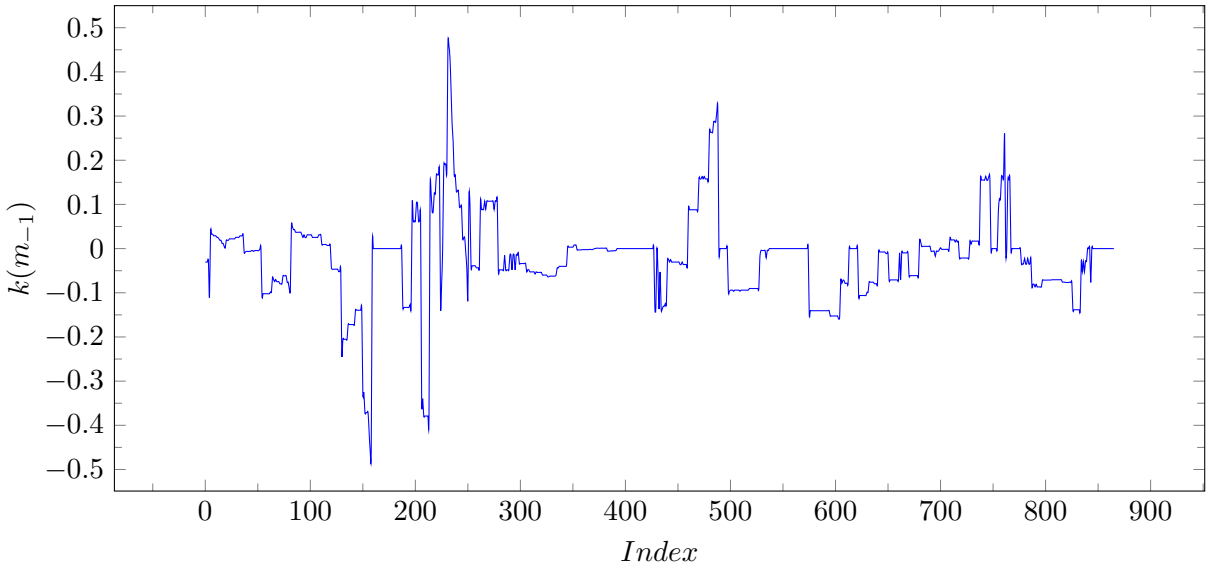


Figure 50: FSI 2019 old planner calculated curvature.