Steering and Speed Control for Autonomous Electric Vehicles

Fábio Alexandre Rodrigues Portelinha
Instituto Superior Técnico / UL, Lisbon, Portugal

***Abstract* - The benefits of using self driving vehicles, regarding safety and environmental aspects, have led to an increased effort from the research and industrial communities to find more competing solutions. This thesis work describes research to obtain an on-line algorithm for autonomous driving of a car, specifically *Veículo Inteligente Electrico de Navegação Autónoma* (VIENA), an electric vehicle developed at *Instituto Superior Técnico* (IST) energy laboratory, from the basis of a Fiat 600 Electra.**

Vehicle models encompass the bicycle model combined with an induction motor controlled with field oriented control. The bicycle model is used to guide the model predictive controller (MPC). A fast and embedded solver (ACADOS) is introduced to optimize the computation time of the MPC. A tire model is introduced to further study speed control while the vehicle is cornering.

Experiments have been conducted in simulation encompassing tests on the controller capability of path following on roads, horizontal or including slopes, while avoiding slip conditions. Promising results have been obtained in terms of completing navigation trajectories, maintaining low path following error and computational complexity. A fast solver was used to produce similar results to the one previously implemented, with a high reduction of computational time for the MPC iterations.

I. Introduction

The recent advances in sensing and computing technology coupled with potential impact on automotive transport industry and social benefits led to research efforts increase in autonomous vehicle technology. In [1], it is referred that in 2014 there were 6.1 million reported collisions, from which 94 % are attributed to driver error and negligence. This ultimately resulted in 32, 675 traffic related fatalities. In [2] it is noted that although only 3 % of vehicle crashes involve a rollover, these are responsible for a third of all passenger deaths. All of this pushes even more the development of autonomous vehicles and all the methodology involved. From an environmental point of view, autonomous vehicles are capable of completing a desired trajectory with higher efficiency when compared to a human actuated vehicle. Going even further, the presence of a sufficient amount of autonomous vehicles can force the non autonomous vehicles to follow energy-optimal trajectories, as referred in [3].

The level of autonomy of a vehicle, according to the Society of Automotive Engineers (SAE), lays in one of six classes from 0 to 5, being 0 a vehicle which does not incorporate any automation. Level 5 describes an unconditionally autonomous vehicle, which does not rely on driver input to avoid undesirable conditions. The work developed aims at producing a vehicle capable of autonomy level 3 or above. At level 3, conditional driving automation is attained, or in other words, the vehicle is able to autonomously drive, still relying on driver input under specific conditions.

This report presents studies on vehicle models, control methodologies and algorithms, tire models for tire-road interaction modelling, and methods to generate energy optimal speed references. The resulting on-line algorithm aims to be implemented in *Veículo Inteligente Electrico de Navegação Autónoma*, (VIENA), an electric vehicle present at the Energy Laboratory in Instituto Superior Técnico (IST) presented in figures 1.

Fig. 1: VIENA

II. VIENA's Vehicle Model

The model used to simulate and control the VIENA car is a simple 2D car model with Ackermann steering as presented in [4]. This model is only viable under certain assumptions: the translations are made only along the body XX axis, the rotations happen only on the ZZ, no lateral or longitudinal slip, and similar to the other models studied, this one is also subject to the non-holonomic constraint. Figure 2 illustrates the reference axis used and variables.

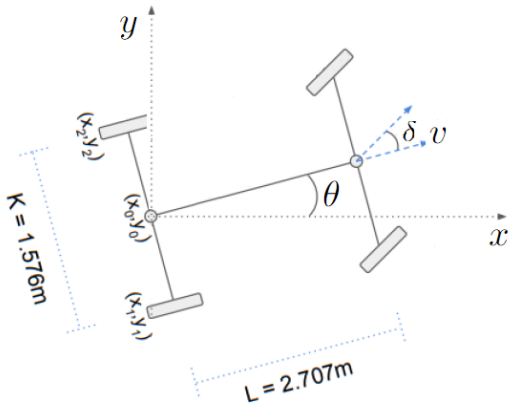

Fig. 2: VIENA reference axis, adapted from [4]

Based on the conditions presented above and on a simplified bicycle model, the general kinematic equations are deduced,

$$\dot{p} = \begin{cases} \dot{x} = v_r \cos(\theta)\cos(\phi) \\ \dot{y} = v_r \sin(\theta)\cos(\phi) \\ \dot{v_r} = a \\ \dot{\theta} = \frac{v_r}{L} tan(\delta)\cos(\phi) \end{cases} . \tag{1}$$

where θ is the vehicle heading, δ is the steering command, and ϕ is the road pitch. This was the model used in previous projects. However, in order to be able to deal with road pitch, a modification in the kinematic equations was required in order to record the vehicle's center of mass height in the inertial coordinate frame. The resulting kinematic equations then become:

$$\dot{p} = \begin{cases} \dot{x} = v_r \cos(\theta)\cos(\phi) \\ \dot{y} = v_r \sin(\theta)\cos(\phi) \\ \dot{z} = v_r \sin(\phi) \\ \dot{v_r} = acc \\ \dot{\theta} = \frac{v_r}{L} \tan(\delta)\cos(\phi) \end{cases} \tag{2}$$

The dynamic bicycle model characterizes the major forces actuating on the vehicle, which are:

1) Air drag, F_a
2) Rolling Resistance, F_r
3) Traction Force, F_t
4) Gravitational Pull, F_g

Figure 3 shows a diagram of the forces actuating in the vehicle, along with the world referential (X, Y, Z) and the vehicle referential (x, y, z).

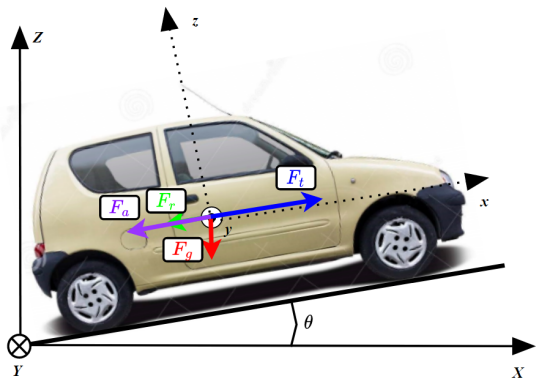

Fig. 3: Kinematic model forces schematic

These forces can be expressed by mathematical expressions. Starting by the air drag, this can be expressed via

$$F_a = -\frac{1}{2}\rho C_d A_f (v - v_w)^2 \tag{3}$$

where ρ is the air density, C_d is the drag coefficient of the vehicle, A_f is the frontal area of vehicle, v is the vehicle speed and v_w is the wind speed. Note that in this equation the wind speed has the same direction as the vehicle speed. When v_w is negative, the wind is contrary to the vehicle movement. The rolling resistance can be computed through

$$F_r = -C_{rr} mg \cos\phi \tag{4}$$

where C_{rr} is the rolling resistance coefficient, m is the vehicle mass, g is the earth's gravity acceleration and ϕ is the vehicle pitch. The gravitational pull is the force that the earth's gravity exerts on the vehicle when one of its components is in the direction of the vehicle trajectory,

$$F_g = -mg \sin\phi. \tag{5}$$

The traction force F_t is obtained from the traction torque T_t expression,

$$T_t = F_t r_w + I_t \alpha_w. \tag{6}$$

where r_w is the wheel radius, I_t is the total inertia (wheels and motor) and α_w is the wheel angular acceleration. Solving (6) with respect to F_t leads to

$$F_t = \left(\frac{T_t}{r_w} - \frac{I_t \alpha_w}{r_w} \right) \tag{7}$$

Following Newton's second law of motion, $a = \frac{F}{m}$, the sum of all the forces applied on the vehicle generates an acceleration inversely proportional to the vehicle's mass. In our case, this means

$$F_t + F_a + F_r + F_g = ma. \tag{8}$$

Inserting (7) in this last expression leads to

$$\frac{T_t}{r_w} + F_a + F_r + F_g = \left(ma + \frac{I_t \alpha_w}{r_w} \right). \tag{9}$$

This means that this dynamic model also accounts for rotational mass. Being that the motor and wheel mass also have an impact in the acceleration of the vehicle it is important

to take this into account. As a general expression, (9) can be expressed as

$$m_{eq}a = \frac{T_t}{r_w} + F_a + F_r + F_g, \tag{10}$$

It is important to mention that F_a and F_r are forces that are normally in the opposite direction of movement, and as such are affected by a minus sign. On the other hand, F_g, can be either positive or negative, according to road pitch. The equivalent rotating mass can be computed through $m_{eq}^{rot} = \frac{1}{r_w^2}(I_w + I_m g_r^2)$, with I_w being the wheel inertia, I_m the motor inertia and g_r the gearbox ratio. This describes the dynamic model used.

The gearbox simply transfers the motor torque and angular speed to the wheels and vice-versa. Associated with this transfer is an efficiency η_{trans} and a ratio g_r,

$$T_w = g_r T_m \eta_{trans} \tag{11}$$

The wheel and motor angular speed ω_w and ω_m, respectively, are related to each other by the gearbox ratio,

$$\omega_w = \frac{\omega_m}{g_r}. \tag{12}$$

III. Speed Control with Cornering

The control structure to be implemented follows the architecture presented in Figure 4.

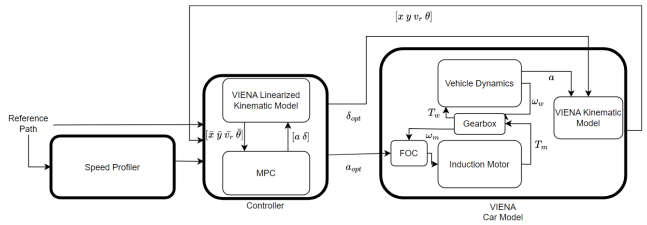

Fig. 4: Complete System Block Diagram

Each component is described next

A. Simulation Model

In order to record all the vehicle dynamics, including vehicle slip, a new model to be implemented in Simulink is proposed. This model follows the implementation in [5]. There are two main parts to this model: the dynamics model and the Magic Formula Tire model. The dynamics model is a bicycle model, similar to the model already implemented in the current Simulink. Figure 5 presents the vehicle model and inertial reference frame.

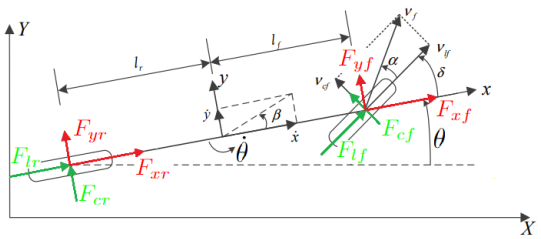

Fig. 5: Three-degree-of-freedom bicycle model, adapted from [5]

The forces actuating in the front and rear tires, both lateral and longitudinal, F_{xf}, F_{yf}, F_{xr} and F_{xy}, can be computed through

$$\begin{cases} F_{xf} = F_{lf}\cos(\delta) - F_{cf}\sin(\delta) \\ F_{xr} = F_{lr} \\ F_{yf} = F_{lf}\sin(\delta) + F_{cf}\cos(\delta) \\ F_{yr} = F_{cr} \end{cases} \tag{13}$$

where δ is the steering angle, F_{lf} and F_{lr} are the longitudinal tire forces in the front and rear tires, F_{cf} and F_{cr} are the lateral tire forces in the front and rear tires, which can be computed via Magic Tire Formula. The latter is a function of tire side slip angle α, tire slip rate s, vertical load F_z and other parameters:

$$\begin{cases} F_l = f_l(\alpha, s, F_z), \\ F_c = f_c(\alpha, s, F_z). \end{cases} \tag{14}$$

The tire side slip angle is related to the tire longitudinal and lateral speed as

$$\alpha = \arctan(\frac{v_c}{v_l}), \tag{15}$$

where v_l is the vehicle wheel longitudinal speed and v_c is the lateral speed. These quantities can be obtained via

$$\begin{cases} v_l = v_y\sin(\delta) + v_x\cos(\delta), \\ v_c = v_y\cos(\delta) - v_x\sin(\delta), \end{cases} \tag{16}$$

where v_x is the linear wheel speed in the x-axis, and v_y the speed in the y-axis. The speed in both axes can be computed from the vehicle speed

$$\begin{cases} v_x = \dot{x} \\ v_y = \dot{y} + l_f\dot{\theta} \end{cases} \tag{17}$$

As for the tire slip rate s, it is considered constant at 20%. Considering this value leads to a good braking effect. In other words it is a simplification to keep a lower complexity. For the vertical load F_z, load transfer is not considered. It is considered that the vehicle accelerates slowly, with no load transfer. In this case the vertical load can be computed

$$\begin{cases} F_{zf} = \frac{l_r mg}{2(l_f + l_r)} \\ F_{zr} = \frac{l_f mg}{2(l_f + l_r)} \end{cases} \tag{18}$$

It is finally possible to compute the tire forces generated. From the tire forces it is possible to compute the force balance in the x-axis, y-axis and around the z-axis using Newton's second law:

$$\begin{cases} m\ddot{x} = 2F_{xf} + 2F_{xr} + m\dot{y}\dot{\theta} \\ m\ddot{y} = 2F_{yf} + 2F_{yr} - m\dot{x}\dot{\theta} \\ I_z\ddot{\theta} = 2F_{yf}l_f - 2F_{yr}l_r \end{cases} \tag{19}$$

As a final note on the vehicle dynamics, one must mention that all of these equations refer to the vehicle body reference frame. In order to convert this into the inertial reference frame one can resort to

$$\begin{cases} \dot{X} = \dot{x}\cos\theta - \dot{y}\sin\theta \\ \dot{Y} = \dot{x}\sin\theta + \dot{y}\cos\theta \end{cases} \tag{20}$$

where X is the longitudinal position and Y is the horizontal position in the inertial coordinate system and φ is the yaw angle.

Let's now dive into the simplified Magic Formula Tire model used in this case. It is simplified because the wheel camber angle influence is ignored, the parameters related to zero point drift are also ignored as well as the empirical parameters that cause asymetry in the formula and higher order terms more than 3 times. This leads to a model with higher accuracy than the linear tire model, while keeping complexity low. This model can characterize the longitudinal force in the purely longitudinal case by

$$\begin{aligned} F_{x0} =& D_x \sin[C_x \arctan\{B_x s_x - \\ & E_x(B_x s_x - \arctan(B_x s_x))\}] + S_{vx}, \end{aligned} \tag{21}$$

where D_x, C_x, B_x, E_x, s_x, and S_{vx} are quantities dependent on tire side slip angle, slip rate, normal load and tire characteristics. The same goes for the pure sideslip case, where the lateral force is given by

$$\begin{aligned} F_{y0} =& D_y \sin[C_y \arctan\{B_y \alpha_y - \\ & E_y(B_y \alpha_y - \arctan(B_y \alpha_y))\}] + S_{vy}, \end{aligned} \tag{22}$$

where once more, the coefficients D_y, C_y, B_y, E_y, α_y, and S_{vy} are also dependent on tire side slip angle, slip rate, normal load and tire characteristics. The mixed case of longitudinal and lateral slip can be expressed as

$$\begin{cases} F_x = G_{x\alpha} F_{x0} \\ F_y = G_{ys} F_{y0}, \end{cases} \tag{23}$$

where $G_{x\alpha}$ and G_{ys} are coefficients that can be computed taking into account the tire slip angle, slip rate, normal load and tire characteristics. Details on how to obtain all of these coefficients are on [5]. It is important to mention that in this case, there are 55 tire parameters related to a $175/70$ $R13$ tire. Taking into account that our vehicle is equipped with a $145/75$ $R13$, which can be considered close enough for this first implementation, since these parameters can be hard to identify. Taking this into account, in our model implementation we will resort to these parameters.

Finally, this results in vehicle dynamics nonlinear model based on Magic Formula tire model:

$$\begin{cases} m\ddot{x} = m\dot{y}\dot{\theta} + 2F_{lf}\cos\delta - 2F_{cf}\sin\delta + 2F_{lr} \\ m\ddot{y} = -m\dot{x}\dot{\theta} + 2F_{lf}\sin\delta + 2F_{cf}\cos\delta + 2F_{cr} \\ I_z\ddot{\theta} = 2l_f F_{lf}\sin\delta + 2l_f F_{cf}\cos\delta - 2l_r F_{cr} \\ \dot{X} = \dot{x}\cos\theta - \dot{y}\sin\theta \\ \dot{Y} = \dot{x}\sin\theta + \dot{y}\cos\theta \end{cases} \tag{24}$$

Having this model fully described, it is now possible to implement it in the Simulink model for VIENA, and understand the vehicle's behaviour under slip circumstances. In the following section, the speed profiler is described along with slip avoiding conditions implemented.

B. Speed Profiler

In this section, the methodology used for the generation of a desirable speed profile is described. The work developed was based on [6], whose methodology is similar to the one presented in [3].

Since one of the main objectives of this speed profiler is to keep a low energy consumption, all the forces which affect the vehicle movement must be modelled. Remembering the main power-absorbing components from section II:

1) Air drag
2) Rolling Resistance
3) Acceleration/deceleration
4) Hill-climbing

Having this in mind, the power at the wheel can be computed through

$$\begin{aligned} P_{wheel}(t) =& M_{eq} v(t)\dot{v}(t) + \frac{1}{2} C_d A \rho (v(t) - v_w)^3 + \\ & mgf_r v(t)\cos(\phi) + mgv(t)\sin(\phi), \end{aligned} \tag{25}$$

where M_{eq} is the equivalent mass of the vehicle (vehicle mass + driveline inertia), m is the vehicle mass, $v(t)$ the vehicle speed, $\dot{v}(t)$ the speed derivative (acceleration), v_w the wind speed in the vehicle reference system (which is considered null as a simplification), C_d the aerodynamic drag coefficient, A the front cross-sectional area of the vehicle, ρ the air density, g the gravitational acceleration, and finally f_r is the rolling resistance coefficient. The angle ϕ translates the pitch of the car. Dissecting Equation (25) in its parcels: the first accounts for power spent at acceleration/deceleration, the second for the air drag, the third for the rolling resistance and finally the last parcel accounts for hill-climbing power expenses.

Based on (25) it is already possible to compute the energy consumed, simply by integrating the expression, as shown in (26). By minimizing the energy at the wheel (considered as a "traction energy"), we can produce speed profile close to energy optimality

$$E_{wheel} = \int_0^{t_f} P_{wheel}(\tau) d\tau. \tag{26}$$

We are now ready to formulate the OCP which must be solved in order to obtain the speed profile. However, it is necessary to define a minimum time (t_{min}) to complete the path, since the solution $v = 0$ has the minimum possible energy, $E = 0$, but the car does not move. t_{min} is user defined and must be sufficiently large to guarantee the feasibility of the problem. To obtain a time interval which does not violate vehicle characteristics, one can subdivide the path into three main segments: one segment of maximum acceleration, followed by a segment of constant maximum speed and, finally, a segment of maximum deceleration until the vehicle stops.

In order to compute the energy needed to complete a certain path, the path is divided into segments of constant acceleration, emulating the actuation of a human driver. This implementation is based on the typical equations for linear motion

$$x(t) = \frac{1}{2} a t^2 + v_i t + x_i, \tag{27}$$

$$v(t) = at + v_i. \tag{28}$$

Two situations arise: null acceleration $a = 0$ and non zero acceleration $a \neq 0$.

In the first case, the time at which the vehicle proceeds at a certain speed can be computed through $\Delta t = \frac{\Delta x}{v_i}$. In these segments, the vehicle speed v_i is constant, which means that the energy consumption can be expressed as

$$E_{wheel} = P_{wheel}\Delta t. \tag{29}$$

On the other hand, when $a \neq 0$, the elapsed time can be obtained by solving $\frac{1}{2}a\Delta t^2 + v_i\Delta t - \Delta x = 0$, and the speed v_i is no longer constant, which must be taken into account when computing the spent energy. The energy spent at the wheels can be computed through

$$\begin{aligned} E_{wheel} =& m(a + g(f_r\cos\theta + \sin\theta)\Delta t_i\left(a\frac{\Delta t_i}{2} + v_i\right) + \\ & \frac{1}{8a}\rho C_d A((a\Delta t_i + v_i)^4 - v_i^4). \end{aligned} \tag{30}$$

It is finally possible to discuss the optimization problem that must be solved to obtain the optimal speed profile at a wheel level.

$$\begin{aligned} \min_{v_n} \quad & \sum_{i=0}^{N} E_{wheel_i} + \alpha|t_{min} - \sum_{n=0}^{N}\Delta t_n| \\ \text{s.t.} \quad & v_0 = v_{init}, \\ & v_N = v_{final}, \\ & v_{min} < v_n < v_{max}, \\ & \dot{v}_{min} < \dot{v_n} < \dot{v}_{max}, \\ & \ddot{v}_{min} < \ddot{v_n} < \ddot{v}_{max}. \end{aligned} \tag{31}$$

The cost function contemplates the sum of energy for every segment of constant acceleration considered. Besides that, the cost function also penalizes deviations from the time interval designated to complete the track. This can be interpreted as a soft constraint on the time available to complete the planned path, making the problem more lenient. This problem fits into the typical non-linear programming problem which can be solved through sequential quadratic programming. This algorithm is implemented using MATLAB's $fmincon$.

In order to further improve the speed profile performance, it is possible to include slip conditions as constraints on this optimization problem. The study on the slip and rollover conditions is presented next, which is based on [2].

Assuming no pitch or roll, to avoid sliding conditions the horizontal force exerted at the wheels, F_h, cannot overcome the friction force between the ground and tires F_f

$$F_h \leq F_f. \tag{32}$$

The friction force F_f is computed through

$$F_f = F_g\mu, \tag{33}$$

where $F_g = mg$, with μ being the friction coefficient determined by the vehicle's operating point (speed and acceleration), tire and road surfaces. The horizontal force for a given curve performed by the vehicle can be divided into two components: longitudinal and lateral forces. The longitudinal force can simply be computed by $F_{long} = ma_n$, with a_n being the longitudinal acceleration. For the lateral forces, the vehicle's path must be approximated to a perfect circle with known radius r. The lateral force is computed through $F_{lat} = m\frac{v^2}{r}$. Finally the horizontal force can be computed

$$F_h = \sqrt{F_{lat}^2 + F_{long}^2}. \tag{34}$$

Figure 6 was taken from [2], where the curvature radius considered is $r = \frac{v}{\omega}$.

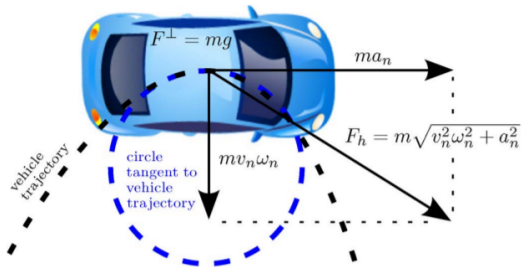

Fig. 6: Horizontal Forces, from [2]

To avoid rollover conditions, there must be an equilibrium between the lateral (centripetal) force and normal (gravitational) force at the center of gravity of the vehicle. This translates into

$$mg\frac{l}{2} = m\frac{v^2}{r}h, \tag{35}$$

where the first term corresponds to the gravitational force applied with a leverage $l/2$ (which is half the car's track width) and the second term to the centripetal force applied at the vehicle's center of gravity, located h above the tire contact surface with the road, figure 7.

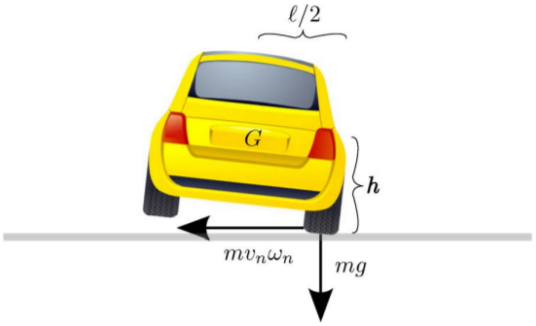

Fig. 7: Rollover Forces, from [2]

Finally, the safety conditions (no slip, no rollover) reduce to

$$\frac{F_h}{F_g} \leq C_1\mu \text{ and } \frac{v^2}{rg} \leq C_2\frac{l}{2h}. \tag{36}$$

The parameters C_1 and C_2 serve as tuning parameters to obtain the desired performance from the system. The static stability factor $\frac{l}{2h}$ usually takes values slightly higher than the friction coefficient μ. This means that the slip condition is prone to

happen before the rollover condition, allowing a simplification since if we avoid slip, the rollover will also be avoided.

The safety condition is inserted in the OCP by a new constraint,

$$\begin{aligned}
\min_{v_n} \quad & \sum_{i=0}^{N} E_{wheel_i} + \alpha|t_{min} - \sum_{n=0}^{N} \Delta t_n| \\
\text{s.t.} \quad & v_0 = v_{init}, \\
& v_N = v_{final}, \\
& v_{min} < v_n < v_{max}, \\
& \dot{v}_{min} < \dot{v_n} < \dot{v}_{max}, \\
& \ddot{v}_{min} < \ddot{v_n} < \ddot{v}_{max}. \\
& \frac{F_h}{F_g} \leq C_1\mu \\
& \frac{v^2}{rg} \leq C_2\frac{l}{2h}
\end{aligned} \tag{37}$$

The constraint before the last is related to the vehicle slip. This constraint is built by computing the radius of the curve at current vehicle position, which then results in

$$\frac{1}{g}\sqrt{\frac{v^4}{r^2} + a^2} \leq C_1\mu \tag{38}$$

The friction coefficient of a dry tarmac road is considered $\mu = 1$.

C. ACADOS Implementation

In [6] an MPC Controller was implemented to control VIENA. Although showing promising results, the controller actually presented some problems that required immediate attention in order for it to ever be viable as a safe on-line algorithm. For starters, its accuracy in reference path following could be improved, as well as reference speed following. The other big problem that the algorithm presented was the high computational time required to compute a single pair of control signals. In order to solve these problems, a new software package was selected to solve the MPC problem. This software package named ACADOS presents itself as a collection of solvers for fast embedded optimization. In [7] all the packages and solvers used within ACADOS are described, as well as the main implementation rules that must be followed.

The general OCP format that the ACADOS software package is able to deal with is

$$\begin{aligned}
\min_{x(\cdot),u(\cdot),z(\cdot)} \quad & \int_0^T l(x(\tau),u(\tau),z(\tau),p) + m(x(T),z(T),p) \\
& /*\, Initial\ Values\, */ \\
\text{subject to} \quad & \underline{x_0} \leq J_{bx,0}x(0) \leq \overline{x_0}, \\
& /*\, Dynamics\, */ \\
& f_{expl}(x(t),\dot{x}(t),u(t),z(t),p) = 0, t \in [0,T) \\
& /*\, Constraints\, */ \\
& \underline{x} \leq J_{bx}x(t) \leq \overline{x}, t \in [0,T) \\
& \underline{u} \leq J_{bu}u(t) \leq \overline{u}, t \in [0,T)
\end{aligned} \tag{39}$$

where

- state vector $x : \mathbb{R} \rightarrow \mathbb{R}^{n_x}$
- control vector $u : \mathbb{R} \rightarrow \mathbb{R}^{n_u}$
- algebraic state vector $z : \mathbb{R} \rightarrow \mathbb{R}^{n_z}$
- model parameters $p \in \mathbb{R}^{n_p}$

This is the simplest formulation that ACADOS is able to receive while still maintaining the functionalities from the previous formulation. In the case at hand, $n_x = 4$, as there are four quantities that express vehicle state: X and Y position of the vehicle, v is the vehicle speed and finally θ is the vehicle orientation, all on the inertial reference frame. As for the control vector, it contains the control signals to be applied. In our case the control signals are the acceleration acc and steering angle δ, which leads to $n_u = 2$. In other words, the state and control vectors take the form

$$x = \begin{bmatrix} X \\ Y \\ v \\ \theta \end{bmatrix}, \qquad u = \begin{bmatrix} acc \\ \delta \end{bmatrix}.$$

In this simple case there is no algebraic state vector z nor model parameters p. After setting the cost function as linear least squares the Lagrange cost term or running cost, $l(x,u,z)$ has the following form

$$l(x,u,z) = \frac{1}{2}\left\| \begin{bmatrix} X \\ Y \\ v \\ \theta \\ acc \\ \delta \\ acc \\ \delta \end{bmatrix} - \begin{bmatrix} x_{nref} \\ y_{nref} \\ v_{nref} \\ \theta_{nref} \\ 0 \\ 0 \\ acc_{prev} \\ \delta_{prev} \end{bmatrix} \right\|_W^2, \tag{40}$$

with x_{nref} and y_{nref} being the reference position at time instant $n \in [1, N-1]$. The same goes for the reference speed v_{nref} and orientation reference θ_{nref}. The sixth and seventh elements of the y_{ref} vector serve as to penalize control signals with high absolute value, having the same effect in the controller as the R matrix in Parrado's implementation. Finally, the last two elements acc_{prev} and δ_{prev} penalize deviations of the control signal from its previous values, guaranteeing the generation of smooth control signals of acceleration and steering, emulating the R_d matrix from Parrado's implementation. Similarly, the Mayer cost term or terminal cost $m(x,u,z)$ takes the form

$$m(x,u,z) = \frac{1}{2}\left\| \begin{bmatrix} X \\ Y \\ v \\ \theta \end{bmatrix} - \begin{bmatrix} x_{Nref} \\ y_{Nref} \\ v_{Nref} \\ \theta_{Nref} \end{bmatrix} \right\|_{W^e}^2, \tag{41}$$

where the reference vector y_{ref}^e contains the last reference point for the prediction horizon which is justified by the subscript $Nref$. This leaves two weighing matrices W and W_e that can be tuned.

Now diving into the constraints from the OCP in 39. The first constraint, as indicated in the expression, is relative to the problem initial conditions and its limits. In our case there are no limits implemented on the initial conditions. That is, the car can be in any position (X, Y), any speed (v) and orientation (θ). The second constraint presented is the model used by

the MPC. Here appears another difference from Parrado's implementation. That is, the model no longer needs to be discretized before being plugged into the MPC. ACADOS uses a multiple shooting technique to discretize the model. As such, a linearized model is plugged into the MPC,

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{v} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v\cos(\theta) \\ v\sin(\theta) \\ acc \\ \frac{v\tan(\delta)}{L} \end{bmatrix} \tag{42}$$

The third and fourth constraints presented define the maximum and minimum limits on the state and control vectors, or in other words, limits on the vehicle's position (X, Y), speed v, orientation θ, acceleration acc and steering angle δ. The vehicle has mechanical constraints on acceleration, steering angle and speed that must be inserted in the MPC constraints. That being said, table I presents the maximum and minimum values for these.

TABLE I: Mechanical Constraints

Parameter	Maximum Limit	Minimum Limit	Units
Speed	50/3.6	-20/3.6	m/s
Acceleration	1	-1	m/s^2
Steering Angle	$\pi/4$	$-\pi/4$	rad

The matrix J_{bx} is a 4-by-4 diagonal matrix with the value 1 in the diagonal which is related to the parameter which we want to constraint from the state vector. In our case, and since we want to constraint speed, J_{bx} is

$$J_{bx} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{43}$$

The same goes for matrix J_{bu}, it is a 2-by-2 diagonal matrix. In this specific case, it is an identity matrix because we want to constraint both acceleration and steering angle. This is the ACADOS implementation, leading to the following OCP

$$\begin{aligned} \min_{U\in\mathbb{R}^{2xN-1}} \quad & \sum_{n=1}^{N-1} \frac{1}{2}\left\| y_n - y_{nref} \right\|^2_W + \frac{1}{2}\left\| y^e - y^e_{ref} \right\|^2_{W^e} \\ \text{subject to} \quad & f_{expl}(x,u) = \dot{x}, t \in [0,T) \\ & v_{r_{min}} \leqslant v_{r_k} \leqslant v_{r_{max}}, k = 0, ..., N \\ & \delta_{min} \leqslant \delta_k \leqslant \delta_{max}, k = 0, ..., N-1 \\ & a_{min} \leqslant a_k \leqslant a_{max}, k = 0, ..., N-1 \end{aligned} \tag{44}$$

D. Performance Metrics

It is important to have clearly defined performance metrics in order to be able to improve controller performance and compare it to other control algorithms. The performance metrics used can be divided into three main categories: error metrics, computational performance metrics, and energy/time metrics. The error metrics are the path following and speed following error. The path-following error was computed based on the euclidean distance from the vehicle current point to the closest point in the path,

$$Dist_{euclidean} = \sqrt{d_x^2 + d_y^2}, \tag{45}$$

where $d_x = x_{ref} - X$ and $d_y = y_{ref} - Y$. The speed following error measures the controller ability to precisely follow the speed profile. This error is the difference between the vehicle and reference speed for the current path point. Both these error metrics have a direct relation to the MPC tuning.

A performance metric on vehicle steering change rate is also used to gauge passenger comfort. This metric evaluates the vehicle steering change rate between two consecutive control signals. In other words, vehicle steering change rate is computed by

$$\delta_{CR} = |\delta_{prev} - \delta|, \tag{46}$$

where δ_{CR} is the steering change rate, δ and δ_{prev} are the current and previous iteration steering angles.

For the computational performance metrics, the performance is measured by the time needed for each MPC iteration and the full time needed for the MPC to solve the path following exercise. This should be enough to show evidence on the fastest implementation for on-line implementation.

Energy and time spent by the vehicle to complete a path are also used as metric. The closest the energy and time spent by the vehicle to the theoretical values, the better is the reference following of the MPC.

IV. Experiments

This chapter presents some experiments performed on the controller. First the algorithm is tested and calibrated using an optimization tool (ACADOS) which is able to optimize the problem in real time. Then, the control of the VIENA car is tested under several tracks (pitched path, uphill-downhill and narrow turns), verifying in which conditions the vehicle would enter in slip mode. Restrictions to the speed of the vehicle are imposed to avoid slip conditions.

A. MPC Weights Fine Tuning using ACADOS

As a starting point, the performance of Parrado's implementation is was studied. Figures 8 and 9 compares the path and vehicle speed to the respective references.

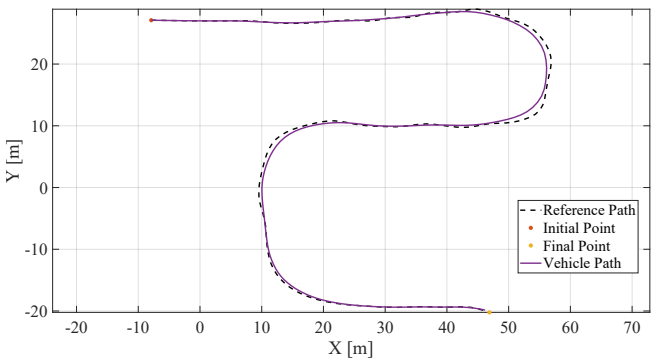

Fig. 8: Parrado: Reference and vehicle path

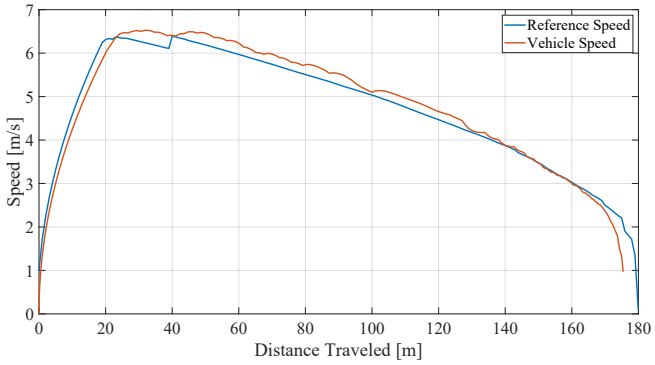

Fig. 9: Parrado: Reference and vehicle speed

From figure 8 it is possible to conclude that the vehicle is capable of successfully following the reference path with, apparently, small tracking error. The fact that the vehicle is able to follow the curve on the inside shows the effects of the MPC's prediction horizon. On the other hand, figure 9 shows that the vehicle presents some difficulties in following the reference speed profile. At around 39 meters of traveled distance there is a jump in the reference speed profile. This is due to the fact that Parrado had a condition implemented in which if the vehicle was more than 5 meters from the expected position at that time instant, then a new speed profile was issued, since it meant that the speed profile was not being followed accurately. In order to improve the controller performance, further tuning on the weight matrices was implemented. This eliminated the need for the speed profile recomputation, as shown in Figure 10.

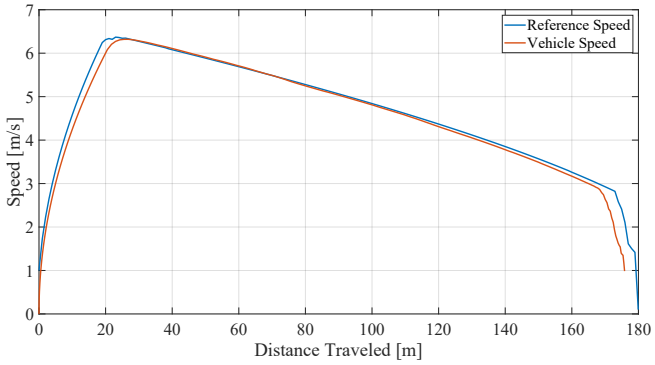

Fig. 10: Parrado Tuned: Reference speed vs vehicle speed

Table II presents the error metrics from Parrado's implementation and its tuned version.

TABLE II: Parrado Error Metrics

	e_{pos}^{max} [m]	e_{pos}^{min} [m]	e_{pos}^{mean} [m]
Parrado	1.8723	0	0.8846
Parrado Tuned	1.7911	0	0.8382
	e_{speed}^{max} [m/s]	e_{speed}^{min} [m/s]	e_{speed}^{mean} [m/s]
Parrado	1	0	0.1917
Parrado Tuned	1	0	0.0678

From this table it is possible to understand that the path following capabilities are not hindered with the new set of matrices, while the reference speed following is greatly improved.

In terms of Computational performance, the results are presented in table III. $\underline{\Delta}t_{MPC}$ is the minimum MPC iteration time, $\bar{\Delta}t_{MPC}$ is the mean MPC iteration time and $\sum \Delta t_{MPC}$ is the total MPC time.

TABLE III: Parrado Computational Performance Metrics

	$\underline{\Delta}t_{MPC}$ [s]	$\bar{\Delta}t_{MPC}$ [s]	$\sum \Delta t_{MPC}$ [s]
Parrado	0.1902	0.3069	63.7707
Parrado Tuned	0.1978	0.2654	56.5256

The computational performance is not affected by simply changing the MPC matrices. The small difference observed may be attributed to the load on the computer that ran the simulation and on the fact that the speed profile is not recomputed. However, taking into account the fact that the mean MPC iteration computation time is higher than the timestep considered for the MPC implementation, which is 0.2 seconds, it really limits this implementation as an on-line algorithm. As a matter of fact, the vehicle takes 41.2 seconds to complete the whole path (excluding MPC computation time) while the MPC alone requires 63.8 seconds to generate all the control variables for the whole path.

As for the energy and time expended by the vehicle, the results are presented in IV.

TABLE IV: Parrado Energy Consumption & Time

		Target Values	Real Values
Parrado	Energy [kWh]	0.5060	0.5356
	Time [s]	44.6519	41.20
Parrado Tuned	Energy [kWh]	0.5188	0.5105
	Time [s]	44.6519	42.6

The closer the vehicle is capable of following the speed profile, the closer will be the target and real values for both time and energy. As such, for the Parrado implementation, since there is an overshoot in speed, the vehicle completes the path quicker at an higher energy expense. The tuned version presents lower relative errors. As expected, since the vehicle is now able to follow the speed profile more closely, both the time and energy spent on the path following exercise are closer to the target values.

Now the ACADOS implementation is introduced. The main objective of this implementation was to decrease the amount of computational time required to compute the control signals. Initially the MPC matrices were tuned by a trial and error method, in order to obtain a reasonable performance from the controller, and only then tuned using a grid search method. Table V presents the error metrics for ACADOS.

TABLE V: ACADOS Error Metrics

	e_{pos}^{max} [m]	e_{pos}^{min} [m]	e_{pos}^{mean} [m]
ACADOS	1.000	0	0.3750
ACADOS Tuned	0.9558	0	0.3680
	e_{speed}^{max} [m/s]	e_{speed}^{min} [m/s]	e_{speed}^{mean} [m/s]
ACADOS	0.8288	0	0.0955
ACADOS Tuned	0.8288	0	0.0948

As can be seen from Table V, the path following error decreases significantly, while the speed following error only

decreases a small amount in terms of mean error. As mentioned in section III-C, the ACADOS formulation eliminates the constraint on steering angle change rate, since with accurate tuning, this constraint can be followed. Figure 11 presents the steering angle and steering change rate for the whole path along with the limits on these variables.

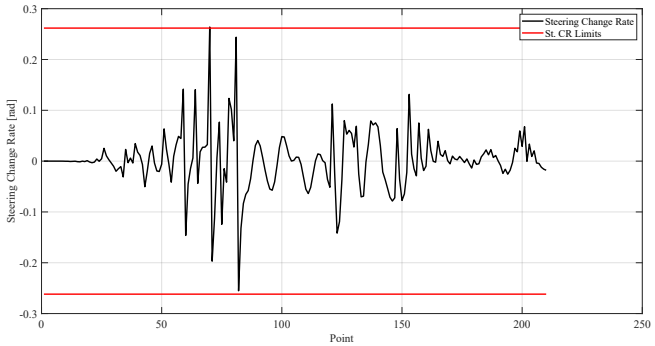

Fig. 11: ACADOS Tuned: Steering Change Rate

This figure shows that the controller maintains a smooth steering throughout the whole path, even though the controller no longer has a constraint on steering angle change rate. Table VI compares the ACADOS computational performance to Parrado's implementation.

TABLE VI: ACADOS: Computational Performance Metrics

	Δt_{MPC} [s]	$\bar{\Delta} t_{MPC}$ [s]	$\sum \Delta t_{MPC}$ [s]
Parrado	0.1902	0.3069	63.7707
ACADOS	1.629×10^{-4}	2.876×10^{-4}	0.0541

The ACADOS implementation is capable of reducing the computational time up to four orders of magnitude, making this an algorithm that could be implemented in real time.

B. Paths Including Slopes

In this section, the controller performance on non-flat paths is discussed. To do so, the tuned ACADOS implementation is used and the path considered is the same from the previous experiment, except with pitch added to it. Figures 12 and 13 show the comparison between reference and attained path and speed.

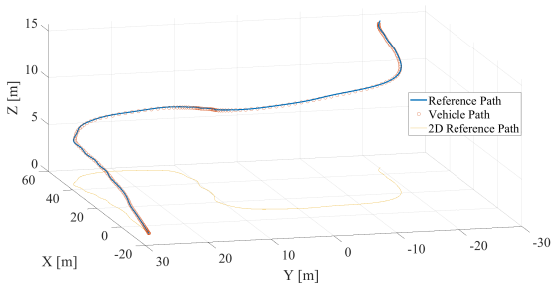

Fig. 12: Pitched Path: Reference and vehicle path

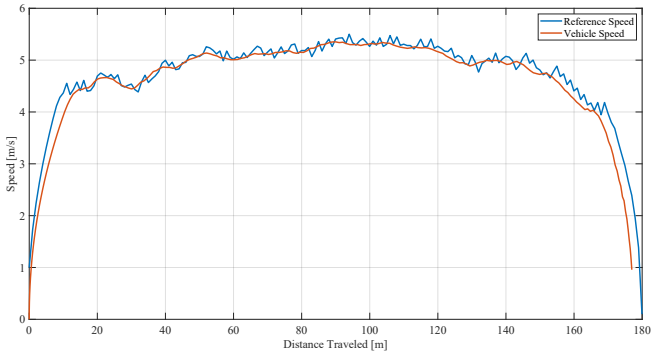

Fig. 13: Pitched Path: Reference and vehicle speed

Based on Figure 12, it is possible to verify that the vehicle is able to closely follow the reference path, without any major difference from the flat paths. However, due to the rolling resistance and gravitational forces, the speed profile in Figure 13 is completely different from the previously presented speed profiles. Since the vehicle is now going uphill, the effects of these forces are more prevailing, and the vehicle cannot simply coast from a high speed to a stop, as it usually did. On the other hand, the lack of smoothness in the speed profile is due to a lack of a smooth constraint on the OCP of the speed profiler.

Table VII shows the error metrics from this experiment.

TABLE VII: Pitched Path: Error Metrics

	e_{pos}^{max} [m]	e_{pos}^{min} [m]	e_{pos}^{mean} [m]
Pitched Path	0.9608	0	0.3112
	e_{speed}^{max} [m/s]	e_{speed}^{min} [m/s]	e_{speed}^{mean} [m/s]
Pitched Path	0.8693	0	0.1340

From Table VII it is possible to conclude that the path following errors remain close to those from the flat path. However, there seems to be an increase in the speed following error. This can be due to the road pitch and the fact that the MPC model does not consider road pitch lead to a discrepancy between the desired acceleration by the controller and actual vehicle acceleration.

An important note on this experiment: maximum pitch that the vehicle can follow is around 10 degrees. If the path pitch goes above this point, the motor maximum torque is exceeded, and the vehicle does not move, or may even start to move backwards. As such, the pitch considered of 5 degrees leads to a road pitch of around 9%. The maximum pitch allowed for highways is 6%, which prompted the use of the 5 degrees of road grade.

C. Speed at Narrow Turns

In [8], a dynamic limit on steering angle is described. It is known that for higher speeds, slip is more prone to happen under high steering angle. As such, to avoid these special cases, a speed dependent limit on steering angle can be implemented to increase the safety of the controller. The steering limit is obtained based on a tire model and is presented in Figure 14.

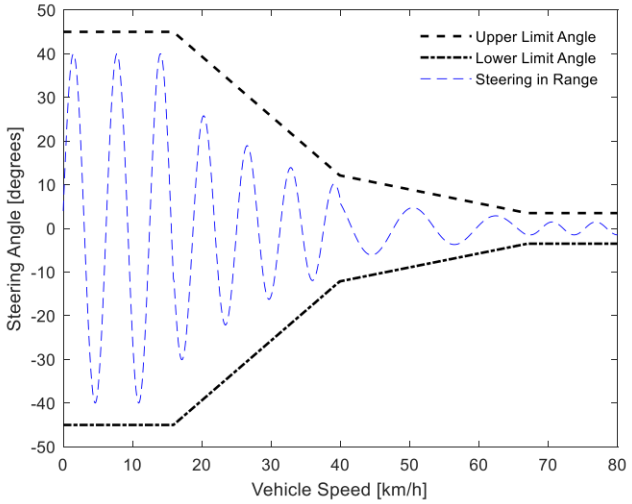

Fig. 14: Speed dependent Steering Limit, from [8]

In order to prove that the slip conditions implemented in the speed profiler are producing reliable results a new test is proposed. Using a 'spiral' path, it is possible to describe a path that sweeps the whole steering angle spectrum, or in other words, all the corner radius that may cause slip conditions. Then the speed profiler is adapted to build the minimum time speed profile instead of the minimum energy speed profile. This is important to guarantee that the speed for each of the steering angles is the maximum speed possible, not obeying any type of energy constraint. Now, considering that the steering angle δ for our kinematics model can be obtained via $\delta = \arctan \frac{L}{R}$, where L is the vehicle wheelbase and R is the corner radius, it is possible to plot the steering angle of this path versus the maximum speed retrieved by the speed profiler. This leads to the plot in Figure 15.

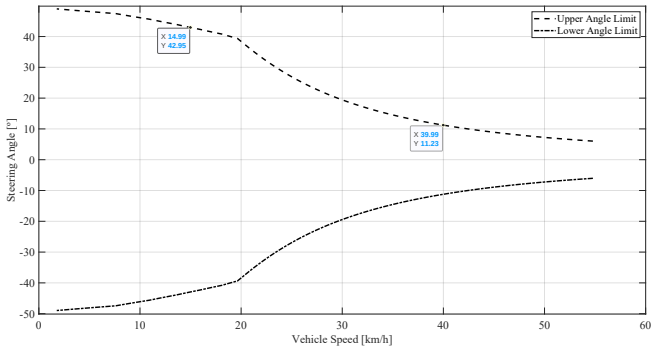

Fig. 15: Steering Angle versus vehicle speed

The variables from figures 14 and 15 are in the same units in order to provide a good comparison. Figure 14 presents a more linear aspect, while the one generated here is not as linear. However, and based on the datatips provided, the limit values of steering angle seem to align between both works. It should be noted that our vehicle has a lower maximum speed, reason for which our dynamic steering limit ends at around 55 km/h.

V. Conclusion and Future Work

This is the follow up on the work from [6], which had already implemented a Model Predictive Controller to solve the path following exercise with the least energy required and developed a model of an electric vehicle. The work now developed led to a new and improved Model Predictive Control controller, which proved to decrease path and speed following error, while keeping the computational time to a minimum. The vehicle kinematic model was updated in order to allow simulations on non-flat, pitched paths. The minimum energy speed profiler was also improved, now taking into account slip conditions, modelling the speed according to the corner in the path. As a final point, MQTT communication was tested in order to allow future implementation in the real VIENA.

From section IV it is possible to understand the level of improvements that ACADOS brought along. Even after re-tuning the MPC from the previous work, the ACADOS implementation is able to produce better results in both path and speed following. The point where ACADOS really shines is in computational time: the OCP solving time drops 3 to 4 orders of magnitude when compared to Parrado's implementation. This in turn allows to implement computationally intense tuning algorithms such as grid search. Finally, the slip conditions on the speed profiler proved to provide good results, modulating the vehicle speed according to the corner in the path.

On the other hand, better models could lead to better performance on a MPC level. The inclusion of more reliable models within the MPC can lead to having better results in the path tracking exercise. Another point that could be improved is the computational time of the reference speed profile generation. The noise effects on the path following exercise need to be studied more accurately. The elimination of noise from sensor data can be more productive and efficient than to 'train' the MPC to account for noise. These are some of the points that can be identified, which, after being improved, the implementation on the real VIENA may become real.

References

[1] Brian Paden, Michal Čáp, Sze Zheng Yong, Dmitry Yershov, and Emilio Frazzoli. "A survey of motion planning and control techniques for self-driving urban vehicles". *IEEE Transactions on Intelligent Vehicles*, 1(1):pages 33–55, 2016.

[2] Johan Wahlström, Isaac Skog, and Peter Händel. "Detection of dangerous cornering in GNSS-data-driven insurance telematics". *IEEE Transactions on Intelligent Transportation Systems*, 16(6):pages 3073–3083, 2015.

[3] Eduardo F. Mello and Peter H. Bauer. "Energy-optimal speed trajectories between stops". *IEEE Transactions on Intelligent Transportation Systems*, 21(10):pages 4328–4337, 2020.

[4] Bruno Tibério. MSc Thesis," An online filter study for inertial properties estimation based on low-cost sensors". https://fenix.tecnico.ulisboa.pt/downloadFile/1126295043835604/Thesis.pdf, 2007.

[5] Fen Lin, Minghong Sun, Jian Wu, and Chengliang Qian. "Path tracking control of autonomous vehicle based on nonlinear tire model". *Actuators*, 10:pages 242, 09 2021.

[6] Filipe Parrado. MSc Thesis, "Steering and speed control for autonomous electric vehicles". 2021.

[7] Robin Verschueren, Gianluca Frison, Dimitris Kouzoupis, Jonathan Frey, Niels van Duijkeren, Andrea Zanelli, Branimir Novoselnik, Thivaharan Albin, Rien Quirynen, and Moritz Diehl. "acados: a modular open-source framework for fast embedded optimal control", 2019.

[8] Vu Minh, Reza Moezzi, Jindřich Cýrus, and Jaroslav Hlava. "Model predictive control for autonomous driving vehicles. *Electronics*, 10:pages 2593, 10 2021.