2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Real-time optimal control of an autonomous RC car with minimum-time maneuvers and a novel kineto-dynamical model

Edoardo Pagot[1], Mattia Piccinini[1] and Francesco Biral[1]

***Abstract*— In this paper, we present a real-time non-linear model-predictive control (NMPC) framework to perform minimum-time motion planning for autonomous racing cars. We introduce an innovative kineto-dynamical vehicle model, able to accurately predict non-linear longitudinal and lateral vehicle dynamics. The main parameters of this vehicle model can be tuned with only experimental or simulated maneuvers, aimed to identify the handling diagram and the maximum performance G-G envelope. The kineto-dynamical model is adopted to generate on-line minimum time trajectories with an indirect optimal control method. The motion planning framework is applied to control an autonomous 1:8 RC vehicle near the limits of handling along a test circuit. Finally, the effectiveness of the proposed algorithms is illustrated by comparing the experimental results with the solution of an off-line minimum-time optimal control problem.**

I. Introduction

The design of real-time motion planning and control methods for autonomous racing vehicles is currently gaining an increasing amount of interest. Research has been carried out to develop robust algorithms capable of driving racing vehicles at the limits of handling [1], [2], [3], [4], but also to enable passenger cars to autonomously plan or track feasible emergency maneuvers [5], [6]. The two research lines share the same challenges and may mutually benefit of the results achieved.

Several techniques have been proposed regarding the on-line trajectory planning for vehicles and robots [7], [8]. Model-predictive control (MPC) is an optimization technique that may be adopted to compute, in a receding-horizon fashion, the optimal vehicle states and trajectory to complete a given circuit in the minimum time. In this context, MPC is named *economic* to highlight that it is not formulated as an optimal tracking problem of a desired equilibrium [9], but the cost is designed to search for optimal non linear manoeuvres. MPC is based on a mathematical model that describes the behavior of the car, and the model becomes heavily non-linear [10] when it has to represent a vehicle driven close to its maximum performance limits. Complex vehicle models complicate the solution of the numerical optimization problems involved in MPC, yielding larger and highly non linear systems of equations with more potential local minima. As a consequence, to practically achieve real-time feasibility, the vehicle model used for path planning with MPC needs to be the result of a wise trade-off between model complexity and numerical efficiency.

In [11] an RC vehicle was controlled in real-time with a model-predictive contouring approach, which minimized the vehicle path deviations from the circuit middle line and maximized the travelled distance. The resulting NLP was solved by means of local approximations with QPs. However, the vehicle trajectory was different from the time optimal one, since the horizons were relatively short and the center line was partially tracked. A linear parameter varying MPC strategy was used in [12] for the on-line tracking of trajectories computed off-line with optimal control. This method was applied to control a 1:10 RC vehicle. [13] employed a viability kernel approach to guarantee recursive feasibility and limit the number of trajectories generated by a high-level planner for an RC vehicle. However, their technique results in a safer and more conservative driving style.

Other authors used methods different from MPC for path planning with racing cars. [14] found a minimum curvature path by solving a QP problem and then optimized the velocity profile with a forward-backward method that considered speed-dependent acceleration limits. A similar approach was adopted in [15], in which the path and the speed profile were computed iteratively. However, in [14] and [15] the race-line was computed off-line and it was different from the one that minimizes lap time. In [16] the authors employed the same experimental setup used in this work, and performed on-line path planning by creating a set of clothoids and choosing the one with minimum length. The main limitation of this method is that it produces minimum time maneuvers only if the vehicle speed is constant.

[2] developed a hierarchical NMPC structure in which the high-level planner solved on-line minimum time optimal control (OC) problems to drive a racing vehicle model. They validated the framework only in a simulation environment. To the best of the authors' knowledge, on-line minimum time NMPC has never been applied to a real vehicle.

The aim of this paper is to present a real-time motion planning and control framework capable of driving an autonomous vehicle near the limits of handling with minimum time trajectories. We propose a non-linear MPC framework that solves on-line minimum time OC problems. The authors of [17], [5], [18] performed motion planning with kinematic point-mass models that only consider acceleration limits and do not take into account the yaw rate dynamics of the real vehicle. To this end, we introduce a novel kineto-dynamical vehicle model, that enables to

[1]Edoardo Pagot, Mattia Piccinini and Francesco Biral are with the Department of Industrial Engineering, University of Trento, Trento, Italy
edoardo.pagot@unitn.it,
mattia.piccinini@unitn.it,
francesco.biral@unitn.it

accurately capture the longitudinal, lateral and yaw dynamics of the vehicle while being computationally efficient for NMPC. One of the main advantages of the proposed model is the possibility to use directly the steering angle as input, in the full vehicle operating conditions, without the need of any low-level controller or correction. The trajectory of a 1:8 RC vehicle, controlled with our motion planning algorithms along a test circuit, compares favorably with the one obtained by solving off-line a minimum-time optimal control problem. A video of the experiments with the real vehicle can be found at `https://youtu.be/HADLEr5eTj0`. Additionally, our NMPC framework is based on an indirect approach, which is quite new for this type of applications, being the direct method used by [1], [2], [3], [5] to solve on-line optimal control problems.

II. Vehicle Model for Motion Planning

The main requirement for a car dynamic model used in MPC is the ability to accurately capture the longitudinal and lateral dynamics of the vehicle in its entire operating range, maintaining the simplest structure possible, and providing output controls that can be directly employed to drive the vehicle. The strategies adopted in the literature are two folds: either use complex mathematical models that involve non-linear tyre models (either single track or double track models [10]), or simplify the dynamics to a kinematic model (*e.g.* a point mass) subject to acceleration constraints (see [19], [17], [2]). The first approach forces to adopt very short planning horizons due to high computational effort, while the second method does not consider at all the steering behaviour of the vehicle.

We present a novel kineto-dynamical approach, able to model the non-linear dynamics of the vehicle near the limits of handling while being computationally efficient for NMPC. Our model includes the steering behaviour (*i.e.* oversteer/understeer tendency) of the car in its entire operating range.

The foundations of the proposed model come from the non-linear equations of a single track model under the quasi steady-state condition [19]: we assume that motion is a sequence of steady-state conditions in which the longitudinal acceleration can be different from zero. By solving the corresponding lateral dynamic equations [10], the steady-state steering characteristic of a vehicle may be expressed as

$$\delta(t) - \frac{\Omega_{ss}(t)}{v_x(t)} L = K_{us}(a_y(t)) \tag{1}$$

with δ, Ω_{ss}, L and K_{us} being the steering angle, the steady-state yaw rate, the wheelbase and the understeer gradient, respectively. In general, K_{us} is a function of the lateral acceleration a_y, longitudinal acceleration a_x, and forward speed v_x. However, in this work we consider K_{us} a non-linear function of only the lateral acceleration a_y. Equation (1) describes the quasi steady-state behaviour of the yaw rate Ω, and the evolution of Ω is modeled as a first order system:

$$\tau_\Omega \dot{\Omega}(t) + \Omega(t) = \Omega_{ss}(t) = v_x(t)\left(\frac{\delta(t) - K_{us}(a_y(t))}{L}\right) \tag{2}$$

where the time constant τ_Ω is, in general, function of the forward speed v_x and longitudinal acceleration. Equation (2) defines how the yaw velocity of the car changes in time, while the longitudinal speed evolves according to the longitudinal dynamic equation:

$$\dot{v}_x(t) = a_x(t) - \frac{k_D}{m} v_x^2(t) - g \sin\theta_{\text{sl}}(t) - c_r\, v_x(t) \tag{3}$$

Equation (3) includes the contribution of the aerodynamic drag (proportional to the square of the forward speed v_x via the aerodynamic drag coefficient k_D), the road slope $\theta_{\text{sl}}(t)$, and other frictional effects mainly due to the drive-line and rotational friction (c_r). The vehicle mass is named m, while g is the gravitational acceleration.

The model inputs are the longitudinal acceleration a_{x0} and the steering angle δ_0, which is assumed to be equal for both front wheels. Actuation dynamics are simulated as first order systems, with time constants τ_a and τ_δ:

$$\begin{cases} \tau_a \dot{a}_x(t) + a_x(t) = a_{x0}(t) \\ \tau_\delta \dot{\delta}(t) + \delta(t) = \delta_0(t) \end{cases} \tag{4}$$

The model is completed with the equations that describe the vehicle position and orientation with respect to the circuit middle line in curvilinear coordinates [20], [21], as depicted in Fig. 1. The curvilinear abscissa of the road is indicated with ζ, while n is the lateral displacement of the vehicle center of mass with respect to the middle line. The yaw angle of the car in the absolute reference frame X_0Y_0 is named ψ, while the angle θ provides the orientation of the line that is locally tangent to the road center. The quantity $\psi - \theta$ is called relative yaw angle ξ. Road curvature is expressed with the function $\kappa_r(\zeta)$.

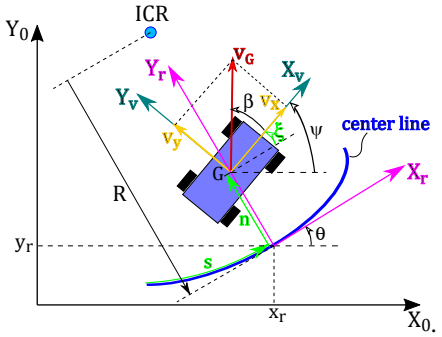

Fig. 1. Curvilinear coordinates $\{\zeta, n, \xi\}$.

The temporal evolution of the vehicle pose can be formulated in curvilinear coordinates as a function of the vehicle longitudinal velocity $v_x(t)$ (expressed in the vehicle non-inertial reference frame) and yaw rate $\Omega(t)$ [21]:

$$\begin{cases} \dot{\zeta}(t) = v_\zeta(t) = -\frac{v_x(t)\cos(\xi(t))}{n(t)\kappa_r(\zeta(t))-1} \\ \dot{n}(t) = v_x(t)\sin(\xi(t)) \\ \dot{\xi}(t) = \Omega(t) + \frac{v_x(t)\cos(\xi(t))\kappa_r(\zeta(t))}{n(t)\kappa_r(\zeta(t))-1} \end{cases} \tag{5}$$

The kineto-dynamical model equations are (4), (3), (2) and (5), its states are $\boldsymbol{x} = \{a_x, \delta, v_x, \Omega, \zeta, n, \xi\}$, while the controls are $\boldsymbol{u} = \{a_{x0}, \delta_0\}$. This model is able to accurately capture non-linear vehicle dynamics even if tire forces are not explicitly modeled, which simplifies the formulation and does not require to directly measure or identify tire parameters.

III. Model Predictive Control Architecture

A. Optimal Control Problem Formulation

The structure of the minimum-time optimal control problem used for MPC is here defined. The target is to determine the controls $\boldsymbol{u} \in \mathcal{U}$ that minimize the *cost functional* $\mathcal{J}$

$$\min_{\boldsymbol{u}\in\mathcal{U}} \left(\mathcal{J}\right) = \sum_{i=1}^{n_x} B_i\big(\boldsymbol{x}_i(0) - \boldsymbol{x}_{i0}\big)^2 + \int_0^T w_T \mathrm{d}t \quad \text{subject to} \quad \begin{cases} \dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \\ \boldsymbol{c}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \geq 0 \end{cases} \tag{6}$$

The argument of the summation composing the first term of (6) is referred to as *Mayer term*, and it is used to impose soft constraints on the initial conditions for model states $\boldsymbol{x}$, whose total number is n_x. Final conditions for $\boldsymbol{x}$ are not constrained. The minimization of the final time T is obtained with the integral term in (6), also known as *Lagrange term*. B_i and w_T are tunable parameters used to trade off the optimization objectives. The minimization is subject to the system of differential equations $\boldsymbol{f}$ describing the vehicle model outlined in Section II, and to a set of inequality constraints $\boldsymbol{c}$. In the OC problem, the road curvilinear abscissa ζ ranges in the known interval $[\zeta_i, \zeta_f]$, while time t varies from 0 to a final unknown value T. On this account, it is advantageous to use ζ as the independent variable [21], rather than t. This space-domain transformation is based on the relation $\frac{\mathrm{d}\zeta}{\mathrm{d}t} = v_\zeta$ of (5), and it enables to rewrite the OC problem as

$$\min_{\boldsymbol{u}\in\mathcal{U}} \left(\mathcal{J}\right) = \sum_{i=1}^{n_x} B_i\big(\boldsymbol{x}_i(\zeta_i) - \boldsymbol{x}_{i0}\big)^2 + \int_{\zeta_i}^{\zeta_f} \frac{w_T}{v_\zeta} \mathrm{d}\zeta \quad \text{subject to} \quad \begin{cases} \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}\zeta} = \boldsymbol{f}(\boldsymbol{x}(\zeta), \boldsymbol{u}(\zeta))/v_\zeta \\ \boldsymbol{c}(\boldsymbol{x}(\zeta), \boldsymbol{u}(\zeta)) \geq 0 \end{cases} \tag{7}$$

The inequality constraints $\boldsymbol{c}$ are defined as

$$\begin{cases} n(\zeta) - \dfrac{W}{2} \geq -\mathrm{M}_R, & \text{(8a)} \\ n(\zeta) + \dfrac{W}{2} \leq \mathrm{M}_L, & \text{(8b)} \\ a_{x0_{\min}} \leq a_{x0}(\zeta) \leq a_{x0_{\max}}, & \text{(8c)} \\ \delta_{0_{\min}} \leq \delta_0(\zeta) \leq \delta_{0_{\max}}, & \text{(8d)} \\ \mathrm{p}(a_x(\zeta))\Big[\Big(\dfrac{a_x(\zeta)}{a_{xM}}\Big)^2 + \Big(\dfrac{a_y(\zeta)}{a_{yM}}\Big)^2\Big] + \\ -\mathrm{n}(a_x(\zeta))\Big[\Big(\dfrac{a_x(\zeta)}{a_{xm}}\Big)^2 + \Big(\dfrac{a_y(\zeta)}{a_{yM}}\Big)^2\Big] \leq 1 & \text{(8e)} \end{cases}$$

All the inequalities in (8) hold $\forall \zeta \in [\zeta_i, \zeta_f]$. Constraints (8a) and (8b) force the vehicle to stay within the circuit margins. W is the vehicle track width, while M_R and M_L - which may be constant or functions of ζ - are the distances from the road center line to the right and left margins, respectively. (8c) and (8d) impose limitations on the control inputs of the model. (8e) models the *G-G diagram*, enforcing limits on the longitudinal, lateral and combined performance of the car. In (8e), it is $a_y(\zeta) = \Omega(\zeta) v_x(\zeta)$, while $\mathrm{p}(\cdot)$ and $\mathrm{n}(\cdot)$ are obtained from the regularized sign function:

$$\begin{cases} \mathrm{p} := x \mapsto \frac{\sin(\arctan(x/h_t))+1}{2} \\ \mathrm{n} := x \mapsto \frac{\sin(\arctan(x/h_t))-1}{2} \end{cases} \tag{9}$$

with h_t being a regularization factor. The functions in (9) enable to model the combined acceleration and braking performance with ellipses having different sizes. The factors $\{a_{xM}, a_{xm}, a_{yM}\}$ may be either constant or functions of v_x. The OC problem (7) is solved in a receding-horizon fashion, and the implementation of the MPC framework is summarized in Algorithm 1.

B. Optimal Control Problem Solution

The optimal control problems at every MPC step are solved using an indirect method based on the Pontryagin Minimum Principle (PMP). We employed a software tool called PINS during the entire procedure starting from the OC problem formulation to the OC problem solution. PINS is a tool developed by E. Bertolazzi, F. Biral and P. Bosetti at the Department of Industrial Engineering of the University of Trento. Thanks to its MapleSoft Maple™ interface-library, PINS enables the user to write the equations of the dynamical system, the target functional and the constraints symbolically; then, PINS automatically formulates the problem (using the options prompted by the user), generates the C++ code and obtains the numerical solution to the OC problem. One of key features of PINS is the approximation of path constraints with penalty or barrier functions in order to eliminate the inequality constraints: this permits to always calculate the controls explicitly, or at least by making use of a sub-iterative scheme. The interested reader can find more information about the OC problem formulation and the indirect approach in [22]. In [23] PINS developers offered a comparison between the three most common approaches for the resolution of OC problems, i.e. the direct method, the indirect method and dynamic programming, and shown the excellent performances of PINS (and indirect method) in terms of problem solution time when handling non-linear dynamical systems on relatively large horizons.

IV. Experimental Setup

A. Autonomous Vehicle Platform

eRumby is the experimental vehicle employed in this work. eRumby is based on a 1:8 scale commercial RC car which features an All Wheel Drive (AWD) powertrain (an electric three-phase brushless motor plus three differentials)

and an electric servo motor to actuate the steering mechanism; however, during all the experiments presented here, the vehicle was set in Rear Wheel Drive (RWD) configuration. The modifications brought to the eRumby prototype include one IMU, one optical encoder for each wheel (mounted by means of custom 3D-printed uprights), a Raspberry Pi 3 and two Arduino Mega. The high-level control code runs on the Raspberry board, while the two Arduino Mega are employed for low-level tasks, namely to drive the actuators and read the sensors.

The experiments were performed inside an arena (roughly 10 m $\times$ 10 m) which is instrumented with an OptiTrack system. By means of several cameras and reflective markers, this system accurately measures the position and orientation of one or more rigid bodies in the 3D space (6 DoF-motion) with high frequency (up to 100 Hz). One marker was mounted on the top of the eRumby vehicle in order to track its position and orientation on the 2D-plane representing the floor of the arena.

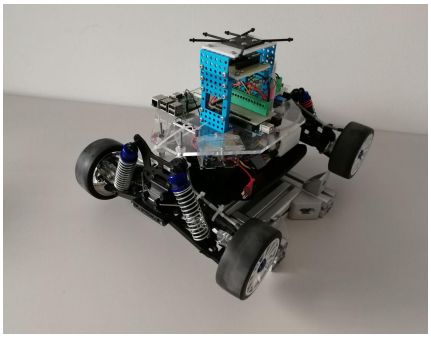

Fig. 2. The eRumby experimental vehicle. The black plastic frame that holds the markers for the OptiTrack system can be seen on the top of the car.

B. Control Framework Implementation

Aiming at the best performance in terms of computing time, we chose to run the MPC algorithm on a 2019 MacBook Pro laptop equipped with a 2.6 GHz 6-Core Intel Core i7 processor. The computed optimal controls were sent to the Raspberry mounted on the experimental vehicle via Wi-Fi, using the User Datagram Protocol (UDP). The OptiTrack system in our facility was connected to a Windows PC, which sent the vehicle pose to the laptop via UDP Ethernet connection. A scheme of the experimental setup is shown in Fig. 3.

The MPC algorithm running on the laptop is shown in Algorithm 1: the OC problem described in Section III is solved with a frequency $1/T_s^{MPC}$, with $T_s^{MPC} = 100$ ms. Despite the curvilinear coordinate ζ being the independent variable of the OC problem, we can set PINS to return the problem-time as a post-processing variable. The problem-time is computed by numerical integration of $\frac{1}{\dot{\zeta}} = \frac{\mathrm{d}t}{\mathrm{d}\zeta}$ with respect to $\mathrm{d}\zeta$, where $\dot{\zeta}$ was given in (5).The optimal states $\boldsymbol{q} = \{\delta, v_x\}$ are used as reference values for the steering and traction actuators, and they are hereafter named *low-level controls*. The optimal low-level controls

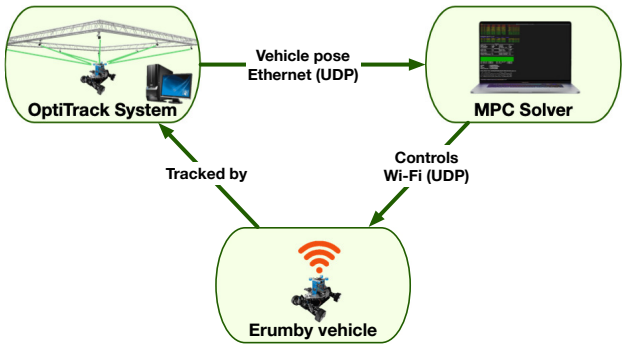

Fig. 3. Communication scheme of the experimental setup.

Algorithm 1: MPC Algorithm

```
while MPC Computing ON do
    Get vehicle pose from OptiTrack
    if time == n T_s^MPC, n ∈ ℕ then
        Find updated curvilinear coordinates along track from the vehicle pose
        Set new initial conditions for the NMPC step
        Solve the Minimum Time OC problem from the current position to the horizon end, using PINS
        Interpolate the new vector q = {δ, v_x} from t_CPU to t_CPU + T_s^MPC
        Create a vector q_s by sampling q with rate 1/T_s^ctrl
        Send the updated control vector q_s to the vehicle via UDP
```

from problem-time zero to problem-time t_{CPU} are discarded; t_{CPU} is the CPU time required to solve the OC problem of the current MPC step. This guarantees that no delay is introduced in the low-level controls, and the vehicle uses the optimal controls of the previous MPC iteration until a new OC solution is available. Then, a time horizon of T_s^{MPC} is selected from $\boldsymbol{q}$ with a sampling of $T_s^{ctrl} = 10$ ms, and the extracted samples are stored in a vector $\boldsymbol{q_s}$ and sent to the vehicle.

Algorithm 2 runs on the vehicle Raspberry: the new low-level controls are read every T_s^{ctrl} from the vector $\boldsymbol{q_s}$ and they are fed to the actuators directly. The vector $\boldsymbol{q_s}$ is updated every time a new MPC step is computed. Due to its stock hardware configuration, the eRumby electric motor can be only speed-controlled. A PI low-level controller closes the loop on the motor angular speed using the feedback from the rear wheel encoders, trying to match the prescribed reference for v_x. The speed controller does not take into account the longitudinal slips of the rear wheels, which could be problematic when running on relatively slippery surfaces like the one in the arena that we used for our experiments. In Section VI we discuss some possible improvements about the aforementioned issue.

C. Handling and Performance Identification

The parameters of the model presented in Section II need to be estimated by means of suitable experimental tests. A maneuver with sinusoidal steering angle δ and constant speed v_x was carried out in order to identify the understeer gradient $K_{us}(a_y)$ and the time constant τ_Ω. The aim of this test is to assess the steering characteristic of the vehicle in almost

Algorithm 2: Control Interpolation Algorithm

while *Autonomous Mode ON* **do**
  Get low-level control vector $\boldsymbol{q_s}$ via UDP
  Check if $\boldsymbol{q_s}$ has been updated
  if $\boldsymbol{q_s}$ *has been updated* **then**
    Reset counter: set i=1
  if *time* == $nT_s^{ctrl}, n \in \mathbb{N}$ **then**
    Extract controls from i^{th} position of $\boldsymbol{q_s}$
    Send the new controls to the vehicle low-level controllers
    Increment counter: set i = i+1
  Low-level controller sets the requested steering servo angle
  Low-level controller tracks the requested wheel angular speed

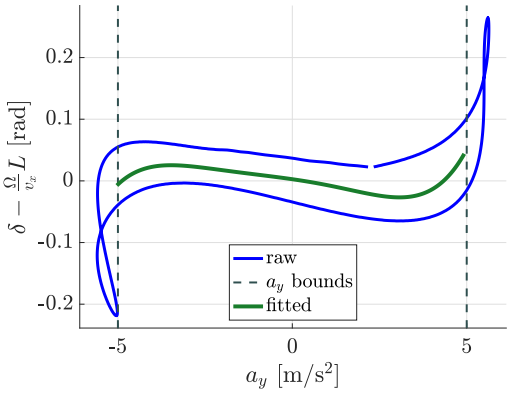

Fig. 4. Handling diagram resulting from a sinusoidal steering maneuver, and polynomial fitting as a function of a_y. Raw data were low-pass filtered.

steady-state conditions, which may be reached if the sine frequency f_s is adequately low. Due to the space limitations of our arena, v_x was limited to 2.5 m/s, while a value of 0.09 Hz was chosen for f_s. The steering law used for the test was $\delta(t) = \delta_A \sin(2\pi f_s t)$, with $\delta_A = 35$ deg. The static steering offset due to tire asymmetries was compensated. The resulting *handling diagram* (HD) [10] is depicted in Fig. 4. The HD shows that the vehicle is oversteering for lateral accelerations a_y in the approximate range $[-4, 4]$ m/s^2, while it is understeering for higher values of $|a_y|$. For $|a_y| > 5$ m/s^2 the vehicle becomes excessively understeering, meaning that, even if the steering angle δ is increased, the yaw rate Ω and the lateral acceleration do not increase significantly. In view of this, we limit a_y in $[-5, 5]$ m/s^2 which is anyway close to the handling limits (roughly $5.5-6$ m/s^2). Polynomial fitting was employed to model the HD with the function $K_{us}(a_y)$ of (1). [16] used a line to fit the HD for the same vehicle, but this approximation is acceptable only for relatively small values of a_y. The HD curve shown in blue in Fig. 4 does not pass through the origin of the graph, implying that Ω is not zero when the sinusoidal δ gets zero. The time constant τ_Ω of (2) was tuned to model this delay between δ and Ω.

The speed-dependant G-G diagram of a vehicle was identified in [19], [2] using a numerical optimization approach, which requires an accurate model and the identification of all its parameters, including tyre data. In contrast to this method, we estimate the G-G diagram (8e) with pure acceleration, braking and sinusoidal steering

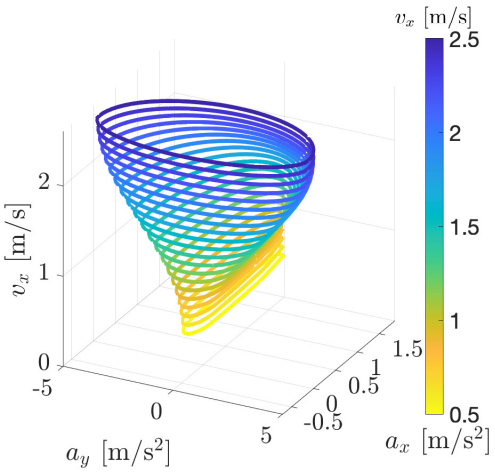

Fig. 5. G-G diagram fitted with variable-size ellipses as a function of v_x.

maneuvers carried out at different speeds. This G-G diagram identification technique is partly based on what [24] proposes with simulated maneuvers. The parameters $\{a_{xM}, a_{xm}, a_{yM}\}$ were fitted using polynomials as a function of v_x. The fitted G-G diagram is shown in Fig. 5.

V. Experimental Results

With the purpose of validating the non-linear MPC algorithm, we applied the entire framework to control the autonomous vehicle prototype presented in Section IV-A. We propose to compare the results obtained in the experiments with the ones provided by an off-line solution of the OC problem (7), considering an entire lap and the same problem parameters. The off-line OC solution is taken as a reference of optimality, in terms of minimum lap time and state trajectories.

In the solution of each receding-horizon MPC step, we used a fixed horizon of 30 m, which covers about 60% of the entire circuit (whose length is 50.7 m). The curvilinear abscissa ζ was discretized with steps of 0.1 m, so that each MPC horizon contains 300 points.

In the tests here presented, the parameter a_{xM} was tuned, as a function of speed, so that the vehicle could not accelerate beyond 2.5 m/s. An entire lap is analyzed, with the vehicle starting at zero initial speed.

Fig. 6 shows the path of the vehicle center of mass. In several points of the circuit the actual path appears to be similar to the one obtained with the off-line optimization. The differences between the two results are due to many factors. First, the HD was identified at a unique speed value (2.5 m/s), while experimental evidence showed that the steering behavior changes as a function of speed and longitudinal acceleration. Second, delays in the UDP communication contribute to a non perfect synchronization between the low-level control and the MPC planning results. Moreover, the operating system running on the Raspberry Pi is not real-time, and the OptiTrack system sometimes fails to correctly localize the vehicle.

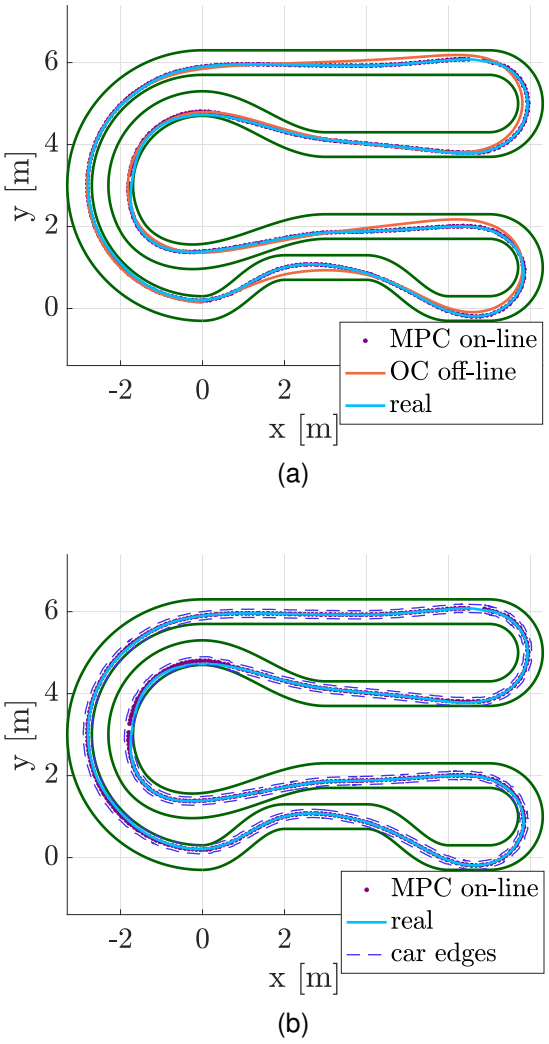

Fig. 6. Comparison of the center of mass path for the real vehicle, the on-line MPC and the off-line OC solutions, without (a) and with (b) car edges.

The speed profile of the vehicle is plotted in Fig. 7a. The MPC algorithm drove the vehicle at top speeds of around 2.5 m/s in the straights, while velocity was decreased to 2 m/s in the two tightest corners. The low-level controller induced a speed overshoot in the first acceleration phase, but overall it provided an adequate speed tracking. The steering angles δ computed by the on-line MPC and the off-line OC are shown in Fig. 7b. The steering angle profile calculated on-line is directly fed to the steering servo, and it features some extra variations with respect to the one obtained off-line. These higher frequency corrections are computed by the MPC algorihm to compensate for communication delays and unmodeled dynamic effects. Fig. 7c shows the lateral acceleration profile, which in corners got very close to the maximum values defined in the HD and G-G diagram constraints. The deviations between the lateral acceleration predicted by the MPC model and the a_y signal measured by the vehicle sensors are mainly due to the non-perfect identification of the steering behavior. As a matter of fact, an identification of the HD as a function of a_y and v_x would be required to drive faster than 2.5 m/s. Fig. 8 depicts how the MPC planner used the G-G diagram. The maximum values of a_x were reached during the initial acceleration phase, while the combined longitudinal-lateral performance was exploited in corners. The motion planner never required negative values of a_x, meaning that aerodynamic and electro-mechanical friction (collected in k_D in (3)) was sufficient to decelerate the vehicle.

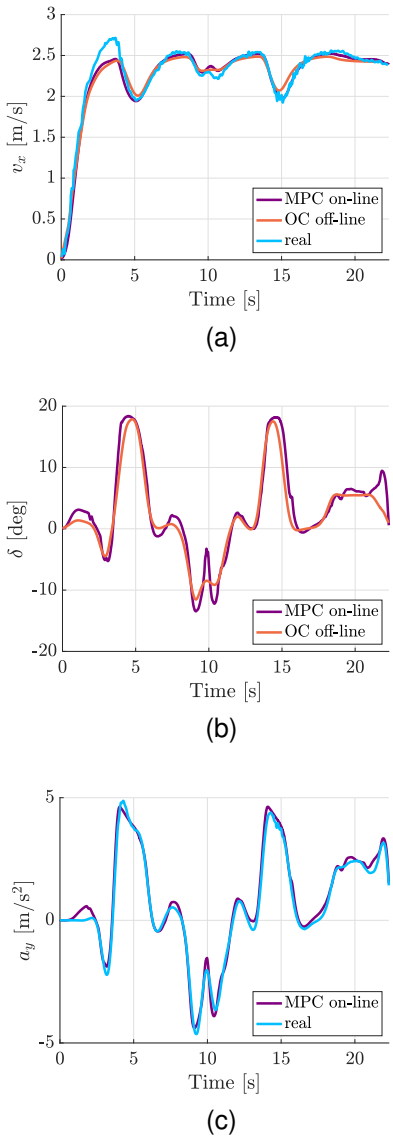

Fig. 7. Speed (a), steering angle (b) and lateral acceleration (c) profiles in time domain.

The overall lap time for the real vehicle was around 0.4 s slower than the one obtained with the off-line OC solution, owing to the differences in trajectory.

All the MPC problems converged, with a mean CPU solution time of 8.90 ms. These low computational times suggest that the kineto-dynamical model used for path planning is numerically efficient, and that the indirect OC solver PINS is suitable for real-time applications. The MPC solution rate

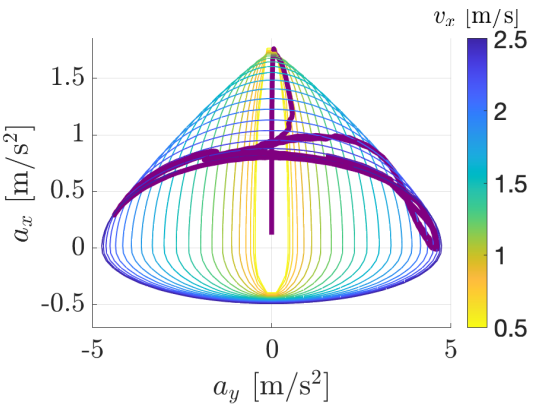

Fig. 8. Exploitation of the maximum performance. Top view of the G-G diagram.

$1/T_s^{MPC}$ could therefore be increased in future work.

VI. Conclusions and Future Work

In this paper, we developed a non-linear MPC framework for the real-time control of an autonomous vehicle at the limits of handling. A novel kineto-dynamical vehicle model was employed for motion planning, which enabled a practical and sufficiently accurate identification of the non-linear longitudinal, lateral and combined vehicle dynamics. The motion planning framework was used to drive an autonomous vehicle prototype, and experimental results revealed that the state trajectories of the real vehicle are close to the time-optimal ones. The low computational times of the receding-horizon MPC steps indicate that the kineto-dynamical model is effective for motion planning, and that the indirect solver PINS is suited for on-line optimal control problems. In future work, the possibility to drive the RC vehicle at higher speed will be investigated by explicitly modeling the effect of v_x and a_x on the handling diagram. Our OptiTrack-covered arena has a limited area: in order to drive the experimental vehicle at a wider speed range, we will move the experiments outdoors. Consequently, the vehicle will be instrumented with a Real-Time Kinematic (RTK) GPS for localization. Moving from the smooth cement floor of the arena to grippy tarmac will also improve the performance of the eRumby low-level speed controller. Nevertheless, the current speed controller will be improved with the addition of a longitudinal slip observer and a traction controller.

References

[1] J. Kabzan, L. Hewing, A. Liniger, and M. N. Zeilinger, "Learning-based model predictive control for autonomous racing," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3363–3370, Oct 2019.

[2] T. Novi, A. Liniger, R. Capitani, and C. Annicchiarico, "Real-time control for at-limit handling driving on a predefined path," *Vehicle System Dynamics*, vol. 0, no. 0, pp. 1–30, 2019.

[3] K. Berntorp, R. Quirynen, T. Uno, and S. D. Cairano, "Trajectory tracking for autonomous vehicles on varying road surfaces by friction-adaptive nonlinear model predictive control," *Vehicle System Dynamics*, vol. 0, no. 0, pp. 1–21, 2019.

[4] F. Christ, A. Wischnewski, A. Heilmeier, and B. Lohmann, "Time-optimal trajectory planning for a race car considering variable tyre-road friction coefficients," *Vehicle System Dynamics*, vol. 0, no. 0, pp. 1–25, 2019.

[5] F. Altché, P. Polack, and A. de La Fortelle, "High-speed trajectory planning for autonomous vehicles using a simple dynamic model," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, Oct 2017, pp. 1–7.

[6] S. E. Li, H. Chen, R. Li, Z. Liu, Z. Wang, and Z. Xin, "Predictive lateral control to stabilise highly automated vehicles at tire-road friction limits," *Vehicle System Dynamics*, vol. 0, no. 0, pp. 1–19, 2020.

[7] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 33–55, March 2016.

[8] D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A review of motion planning techniques for automated vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 1135–1145, April 2016.

[9] T. Faulwasser, L. Grüne, and M. A. Müller, "Economic nonlinear model predictive control," *Foundations and Trends® in Systems and Control*, vol. 5, no. 1, pp. 1–98, 2018.

[10] M. Guiggiani, *The Science of Vehicle Dynamics: Handling, Braking, and Ride of Road and Race Cars*, ser. SpringerLink : Bücher. Springer Netherlands, 2014.

[11] A. Liniger, A. Domahidi, and M. Morari, "Optimization-based autonomous racing of 1: 43 scale rc cars," *ArXiv*, vol. abs/1711.07300, 2015.

[12] E. Alcalá, V. Puig, J. Quevedo, and U. Rosolia, "Autonomous racing using linear parameter varying-model predictive control (lpv-mpc)," *Control Engineering Practice*, vol. 95, p. 104270, 2020.

[13] A. Liniger and J. Lygeros, "Real-time control for autonomous racing based on viability theory," *IEEE Transactions on Control Systems Technology*, vol. 27, no. 2, pp. 464–478, 2019.

[14] 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*, vol. 0, no. 0, pp. 1–31, 2019.

[15] N. Kapania, J. Subosits, and J. Gerdes, "A sequential two-step algorithm for fast generation of vehicle racing trajectories," *Journal of Dynamic Systems, Measurement, and Control*, vol. 138, 04 2016.

[16] D. Piscini, E. Pagot, G. Valenti, and F. Biral, "Experimental comparison of trajectory control and planning algorithms for autonomous vehicles," *IECON 2019 - 45th Annual Conference of the IEEE Industrial Electronics Society*, vol. 1, pp. 5217–5222, 2019.

[17] J. R. Anderson, B. Ayalew, and T. Weiskircher, "Modeling a professional driver in ultra-high performance maneuvers with a hybrid cost mpc," in *2016 American Control Conference (ACC)*, July 2016, pp. 1981–1986.

[18] B. Alrifaee and J. Maczijewski, "Real-time trajectory optimization for autonomous vehicle racing using sequential linearization," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, June 2018, pp. 476–483.

[19] M. Veneri and M. Massaro, "A free-trajectory quasi-steady-state optimal-control method for minimum lap-time of race vehicles," *Vehicle System Dynamics*, vol. 0, no. 0, pp. 1–22, 2019.

[20] V. Cossalter, M. Lio, R. Lot, and L. Fabbri, "A general method for the evaluation of vehicle manoeuvrability with special emphasis on motorcycles," *Vehicle System Dynamics - VEH SYST DYN*, vol. 31, pp. 113–135, 02 1999.

[21] R. Lot and F. Biral, "A curvilinear abscissa approach for the lap time optimization of racing vehicles," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 7559 – 7565, 2014, 19th IFAC World Congress.

[22] E. Bertolazzi, F. Biral, and M. Da Lio, "real-time motion planning for multibody systems," *Multibody System Dynamics*, vol. 17, no. 2, pp. 119–139, 2007.

[23] F. Biral, E. Bertolazzi, and P. Bosetti, "Notes on numerical methods for solving optimal control problems," *IEEJ Journal of Industry Applications*, vol. 5, no. 2, pp. 154–166, 2016.

[24] 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)*, June 2017, pp. 141–147.