Trajectory Planning, Guidance and Control of a Race Car

Miguel Ângelo D'Ajuda
miguel.dajuda@tecnico.ulisboa.pt
Instituto Superior Técnico, Universidade de Lisboa, Lisboa, Portugal

***Abstract*—The goal of this dissertation is the planning of an optimal trajectory for a car prototype around a predefined race track and the control of the vehicle itself. The vehicle used was one of "Mobile Energy Sustainability Project"s (PSEM) unused cars, specifically, GP17Evo which was reworked as an autonomous vehicle.**

To study the trajectory and control of the GP17Evo prototype, TORCS was used as a simulator, where it was possible to recreate the vehicle digitally. MPC was the key tool used to solve both the path optimization step and the reference tracking task. A model of the vehicle was also built in order to capture the dynamics of the system, which are needed for the problem formulation. Later, in order to validate the simulation results, the prototype will drive in a track or test circuit.

The global optimal trajectory was calculated successfully for several maximum velocities and the proposed MPC controller was able to steer and accelerate the car to the finish line smoothly. These results were validated both in MATLAB and TORCS.

***Index Terms*—MPC, Autonomous Vehicles, Electrical Vehicles, Trajectory Planning, Control, PSEM**

I. Introduction

Electrical autonomous vehicles are changing the paradigm of the automotive world, not only in respect to public roadways, but also making its way to racing competitions. These two worlds have a symbiotic relationship — even though they have distinct purposes, the advancements in technology required to make progress is very similar.

After being known for partaking in Shell Eco-Marathon, in 2013 PSEM began building electric vehicles using only two 12 Volt lead batteries and a 240 Watt DC motor. Ever since, the team as been growing and evolving, creating extremely efficient vehicles capable of running for one hour straight at around 65 km/hour. PSEM aspired to push the vehicle's performance limits, proposing to adapt one of its car as a driver-less lightweight version to serve as a demonstration of highly efficient use of such a minimal power-train.

Therefore, the main focus of this thesis is the design of optimal trajectories and controllers for an autonomous race car. From the localization of the boundaries of a race track obtained in a calibration phase, the final goal is to race through the track autonomously taking into account the time criteria. To achieve this, we search for existing optimal trajectory planning, guidance and control methodologies and optimization tools in the field of ground autonomous vehicles. We also survey existing car race simulators for research that allows us to access and control in real-time the navigation and guidance system.

II. State of the Art

A. Trajectory Planning

This stage is usually formulated as an optimization problem since we want to minimize a specific cost function. This is a very familiar subject, with multiple possible solutions as shown in [6].

As for purely offline solutions, several strategies have been developed over the years [7], [10], [15], [22] on this topic. Formulating this as a geometrical problem, the goal can either be to calculate the shortest path (SP), minimizing the distance traveled, or the minimal curvature path (MCP), maximizing the velocity of the car, or even a conjugation of both. The main disadvantage of these solutions resides on the locked nature of the trajectory *i.e.* once the path is calculated, it is never changed. Furthermore, we are not minimizing the time it takes to complete a lap directly, we are constructing theories around suppositions of what would the best trajectory look like, disregarding the time dimension in the first step of the solution.

In recent years, with the increasing capabilities of processors and graphics cards, it is possible to calculate the solution for these problems in less than a second, making it viable to implement Model Predictive Controllers (MPCs) in real time [8], [11], [26], [29], [38], [39], [41], [48], [49]. This approach is becoming very popular among autonomous racing applications since it is very versatile, capable of dealing with measurement errors, obstacles in the track [21], [40] or even overtaking other cars [16], [28]. MPC can be used as a high performance controller to follow a reference and it can also be used with a custom cost function which maximizes progress to calculate the best trajectory in real time. Probabilistic approaches have also been studied but showing least promising results. One example is the Monte-Carlo Tree Search (MCTS) [24].

Vehicle dynamics [37] come as an important pillar for this work since the MPC will rely on these equations to predict the behaviour of the car. The most common representation uses the bicycle model [8], [20], [29], [30], [36], [48], [49] (or a derivation) to describe the matrix relation between the steering angle and acceleration and the yaw rate, velocities, progress along the track and deviations from the planned path [3], [39]. This model is fairly simple but can lead to an imprecise approximation of reality when dealing with a vehicle at the limits of handling [34], [46], where even tire dynamics are

taken into account, opening the door for more complex physics systems.

B. Guidance and Control

The classic approaches for controlling the car present a solution based solely on a geometrical interpretation of the present state and desired state. This is the case for the pure pursuit method [12], [42] that tries to move the car to a look-ahead point. Another popular geometrical solution is the Stanley controller [1], [47] named after the first car that it was implemented on. This controller design was so successful that it won the DARPA (Defense Advanced Research Projects Agency) Grand Challenge in 2005. [27].

Other approaches use feedforward or feedback systems such as PID controllers [25], [52]. These architectures often use several layers (called sub-controllers) and optimize each variable in a different stage of the process [50].

Surpassing the future prediction barrier, LQR is a very reliable option [19], [35] specially when compared to the previous discussed controllers. Multiple LQR based solutions have been developed, including a Lyapunov stability approach [2], [43], [44], adding feedforward loops [51] and using an alternating direction method of multipliers [32].

The other optimization based approach is MPC, as referred in the previous section. Unlike LQR, it solves an optimization problem for each time step from which the desired actuation inputs are calculated directly. This solution is very versatile, being able to calculate exactly the best manipulated variable values in real time, taking into account the current states and the desired ones for a specified upcoming range. Depending on the solvers used, MPC can handle all types of formulations: constrained/unconstrained and linear [23]/nonlinear [18] problems and models, the obvious downside being the computational burden associated with such optimizations. Luckily, the evolution of processor's capabilities is setting the pillars for these types of approaches to control problems.

The final possible solution to this guidance problem involves the use of Machine Learning Control [17] basing its performance on data and not the dynamics of the system. This can drive the focus away from the system models which can be hard to characterize and turn it into a search for vast data sets of the controller being put to the test. However, these types of data driven controllers are still in an early stage and require even more processing power which we simply do not have, yet.

III. Methodologies

Considering PSEM intends to build the autonomous version of GP17Evo for demonstration purposes, we can calculate an optimal trajectory beforehand, offline. This way, the real-time computational burden will be reduced and we can ensure a global optimization instead of a local solution. After generating the reference path, an MPC is used for reference tracking online. We use a NLMPC (Non Linear MPC) formulation model to maximize the robustness of the system, since the track conditions may vary as well as other external factors.

The focus of this thesis is going to be around building and testing an MPC in a simulation environment, as demonstrated by figure 1. The first two blocks are the same since that is our part of the contribution to the autonomous driving project. The simulation will replace the real car and the added noise will simulate the imperfections of the state estimator. The

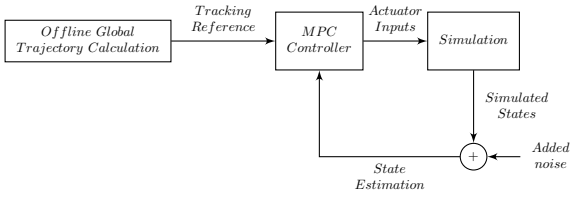

Fig. 1: System architecture of the simulation

simulation will replace the real car and the added noise will simulate the imperfections of the state estimator.

A. Vehicle Model

Based on [8], the vehicle is modeled in a state space representation where

$$\mathbf{X} = \begin{bmatrix} v_x & v_y & \dot{\psi} & e_\psi & e_y & s \end{bmatrix}^T \tag{1}$$

is a column matrix containing the state variables and

$$\mathbf{U} = \begin{bmatrix} a & \delta \end{bmatrix}^T \tag{2}$$

is a column matrix containing the inputs of the system, all of which are represented in figure 3. The state variables v_x and v_y are the longitudinal and lateral velocities of the vehicle (moving axis), $\dot{\psi}$ is the yaw rate, e_ψ and e_y are the heading and lateral distance errors, measured between the car and the center line of the track and s is the distance traveled along the center line (used to assess progress), as demonstrated in figure 2. The inputs a and δ are the longitudinal acceleration and the steering angle. A bicycle model with 3 degrees of freedom

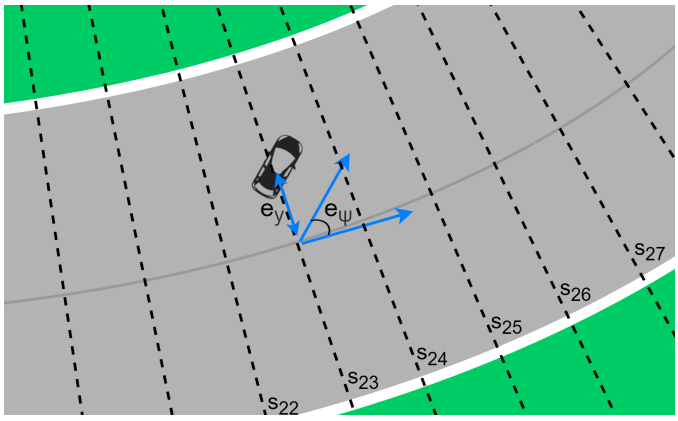

Fig. 2: Representation of the state variables not included in the car dynamics

was chosen (v_x, v_y and $\dot{\psi}$), which is a typical formulation for cars with front wheel steering.

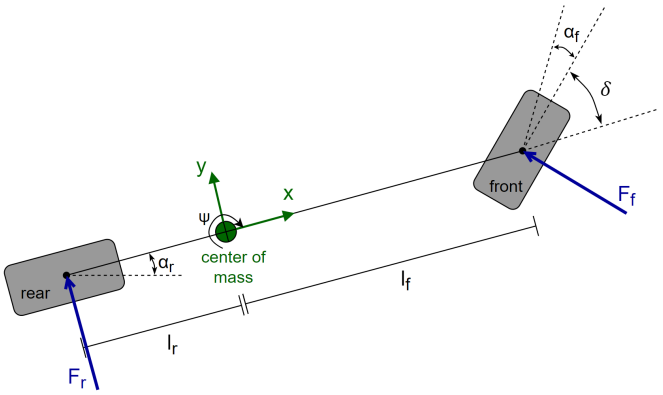

Fig. 3: Representation of the state variables and inputs

The Euler discretization method was used to keep coherence between both simulations (TORCS and MATLAB) and the real case. To that end,

$$\mathbf{X}_{t+1} = \mathbf{X}_t + \Delta t \ f(\mathbf{X}_t; \mathbf{U}_t) \tag{3}$$

shows that the state variables in $\mathbf{X}$ at time $t+1$ depend on their previous values and a function f multiplied by a time step, where f is the slope of $\mathbf{X}$.

As for the tire model, [5] suggests a representation capable of describing the behaviour of the tire while both turning and braking. Other works like [4] or [45] suggest models based on cornering stiffness estimations. For our purpose, considering the importance of low computational times and the wide turning radius of the vehicle, we will use a simplified model that approximates cornering stiffness to a constant. This way, we can define the forces applied on the tires as

$$\begin{cases} F_f = C_{\alpha f} \ \alpha_f \\ F_r = C_{\alpha r} \ \alpha_r \end{cases} , \tag{4}$$

where $C_{\alpha f}$ and $C_{\alpha r}$ are the front and rear cornering stiffness.

The other three state space variables (s, e_y and e_ψ) have the purpose of situating the car relative to the track. Therefore, before establishing this bridge, we need to define an inertial curvilinear reference frame that describes the vehicle position relative to the track as a function of v_x, v_y and $\dot{\psi}$. The simplest way to obtain these equations is to do the opposite and more intuitive task first, write v_x, v_y and $\dot{\psi}$ as a function of s, e_y and e_ψ [31]. Furthermore, it is necessary to define the track analytically [13].

Consequently, we can include the equations for the other three state space variables (e_ψ, e_y and s) in order to build the row matrix

$$f(\mathbf{X}_t; \mathbf{U}_t) = \\ = \begin{bmatrix} \dot{\psi} v_y + a \\ \frac{C_{\alpha f}+C_{\alpha r}}{m}\frac{v_y}{v_x} + \frac{l_f C_{\alpha f} - l_r C_{\alpha r}}{m}\frac{\dot{\psi}}{v_x} - \dot{\psi} v_x - \frac{C_{\alpha f}}{m}\delta \\ \frac{l_f C_{\alpha f} - l_r C_{\alpha r}}{I_z}\frac{v_y}{v_x} + \frac{l_f^2 C_{\alpha f} + l_r^2 C_{\alpha r}}{I_z}\frac{\dot{\psi}}{v_x} - \frac{l_f C_{\alpha f}}{I_z}\delta \\ \dot{\psi} - \frac{v_x \cos e_\psi - v_y \sin e_\psi}{1 - e_y C(s)} C(s) \\ v_x \sin e_\psi + v_y \cos e_\psi \\ \frac{v_x \cos e_\psi - v_y \sin e_\psi}{1 - e_y C(s)} \end{bmatrix} . \tag{5}$$

B. Trajectory Planning

In order to calculate the optimal trajectory, we are going to solve an optimization problem offline that encapsulates the vehicle dynamics and limitations. Since an MPC design was going to be developed for the tracking stage, we can make use of a more generic, open loop implementation of this formulation to calculate the optimal trajectory. The solution of this problem is an array of state variables which will then serve as a reference for the tracking MPC. The formulation can be expressed as

$$\begin{aligned} \text{Maximize} \quad & \sum_{i=1}^{N} s_i^2 \ - \ 500 \times \sum_{i=1}^{N} v_{y_i}^2 \\ \text{w.r.t.} \quad & \mathbf{X}_i, \mathbf{U}_i \\ \text{subject to} \quad & \mathbf{X}_{i+1} = \mathbf{X}_i + \Delta t \ f(\mathbf{X}_i; \mathbf{U}_i) \\ & v_{min} \leq v_{x_i} \leq v_{max} \\ & -0.9 \times \frac{w_{track}}{2} \leq e_{y_i} \leq 0.9 \times \frac{w_{track}}{2} \\ & 0 \leq s_i \\ & a_{min} \leq a_i \leq a_{max} \\ & \delta_{min} \leq \delta_i \leq \delta_{max} \\ & \dot{a}_{min} \leq \dot{a}_i \leq \dot{a}_{max} \\ & \dot{\delta}_{min} \leq \dot{\delta}_i \leq \dot{\delta}_{max} \ . \end{aligned} \tag{6}$$

Given the right parameters, the optimization problem can be solved in order to obtain our reference trajectory. Looking at equation (6), we need to define v_{min}, v_{max}, w_{track}, a_{min}, a_{max}, δ_{min}, δ_{max}, $\dot{a}_{min}$, $\dot{a}_{max}$, $\dot{\delta}_{min}$ and $\dot{\delta}_{max}$. Due to the singularity, the minimum velocity cannot be zero and the maximum velocity should be set higher than the current recorded fastest speed of the vehicle since it will be around 50% lighter without a pilot. The track width can be obtained from the simulator. The maximum acceleration and deceleration can be calculated experimentally from the time it takes the car to get from zero to a certain speed and back to zero. The limitations of the steering angle were defined in the designing stages of the GP17Evo. All of these values can be found on table I The maximum longitudinal velocity is changed for tests at lowers speeds. The track width constraint is scaled down by a factor of 10% to provide a buffer for eventual deviations from the reference trajectory. Besides maximizing the square of the progresses, there is a term minimizing the square of the lateral velocities in order to produce a smoother path.

TABLE I: Car parameters.

Parameter	Minimum value	Maximum value
v_x	0.001 m/s	27.77 m/s
w_{track}	10 m	
a	-5 m/s^2	1 m/s^2
δ	-5 °	5 °
$\dot{a}$	-5 m/s^3	5 m/s^3
$\dot{\delta}$	-1 °/s	1 °/s

C. Guidance and Control

Once the state variables reference is defined, we need to design a tracking controller. For a better understanding of its logic the pseudo-code in algorithm 1 presents how the whole mechanism works.

Algorithm 1 MPC formulation

$\mathbf{X}_{ref} \leftarrow$ load reference trajectory
$Duration, Ts \leftarrow$ duration and sample time
$p, h \leftarrow$ prediction and control horizon
$\mathbf{X}_0, \mathbf{U}_0 \leftarrow$ initial state and control input
$\mathbf{W}_X, \mathbf{W}_U, \rho \leftarrow$ tuning weights
for $k \leftarrow 1 : Duration/Ts$ **do**
  $\mathbf{U} \leftarrow$ Optimize $(\mathbf{X}_0, \mathbf{U}_0, \mathbf{X}_{REF_{k+1,\ldots,k+p}})$
  $\mathbf{X}_0 \leftarrow f(\mathbf{X}_0, \mathbf{U}_1, Ts)$ ▷ update system state
  $\mathbf{U}_0 \leftarrow \mathbf{U}_1$
  history $\leftarrow \mathbf{X}_0, \mathbf{U}_0$ ▷ store variables for later
end for

Now, looking closer at the heart of the MPC, the optimization problem formulation is shown in equation (7).

$$\begin{aligned}
\text{Minimize} \quad & \sum_{i=1}^{p} \left\{ \sum_{j=1}^{6} \left\{ \frac{W_{X_j}}{S_{X_j}} (X_{REF_{j,i}} - X_{j,i}) \right\}^2 + \right. \\
& \left. + \sum_{j=1}^{2} \left\{ \frac{W_{U_j}}{S_{U_j}} (U_{j,i} - U_{j,i-1}) \right\}^2 + \rho \epsilon_i^2 \right\} \\
\text{w.r.t.} \quad & \mathbf{X}_i, \mathbf{U}_i, \epsilon_i \qquad (7) \\
\text{subject to} \quad & \mathbf{X}_{i+1} = \mathbf{X}_i + \Delta t \; f(\mathbf{X}_i; \mathbf{U}_i) \\
& v_{min} - \rho\epsilon_i \leq v_{x_i} \leq v_{max} + \rho\epsilon_i \\
& -\frac{w_{track}}{2} - \rho\epsilon_i \leq e_{y_i} \leq \frac{w_{track}}{2} + \rho\epsilon_i \\
& 0 - \rho\epsilon_i \leq s_i \\
& a_{min} - \rho\epsilon_i \leq a_i \leq a_{max} + \rho\epsilon_i \\
& \delta_{min} - \rho\epsilon_i \leq \delta_i \leq \delta_{max} + \rho\epsilon_i \\
& \dot{a}_{min} - \rho\epsilon_i \leq \dot{a}_i \leq \dot{a}_{max} + \rho\epsilon_i \\
& \dot{\delta}_{min} - \rho\epsilon_i \leq \dot{\delta}_i \leq \dot{\delta}_{max} + \rho\epsilon_i \\
& [\mathbf{U}_{h+1} \quad \mathbf{U}_{h+2} \quad \ldots \quad \mathbf{U}_{p-1} \quad \mathbf{U}_p] = \\
& = [\mathbf{U}_h \quad \mathbf{U}_h \quad \ldots \quad \mathbf{U}_h \quad \mathbf{U}_h]
\end{aligned}$$

Since not all state and input variables have the same order of magnitude, before weighting them, we divide each one by a scale factor (S_{X_j} or S_{U_j}), to ensure that their values are comparable [9], [33]. The weight matrix $\mathbf{W}_X$ and $\mathbf{W}_U$ define the importance or penalization for each term of the cost function. Looking at the cost function, we are minimizing the sum over the prediction horizon of three different terms, the state reference tracking, the manipulated variables rate of change suppression and the slack variable constraint violation. The reference tracking term penalizes the square of the deviations from the desired state in order for the car to follow the planned path. The input move suppression penalizes the square of the derivative in the control inputs, aiming to maintain stability and reduce the energy cost of running the system. The last term penalizes the square of the slack variable, making it harder for the original constraints to be broken.

IV. Results

A. Trajectory Planning

The resulting optimal trajectory can be represented as a 2D scatter plot, as shown in figure 4.

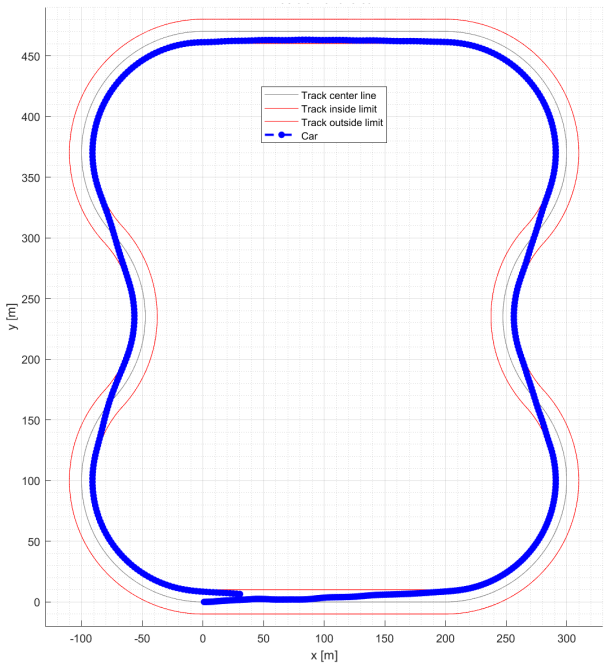

Fig. 4: Improved desired trajectory for the car with maximum allowed $v_x = 100$ km/h

The control inputs and state variables are represented in figures 5 and 6 respectively.

Three other speeds (25 km/h, 50 km/h and 75 km/h) were also used for the simulation in order to test those conditions as well. Their trajectories are equal, as expected. For the velocity to have an effect on the trajectory, the car would have to reach speeds beyond what is possible with the 240 W DC motor that the GP17Evo prototype uses or the track would have to be smaller, like shown earlier.

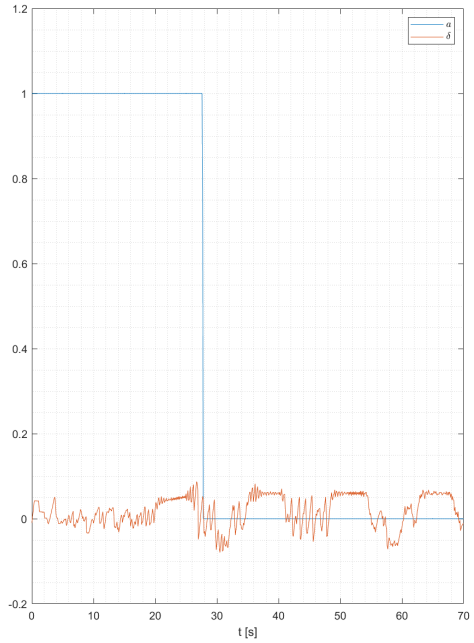

Fig. 5: Inputs for the optimal trajectory with maximum allowed $v_x = 100$ km/h

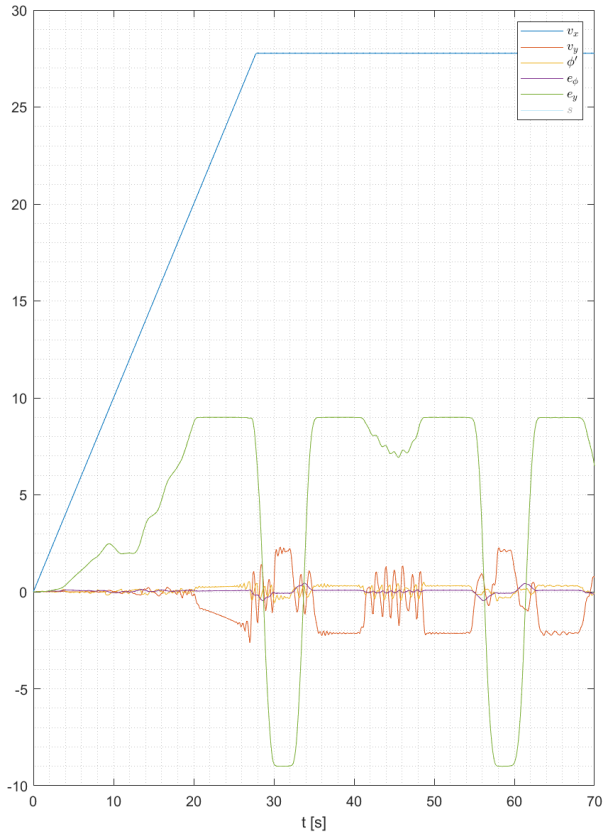

Fig. 6: Evolution of the state variables (excluding s) with maximum allowed $v_x = 100$ km/h

B. Guidance and Control

The guidance and control section was divided into three phases, the PID controller from the TORCS tutorial, the MPC controller in MATLAB, and the MPC controller in TORCS. The first phase serves the purpose of better understanding the mechanics of the TORCS simulator, as well as validating the parameters of the generated car in-game. The second phase is the development and design of the MPC controller and the last phase will serve as the final test since it has the most complete physics representation of the car.

Starting in MATLAB, before tuning the cost function weights, it is good practice to establish the control and prediction horizons first, making sure they cover the transient response of the system, but not much further since it will cause longer computational times. The process of choosing the control and prediction horizons, h and p, is done by trial and error. Generally speaking, bigger horizons mean better performance, even though it might not be the case [14].

Firstly in order to evaluate the results, we need to define a global error performance metrics for the controller. The generalized formula to calculate the aggregate error is given by

$$\text{Aggregate Normalized Error} = = \frac{\sqrt{\sum_{j=1}^{6}\sum_{i=1}^{N}\left(\frac{X_{REF_{j,i}}-X_{j,i}}{S_j}\right)^2}}{N}, \quad (8)$$

where N is the number of samples for the run and the Aggregate Normalized Error uses the scale factor S_j defined in the MPC formulation. Several combinations of prediction and control horizons were experimented with. In this stage, all the weights $\mathbf{W}_X$ and $\mathbf{W}_U$ will be equal to 1 and the equal concern for relaxation ρ will be 10^5, which are typical values for the start of the design. Table II summarizes the simulation results. We can observe that, generally speaking, our performance metrics (aggregate error, number of slack variable activations – N.S.V.A. – and lap time) decrease with the increase of the control horizon, which was expected. It should be noted that there is a trade off here since the value of h affects the computational time significantly whilst the performance is almost at its peak for values much smaller than $h = 10$. After weighting all factors in, the horizons chosen will be $p = 20$ and $h = 4$ which is typical for an application like this.

Now that the prediction and control horizons are fixed, we can start tuning the weights for the cost function in (7). The individual tracking error of each state variable can be calculated using

$$X_j \text{ Error} = \frac{\sqrt{\sum_{i=1}^{N}(X_{ref_{j,i}} - X_{j,i})^2}}{N}. \quad (9)$$

Firstly, we will try to understand the correlations between variables, changing the values of W_{X_j} one at a time and analysing the results, maintaining all the other parameters the same (weights equal to one, etc). The results of all the relevant experiments is shown in table III.

From the overall experiments conducted thus far, certain conclusion and suppositions can be made. Firstly, the MPC

TABLE II: Prediction and control horizons

p	h	Aggregate Error	N.S. V.A.	Lap Time [m:ss.dc]
10	2	21.9×10^{-3}	193	1:26.7
	5	23.8×10^{-3}	97	1:27.8
	10	20.9×10^{-3}	97	1:20.4
20	2	9.64×10^{-3}	76	1:11.1
	3	4.10×10^{-3}	13	1:09.1
	4	3.43×10^{-3}	13	1:09.1
	5	3.59×10^{-3}	15	1:09.1
	10	3.36×10^{-3}	7	1:08.9
25	2	14.0×10^{-3}	115	1:16.4
	3	8.95×10^{-3}	45	1:10.5
	4	5.87×10^{-3}	37	1:09.4
	5	3.86×10^{-3}	15	1:09.2
	10	2.94×10^{-3}	7	1:09.0
30	2	11.5×10^{-3}	95	1:14.1
	5	6.03×10^{-3}	41	1:09.8
	10	3.81×10^{-3}	20	1:09.1
40	2	13.3×10^{-3}	153	1:16.4
	5	8.65×10^{-3}	74	1:11.2
	10	6.17×10^{-3}	58	1:09.7

variables work in an equilibrium fashion, where they are all connected and none of them can be overlooked. In order to improve the performance of one state variable, another one might need to be changed. This can be explained by looking at the cost function as if it is a six way scale (excluding the input and slack variable terms), where for it to be in equilibrium, all of the loads have to be similar. If one variable is getting too far away from the reference values, it will overwhelm the cost function and allow for bigger deviations in the other ones as well.

Finally, after testing just combination of $\mathbf{W}_X$, we can start combining all the parameters at once, table IVb shows the performance metrics of all the pertinent tests performed. Table IVa contains the combinations of parameters used in each one of those tests. Maintaining the weights $\mathbf{W}_X$ and $\mathbf{W}_U$ constant, three values for ρ were tested, 10^5 (default), 10^9 and 10^1. Comparing the cases $\mathbf{W}^{(0)}$, $\mathbf{W}^{(4)}$ and $\mathbf{W}^{(5)}$, we can see that the performance metrics stay the same, with the exception of the number of slack variable activations. Thus, advocating the use of a large constraint breaking penalty.

Looking at the solution with the lowest aggregate error thus far, the main issue is how close e_y gets to its maximum and minimum accepted values. There should be some margin for the case where noise is added. Therefore, the best combination of weights is also based on the proximity of the constraints the solution produces, not only the aggregate error, lap time and computational time. Taking this factor into consideration, $\mathbf{W}^{(7)}$ was the best combination found.

Once the weights for the ideal case are determined, we can test how robust it is, adding noise to each state variable, one by one. Table V presents the maximum standard deviations for each variable, using the previously defined weights ($\mathbf{W}^{(7)}$). We can see that e_ψ and e_y are the most noise sensitive measurements. Once again, these values were determined not only on the resulting solution but also on the computational time it took to get there. It should be noted that, by varying the weights, these maximum standard deviations would reach much larger values.

Trying to replicate the real case, we can use estimations of each standard deviation and tune the controller to fit the real world scenario. As a result, we get table VI. In order to oppose the noise disturbances, the input change penalization was increased and the weights adjusted according to how reliable their signals are expected to be. Tables VIIIa and VIIIb show the four most relevant experiments conducted.

By increasing $\mathbf{W}_U$ we are creating a smoother signal as seen by the actuators. However, this is not enough, it is necessary to adjust $\mathbf{W}_X$ to assure all variables are followed without lag. Figures 7, 8 and 9 show the results for the best case, using $\mathbf{W}_n^{(3)}$. Not only did he trajectory remain very

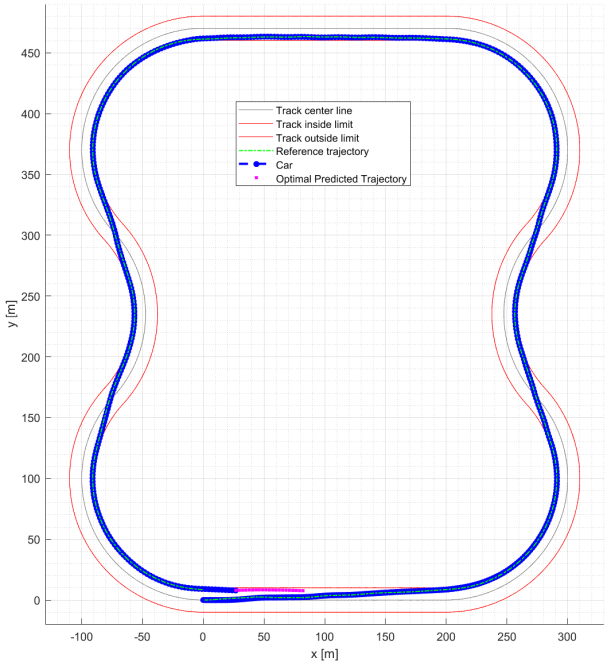

Fig. 7: Position of the car for $\mathbf{W}_n^{(3)}$

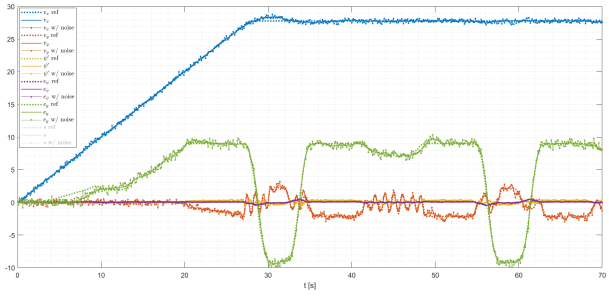

Fig. 8: Evolution of the state variables (excluding s) for $\mathbf{W}_n^{(3)}$

stable, the lap time was the same as in the case without noise. Looking at the state variables evolution, we can easily infer

TABLE III: Individual weight changing effects

X_j	W_{X_j}	v_x Error [m/s]$\times 10^{-3}$	v_y Error [m/s]$\times 10^{-3}$	$\dot{\psi}$ Error [rad/s]$\times 10^{-3}$	e_ψ Error [rad]$\times 10^{-3}$	e_y Error [m]$\times 10^{-3}$	s Error [m]$\times 10^{-3}$	A. Error $\times 10^{-3}$	N.S. V.A.	Lap Time [m:ss.dc]
-	-	9.58	10.7	2.48	0.802	19.3	48.0	3.43	13	1:09.1
v_x	0	22.2	11.1	2.38	0.903	19.6	232	3.53	10	1:09.0
	0.1	22.7	10.8	2.35	0.878	20.6	211	3.49	6	1:09.0
	10	1.23	11.4	2.55	0.834	20.5	103	3.58	10	1:09.2
	2000	0.0132	11.4	2.55	0.822	20.2	98.2	3.57	40	1:09.2
v_y	0	8.94	9.86	2.09	0.786	17.3	48.8	3.05	13	1:09.1
	0.1	8.88	9.85	2.09	0.786	17.3	48.7	3.05	6	1:09.1
	3	201	27.1	6.13	2.74	141	2770	13.7	100	1:17.4
$\dot{\psi}$	0	9.31	11.6	3.13	0.904	21.5	44.9	4.01	12	1:09.1
	0.1	9.32	11.6	3.11	0.902	21.5	44.9	4.00	10	1:09.1
	3	15.8	13.7	2.50	1.11	35.4	128	4.30	23	1:09.3
e_ψ	0	9.65	10.1	2.32	0.865	16.4	48.6	3.24	8	1:09.1
	0.1	9.65	10.1	2.32	0.862	16.4	48.5	3.24	15	1:09.1
	10	210	41.7	7.91	1.14	67.9	6260	14.5	86	1:21.8
e_y	0	8.89	12.2	2.61	0.924	45.2	152	4.38	9	1:09.3
	0.1	8.91	12.0	2.65	0.920	43.6	146	4.32	6	1:09.3
	10	48.2	20.3	4.01	1.17	9.65	529	5.92	12	1:10.5
	50	241	39.1	8.20	2.33	7.47	7420	14.8	68	1:23.3
s	0	9.46	10.6	2.44	0.795	18.6	47.1	3.39	8	1:09.1
	0.1	9.47	10.6	2.44	0.796	18.7	46.9	3.39	7	1:09.1
	1	9.04	10.2	2.33	0.774	18.5	36.5	3.26	12	1:09.1
	1000	5.96	11.9	2.58	0.828	17.0	5.61	3.61	31	1:09.1

TABLE IV: Combined state variable weights effects

(a) Weights combinations

	W_{X_1}	W_{X_2}	W_{X_3}	W_{X_4}	W_{X_5}	W_{X_6}	W_{U_1}	W_{U_2}	ρ
$\mathbf{W}^{(0)}$	1	0.1	0.5	0.2	0.5	0.5	1	1	10^5
$\mathbf{W}^{(1)}$	1	0.1	0.5	0.2	0.5	0.5	0	0	10^5
$\mathbf{W}^{(2)}$	1	0.1	0.5	0.2	0.5	0.5	0.5	0.5	10^5
$\mathbf{W}^{(3)}$	1	0.1	0.5	0.2	0.5	0.5	2	2	10^5
$\mathbf{W}^{(4)}$	1	0.1	0.5	0.2	0.5	0.5	1	1	10^9
$\mathbf{W}^{(5)}$	1	0.1	0.5	0.2	0.5	0.5	1	1	10^1
$\mathbf{W}^{(6)}$	1	0.1	0.5	0.2	0.7	0.5	0.7	1	10^3
$\mathbf{W}^{(7)}$	1	0.5	0.7	0.3	0.7	0.5	0.8	1.2	10^7

(b) Performance metrics

$\mathbf{W}$	v_x Error [m/s]$\times 10^{-3}$	v_y Error [m/s]$\times 10^{-3}$	$\dot{\psi}$ Error [rad/s]$\times 10^{-3}$	e_ψ Error [rad]$\times 10^{-3}$	e_y Error [m]$\times 10^{-3}$	s Error [m]$\times 10^{-3}$	A. Error $\times 10^{-3}$	N.S. V.A.	Lap Time [m:ss.dc]
$\mathbf{W}^{(0)}$	8.00	6.26	2.12	0.859	15.7	47.7	2.65	15	1:09.1
$\mathbf{W}^{(1)}$	8.55	14.1	2.60	0.926	18.4	130	3.99	12	1:09.3
$\mathbf{W}^{(2)}$	1.52	11.2	2.25	0.853	16.7	85.4	3.32	4	1:09.2
$\mathbf{W}^{(3)}$	27.2	28.9	5.88	2.72	73.7	2.84	9.44	77	1:09.7
$\mathbf{W}^{(4)}$	8.00	6.26	2.12	0.859	15.7	47.7	2.65	5	1:09.1
$\mathbf{W}^{(5)}$	8.00	6.26	2.12	0.859	15.7	47.7	2.65	37	1:09.1
$\mathbf{W}^{(6)}$	3.25	7.04	2.21	0.843	14.6	64.1	2.75	9	1:09.1
$\mathbf{W}^{(7)}$	5.14	7.65	2.22	0.825	16.1	57.1	2.85	11	1:09.1

TABLE V: Maximum noise standard deviation tolerated individually.

State Variable	v_x [m/s]	v_y [m/s]	$\dot{\psi}$ [rad/s]	e_ψ [rad]	e_y [m]	s [m]
Maximum σ	2.5	1.2	0.35	0.1	0.7	20

TABLE VI: Expected noise standard deviation.

State Variable	v_x [m/s]	v_y [m/s]	$\dot{\psi}$ [rad/s]	e_ψ [rad]	e_y [m]	s [m]
Expected σ	0.2	0.25	0.02	0.02	0.38	2

TABLE VII: MPC tuned parameters with added noise

Parameter	p	h	W_{X_1}	W_{X_2}	W_{X_3}	W_{X_4}	W_{X_5}	W_{X_6}	W_{U_1}	W_{U_2}	ρ
Value	20	4	1.5	0.5	0.7	0.3	1	0.3	1.2	1.7	10^7

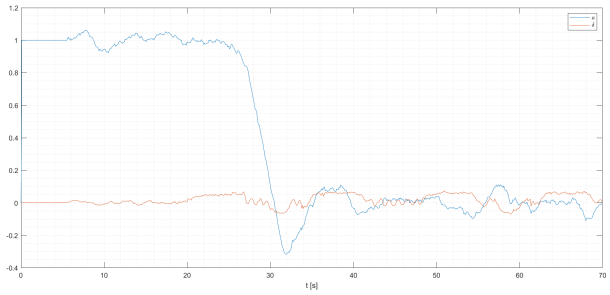

Fig. 9: Evolution of the inputs for $\mathbf{W}_n^{(3)}$

that the noise did not affect the behaviour of the car, once the weights were chosen properly. The best combination of MPC parameters can be found on table VII.

Testing now with added noise, we get table IX, where the weight combination is the best from the previous experiments on MATLAB with the same added noise. Figure 10 shows the evolution of the state variables for this case and figures 11 and 12 show the control inputs.

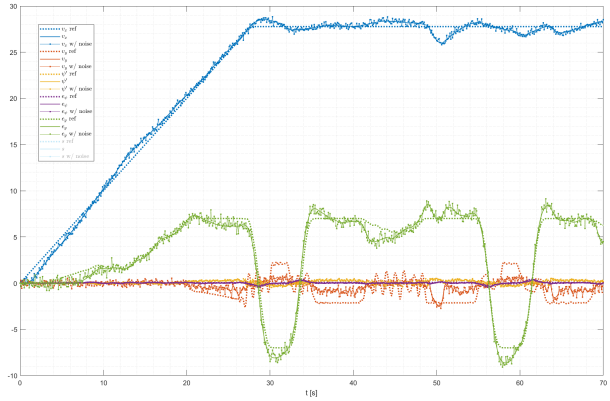

Fig. 10: Evolution of the state variables (excluding s), with noise.

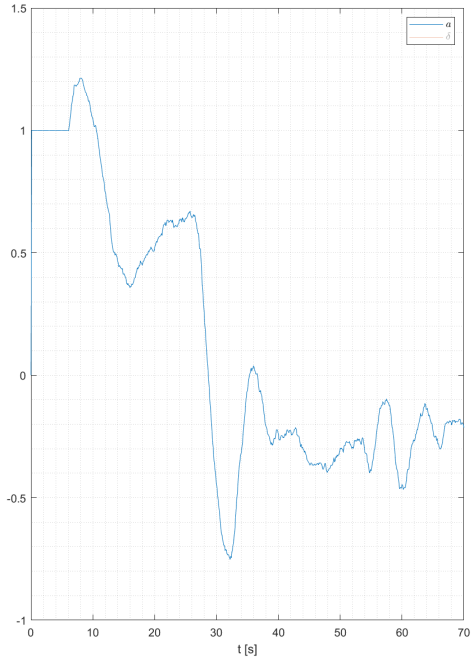

Fig. 11: Acceleration

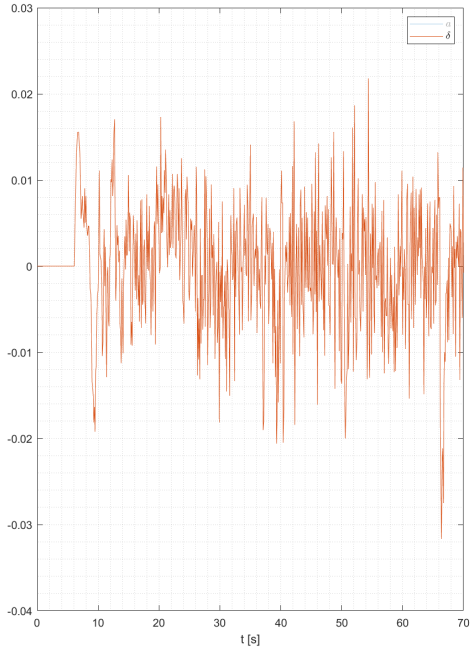

Fig. 12: Steering angle

V. Conclusions

As defined in the objectives of this work, the car was able to navigate through the track while following an optimal trajectory. The theoretical path to follow was calculated from an optimization algorithm as a global minimum of the lap time. In order to follow this trajectory, a state of the art MPC controller was implemented as a robust solution for path following stage. To validate these results, a race simulator was

TABLE VIII: Combined state variable weights effects with added noise

(a) Weights combinations

	W_{X_1}	W_{X_2}	W_{X_3}	W_{X_4}	W_{X_5}	W_{X_6}	W_{U_1}	W_{U_2}	ρ
$\mathbf{W}_n^{(0)}$	1	0.1	0.5	0.2	0.5	0.5	1	1	10^5
$\mathbf{W}_n^{(1)}$	1	0.5	0.7	0.3	0.7	0.5	0.8	1.2	10^7
$\mathbf{W}_n^{(2)}$	1	0.5	0.7	0.3	1	0.3	1	1.4	10^6
$\mathbf{W}_n^{(3)}$	1.5	0.5	0.7	0.3	1	0.3	1.2	1.7	10^6

(b) Performance metrics

W_n	v_x Error [m/s]$\times 10^{-3}$	v_y Error [m/s]$\times 10^{-3}$	$\dot{\psi}$ Error [rad/s]$\times 10^{-3}$	e_ψ Error [rad]$\times 10^{-3}$	e_y Error [m]$\times 10^{-3}$	s Error [m]$\times 10^{-3}$	A. Error $\times 10^{-3}$	N.S. V.A.	Lap Time [m:ss.dc]
$\mathbf{W}_n^{(0)}$	9.48	27.0	5.23	2.20	60.2	261	8.32	29	1:09.6
$\mathbf{W}_n^{(1)}$	4.51	13.0	3.19	1.01	19.2	87.6	4.20	11	1:09.2
$\mathbf{W}_n^{(2)}$	8.02	13.7	3.34	1.16	22.1	123	4.48	30	1:09.2
$\mathbf{W}_n^{(3)}$	4.94	10.3	2.90	0.942	14.9	57.5	3.61	21	1:09.2

TABLE IX: Performance metrics (with noise)

Target v_x [km/h]	v_x Error [m/s]$\times 10^{-3}$	v_y Error [m/s]$\times 10^{-3}$	$\dot{\psi}$ Error [rad/s]$\times 10^{-3}$	e_ψ Error [rad]$\times 10^{-3}$	e_y Error [m]$\times 10^{-3}$	s Error [m]$\times 10^{-3}$	A. Error $\times 10^{-3}$	N.S. V.A.	Lap Time [m:ss.dc]
100	18.7	33.9	5.35	2.13	31.6	148	9.03	0	1:09.1
75	13.1	13.6	3.05	1.64	40.5	177	8.82	6	1:24.4
50	12.2	4.45	2.05	1.31	28.9	267	8.83	11	1:57.4
25	7.72	1.06	1.04	1.71	53.7	477	22.1	54	3:49.8

modified in order to test the previously mentioned stages for our car, GP17Evo.

The MPC was successfully implemented as a closed loop system in real time, using a prediction horizon of 2 seconds and a control horizon of 0.4 seconds. The choice of the weights was adequate to the specified noises from each state variable, both in the early stages of the MATLAB simulation as well as in the TORCS final experiment. Summing up, two MPC formulations were built and used, making it possible for the vehicle to race around the track despite of the user generated added disturbances, which was the primary goal of this work. As a final contribution, all the code produced both in C++ and MATLAB can be reused for later experiments.

ACKNOWLEDGMENT

I would like to use this opportunity to thank all the people I came across in this journey.

Firstly, my former project leader, Vítor Teixeira for suggesting this topic when I showed interest in developing a dissertation with PSEM. Beatriz Silva for embarking on this expedition with me, sharing the burden of navigating blindfolded in the autonomous race car world. Next, my supervisors, Professor Alexandre Bernardino and Professor Paulo Branco for accepting this dissertation subject even though it was proposed by three young students. I also thank Inês Saiote for believing in me, João Vargas and Solange Santos for their mechanical engineering expertise and everyone else that helped me directly or indirectly in this lengthy learning process.

Finally, I would like to thank PSEM and all its members for giving me the opportunity to learn from my mistakes and to enter the electric vehicle world, where I intend to stay.

REFERENCES

[1] A. AbdElmoniem, A. Osama, M. Abdelaziz, and S. A. Maged. A path-tracking algorithm using predictive stanley lateral controller. *International Journal of Advanced Robotic Systems*, 17(6):1729881420974852, 2020.

[2] E. Alcala, V. Puig, J. Quevedo, T. Escobet, and R. Comasolivas. Autonomous vehicle control using a kinematic lyapunov-based technique with lqr-lmi tuning. *Control Engineering Practice*, 73:1–12, 2018.

[3] F. Altché, P. Polack, and A. de La Fortelle. A simple dynamic model for aggressive, near-limits trajectory planning. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 141–147, 2017.

[4] G. Baffet, A. Charara, and D. Lechner. Estimation of vehicle sideslip, tire force and wheel cornering stiffness. *Control Engineering Practice*, 17(11):1255–1264, 2009.

[5] E. Bakker, H. B. Pacejka, and L. Lidner. A new tire model with an application in vehicle dynamics studies. In *SAE International*, volume 98, pages 6101–6113, 1989.

[6] J. T. Betts. Survey of numerical methods for trajectory optimization. *Journal of Guidance, Control, and Dynamics*, 21(2):193–207, 1998.

[7] F. Braghin, F. Cheli, S. Melzi, and E. Sabbioni. Race driver model. *Computers & Structures*, 86(13):1503–1516, 2008. Structural Optimization.

[8] M. Brunner, U. Rosolia, J. Gonzales, and F. Borrelli. Repetitive learning model predictive control: An autonomous racing example. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 2545–2550, 2017.

[9] A. E. Bryson and Y.-C. Ho. *Applied optimal control: optimization, estimation, and control*. Routledge, 1975.

[10] L. Cardamone, D. Loiacono, P. L. Lanzi, and A. P. Bardelli. Searching for the optimal racing line using genetic algorithms. In *Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games*, pages 388 – 394, 09 2010.

[11] A. Carvalho, Y. Gao, A. Gray, H. E. Tseng, and F. Borrelli. Predictive control of an autonomous ground vehicle using an iterative linearization approach. In *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, pages 2335–2340, 2013.

[12] R. C. Coulter. Implementation of the pure pursuit path tracking algorithm, 01 1992.

[13] A. Dhir and S. Sankar. Analytical track models for ride dynamic simulation of tracked vehicles. *Journal of Terramechanics*, 31(2):107–138, 1994.

[14] F. Di Palma and L. Magni. On optimality of nonlinear model predictive control. *Systems & Control Letters*, 56(1):58–61, 2007.

[15] A. Divelbiss and J. Wen. Trajectory tracking control of a car-trailer system. *IEEE Transactions on Control Systems Technology*, 5(3):269–278, 1997.

[16] S. Dixit, U. Montanaro, M. Dianati, D. Oxtoby, T. Mizutani, A. Mouzakitis, and S. Fallah. Trajectory planning for autonomous high-speed overtaking in structured environments using robust mpc. *IEEE Transactions on Intelligent Transportation Systems*, 21(6):2310–2323, 2020.

[17] T. Duriez, S. L. Brunton, and B. R. Noack. *Machine learning control-taming nonlinear dynamics and turbulence*. Springer, 2017.

[18] T. Faulwasser. *Optimization-based solutions to constrained trajectory-tracking and path-following problems*. PhD thesis, Magdeburg, Universität, Diss., 2012, 2013.

[19] D. Fridovich-Keil, E. Ratner, L. Peters, A. D. Dragan, and C. J. Tomlin. Efficient iterative linear-quadratic approximations for nonlinear multi-player general-sum differential games. In *2020 IEEE international conference on robotics and automation (ICRA)*, pages 1475–1481. IEEE, 2020.

[20] S. Fuchshumer, K. Schlacher, and T. Rittenschober. Nonlinear vehicle dynamics control - a flatness based approach. In *Proceedings of the 44th IEEE Conference on Decision and Control*, pages 6492–6497, 2005.

[21] Y. Gao. *Model predictive control for autonomous and semiautonomous vehicles*. University of California, Berkeley, 2014.

[22] A. Heilmeier, A. Wischnewski, L. Hermansdorfer, J. Betz, M. Lienkamp, and B. Lohmann. Minimum curvature trajectory planning and control for an autonomous race car. *Vehicle System Dynamics*, 58(10):1497–1527, 2020.

[23] M. Isaksson Palmqvist. Model predictive control for autonomous driving of a truck. Master's thesis, KTH, School of Electrical Engineering (EES), 2016.

[24] J. Jönsson and F. Stenbäck. Monte-carlo tree search in continuous action spaces for autonomous racing. Master's thesis, Halmstad University, Halmstad, Sweden, june 2020.

[25] N. R. Kapania and J. C. Gerdes. Design of a feedback-feedforward steering controller for accurate path tracking and stability at the limits of handling. *Vehicle System Dynamics*, 53(12):1687–1704, 2015.

[26] A. Katriniok and D. Abel. Ltv-mpc approach for lateral vehicle guidance by front steering at the limits of vehicle dynamics. *Proceedings of the IEEE Conference on Decision and Control*, pages 6828–6833, 12 2011.

[27] G. Kehoe, H. Jula, and F. Ariaei. Developing successful autonomous ground vehicles: Lessons learned from darpa challenges. *IFAC Proceedings Volumes*, 42(15):399–406, 2009. 12th IFAC Symposium on Control in Transportation Systems.

[28] R. Lattarulo and J. Pérez Rastelli. A hybrid planning approach based on mpc and parametric curves for overtaking maneuvers. *Sensors*, 21(2), 2021.

[29] A. Liniger, A. Domahidi, and M. Morari. Optimization-based autonomous racing of 1:43 scale rc cars. *Optimal Control Applications and Methods*, 36(5):628–647, Jul 2014.

[30] A. Liniger and J. Lygeros. Real-time control for autonomous racing based on viability theory. *IEEE Transactions on Control Systems Technology*, 27(2):464–478, 2019.

[31] R. Lot and F. Biral. A curvilinear abscissa approach for the lap time optimization of racing vehicles. In *IFAC*, volume 19, 08 2014.

[32] J. Ma, Z. Cheng, X. Zhang, M. Tomizuka, and T. H. Lee. Alternating direction method of multipliers for constrained iterative lqr in autonomous driving. *arXiv*, 2020.

[33] E. Okyere, A. Bousbaine, G. T. Poyi, A. K. Joseph, and J. M. Andrade. Lqr controller design for quad-rotor helicopters. *The Journal of Engineering*, 2019(17):4003–4007, 2019.

[34] M. Park and Y. Kang. Experimental verification of a drift controller for autonomous vehicle tracking: a circular trajectory using lqr method. *International Journal of Control, Automation and Systems*, 19:1–13, 09 2021.

[35] C. Piao, X. Liu, and C. Lu. Lateral control using parameter self-tuning lqr on autonomous vehicle. In *2019 International Conference on Intelligent Computing, Automation and Systems (ICICAS)*, pages 913–917, 2019.

[36] P. Polack, F. Altché, B. d'Andréa Novel, and A. de La Fortelle. The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles? In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 812–818, 2017.

[37] R. Rajamani. *Vehicle Dynamics and Control*. Mechanical Engineering Series. Springer US, 2011.

[38] U. Rosolia and F. Borrelli. Learning model predictive control for iterative tasks: A computationally efficient approach for linear system. In *IFAC-PapersOnLine*, volume 50, pages 3142–3147, 2017. 20th IFAC World Congress.

[39] U. Rosolia and F. Borrelli. Learning how to autonomously race a car: A predictive control approach. *IEEE Transactions on Control Systems Technology*, 28(6):2713–2719, 2020.

[40] U. Rosolia, F. Braghin, A. Alleyne, and E. Sabbioni. Nlmpc for real time path following and collision avoidance. *SAE International Journal of Passenger Cars - Electronic and Electrical Systems*, 8, 05 2015.

[41] U. Rosolia, A. Carvalho, and F. Borrelli. Autonomous racing using learning model predictive control. In *2017 American Control Conference (ACC)*, pages 5115–5120, 2017.

[42] M. Samuel, M. Hussein, and M. Binti. A review of some pure-pursuit based path tracking techniques for control of autonomous vehicle. *International Journal of Computer Applications*, 135:35–38, 02 2016.

[43] S. Sastry. *Lyapunov Stability Theory*, pages 182–234. Springer New York, New York, NY, 1999.

[44] D. Shevitz and B. Paden. Lyapunov stability theory of nonsmooth systems. *IEEE Transactions on Automatic Control*, 39(9):1910–1914, 1994.

[45] C. Sierra, E. Tseng, A. Jain, and H. Peng. Cornering stiffness estimation based on vehicle lateral dynamics. *Vehicle System Dynamics*, 44(sup1):24–38, 2006.

[46] P. A. Theodosis and J. C. Gerdes. Generating a Racing Line for an Autonomous Racecar Using Professional Driving Techniques. In *ASME 2011 Dynamic Systems and Control Conference and Bath/ASME Symposium on Fluid Power and Motion Control, Volume 2*, Dynamic Systems and Control Conference, pages 853–860, 10 2011.

[47] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney. Stanley: The robot that won the darpa grand challenge. *Journal of Field Robotics*, 23(9):661–692, 2006.

[48] R. Verschueren, S. De Bruyne, M. Zanon, J. V. Frasch, and M. Diehl. Towards time-optimal race car driving using nonlinear mpc in real-time. In *53rd IEEE Conference on Decision and Control*, pages 2505–2510, 2014.

[49] R. Verschueren, M. Zanon, R. Quirynen, and M. Diehl. Time-optimal race car driving using an online exact hessian based nonlinear mpc algorithm. In *2016 European Control Conference (ECC)*, pages 141–147, 2016.

[50] L. Xu, Y. Wang, H. Sun, J. Xin, and N. Zheng. Design and implementation of driving control system for autonomous vehicle. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 22–28, 2014.

[51] T. Yang, Z. Bai, Z. Li, N. Feng, and L. Chen. Intelligent vehicle lateral control method based on feedforward + predictive lqr algorithm. *Actuators*, 10:228, 09 2021.

[52] P. Zhao, J. Chen, Y. Song, X. Tao, T. Xu, and T. Mei. Design of a control system for an autonomous vehicle based on adaptive-pid. *International Journal of Advanced Robotic Systems*, 9(2):44, 2012.