Towards the Design of Robotic Drivers for Full-Scale Self-Driving Racing Cars

Danilo Caporale*,†, Alessandro Settimi†, Federico Massa†, Francesco Amerotti‡, Andrea Corti‡, Adriano Fagiolini††, Massimo Guiggiani‡, Antonio Bicchi†,‡,‡‡ and Lucia Pallottino†,‡

***Abstract*— Autonomous vehicles are undergoing a rapid development thanks to advances in perception, planning and control methods and technologies achieved in the last two decades. Moreover, the lowering costs of sensors and computing platforms are attracting industrial entities, empowering the integration and development of innovative solutions for civilian use. Still, the development of autonomous racing cars has been confined mainly to laboratory studies and small to middle scale vehicles. This paper tackles the development of a planning and control framework for an electric full scale autonomous racing car, which is an absolute novelty in the literature, upon which we report our preliminary experiments and perspectives on future work. Our system leverages real time Nonlinear Model Predictive Control to track a pre-planned racing line. We describe the whole control system architecture including the mapping and localization methods employed.**

Fig. 1. Roborace Robocar: a platform for the development of robotic drivers.

I. INTRODUCTION

Challenges are one of the main drives of robotics development, that call for integration of recent research results in real world applications. In the particular case of self-driving cars, the DARPA Grand Challenge [1] and Urban Challenge [2] have pushed the robotics community to develop autonomous cars meant to drive in unstructured or urban environments.

Following the encouraging results of these challenges, car producers have started developing self-driving technologies, which are foreseen to become a disruptive reality in the next decade [3] and possibly the first widespread Artificial Intelligence (AI) applications in robotics.

From deploying single vehicles to the consumer market to fleets of shared vehicles to be used as a service, there is a strong interest in the development of level 4 and 5 autonomous behaviors [4] for these cars. Of course, to achieve that, human safety is one of the main concerns. In fact, one of the expected results of the mass adoption of these systems is a sensible reduction of the rate of accidents, especially those due to the errors of human drivers. Despite this being a reasonable outcome, at the moment of writing this paper it is not yet clear where we stand with respect to this goal, and extensive testing would be necessary to confirm that this goal is within reach [5].

In this work, we focus on the particular scenario of a racing track environment, which allows driverless vehicles testing in extreme conditions (high accelerations that trigger the nonlinear behavior of vehicles) without jeopardizing human safety. The interest in this application is justified by the recent creation of racing challenges for full scale (Roborace[1]), middle scale (Formula SAE[2], see [6], [7]), and small scale (F1Tenth[3]) vehicles. Autonomous racing capabilities for non-electric vehicles have been demonstrated e.g. in [8], but to the best of the author's knowledge this is the first work demonstrating self-driving capabilities full scale electric racing cars.

We report preliminary experiments conducted on Roborace's DevBot, the development version of Robocar (see Fig. 1). The car can be driven either by a human pilot or by an autonomous system. The control architecture of the vehicle is hierarchical and it includes, at low-level, fast response control-loops implemented by the manufacturer, ensuring closed loop control of the actuators and raw input processing from the on-board sensors, while exposing at higher level an open platform for the implementation of user-defined control strategies and perception schemes. This key feature first enables rapid development and testing of full-stack autonomous driving control solutions, in which the designer can focus on driverless technology. It also seamlessly allow for the implementation of learning-based approaches in which the human or autonomous driver can execute arbitrarily complex maneuvers [9]. Moreover, compared to small scale vehicles, the hardware and software developed for this application are readily amenable to technology transfer to urban vehicles, for high speed or emergency maneuvering.

With respect to our previous work [10], where a mapping and localization system was introduced together with several

* Corresponding author: d.caporale@centropiaggio.unipi.it
† Centro di Ricerca E. Piaggio - Università di Pisa - Pisa, Italy
‡ Università di Pisa - Pisa, Italy
†† Università degli Studi di Palermo - Palermo, Italy
‡‡ Istituto Italiano di Tecnologia - Genova, Italy

[1] https://roborace.com/
[2] https://www.fsaeonline.com/
[3] http://f1tenth.org/

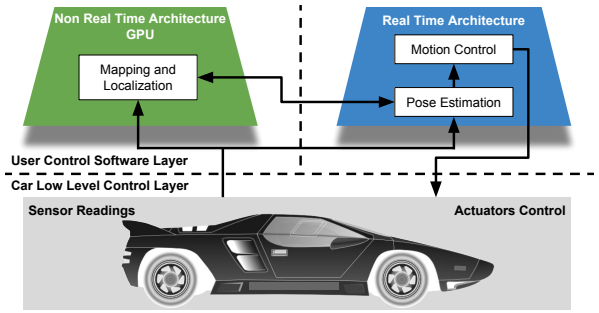


Fig. 2. Hierarchical control system architecture. Cars sensors and actuators are available as measurements and control actions for the user. Mapping and localization are performed on a non real time machine, while pose estimation and motion control run on a real time machine.

possibilities for controlling the car, in this paper we present a full stack architecture for motion planning, mapping, localization, and control, focusing on the development and experimental validation of the driver.

AI methods such as reinforcement learning [11] or end-to-end deep learning [12] are promising in deriving data based controls for this kind of applications, but pose significant challenges in predicting and understanding how the AI will react in emergency or unplanned situations, that is the ability of the expert system to generalize from learnt scenarios.

Other approaches can be found in literature which already demonstrated their effectiveness in motion planning and control for self-driving of urban cars. A recent survey can be found in [13]. With respect to the methods therein reported, the focus for racing vehicles is on racing line optimization over the entire track, obstacle avoidance through real time re-planning, and control of the nonlinear dynamics of the system. Recent results are reported in [14], where the problem of real time motion planning considering obstacles is considered and experimentally validated on small scale vehicles. Sampling-based methods [15] can be used both for motion planning with differential constraints for racing line generation or feedback planning for online control in presence of moving obstacles [16].

In this work we do not discuss the problem of overtaking competing vehicles, while we focus on racing line generation over the entire track and vehicle control. State-of-the-art methods in the literature report on two main approaches for racing line generation, where the geometrical path and the reference speed profile are computed separately or with a unique optimization problem. The first method is often preferred due to the fact that once a path has been defined, usually through nonconvex optimization, obtaining a speed profile becomes a convex problem, see [17], [18]. The second method requires nonlinear programming approaches and has been studied in [19], [20]. A recent comparison between nonlinear programming and optimal control methods for minimum lap-time racing line computation is given in [21], where the authors state that the different methods lead to similar results.

In the following we describe the overall architecture of our autonomous driving system, with particular focus on the motion planning (i.e., racing line optimization) and control. In particular, we based our motion control algorithm on a model based approach, and tested it on a commercial hard real time embedded platform for automotive applications.

II. RACING ENVIRONMENT

Race tracks are designed with sharp turns, long straights to reach high speed, and escape ways to recover from erroneous maneuvers, with the goal of pushing the performance of the vehicles to their limits. In this context one finds the ideal conditions for experimental validation of advanced driverless algorithms within a structured environment affected by uncertanties, such as the presence of adversarial vehicles that behave as non-cooperative entities, see [22], and an uncertain knowledge on the vehicle model.

With reference to typical requirements that one might consider for an autonomous race, the following are a set of reasonable constraints to consider for the single vehicle control:

- $V_{\max}$: maximum allowed speed for the vehicle;
- $g_{\text{lat,max}}$, $g_{\text{long,max}}$: maximum lateral and longitudinal normalized accelerations allowed by the road/grip conditions and the low level traction control;
- $L(s)$, $R(s)$: let s be the arc length of the track centerline, these are respectively the left and right track margins along the track which are allowed for a safe operation of the vehicle.

Other constraints, not considered here, arise when dealing with simultaneous presence of competing vehicles.

III. CONTROL SYSTEM ARCHITECTURE

The control system architecture developed is depicted in Fig. 2. A hierarchical control architecture is available on the car. Low-level controllers perform: i) torque vectoring control [23] for four independent electric motors; ii) tracking of the steering reference. At a higher level, the vehicle user can specify: i) a reference for the traction force F_T, that can be either positive (during traction phases) or negative (during braking phases); ii) a reference for the steering δ. High level force and steering references are computed in hard real time on a SpeedGoat board running at 250 Hz. This hierarchical approach is typical of autonomous vehicles and allows the user to control the vehicle with the same kind of actuation a human would do. The perception pipeline we adopted in this work comprises an IMU, differential GPS (both available with a sampling rate of 250 Hz) and multiple LIDAR sensors placed around the vehicle to cover the side-front-side area (that we fused to make them available as a single point cloud signal at a sampling rate of 25 Hz). Other sensors available on the Robocar were not necessary for these preliminary tests. The perception software runs in a non real time fashion on a NVIDIA DRIVE PX2 board.

The key elements of our system are: the motion planning, used to precompute a racing line; the perception pipeline, which is essential to perform a hybrid LIDAR and model based localization and mapping; and a model predictive controller used to compute the steering and force commands

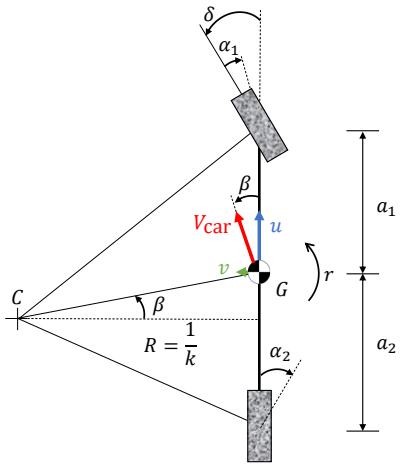


Fig. 3. Kinematic quantities of a cornering vehicle. Point C is the velocity center, perpendicular to the vehicle slip angle β.

required to track the desired racing line. In the following sections we describe more in detail the development and implementation of these components.

IV. SYSTEM MODELING AND IDENTIFICATION

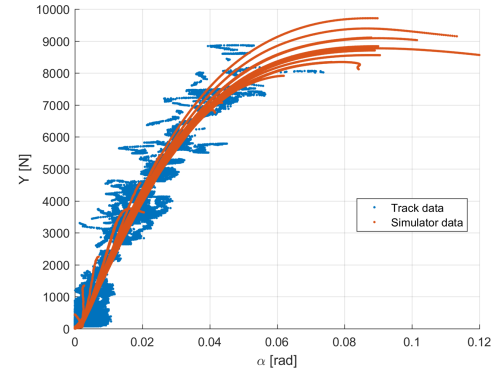
When choosing a reference model for the control of a racing vehicle, a trade-off between model complexity and model representativity must be faced. In our case, we adopted three different models: a kinematic model for path optimization, a dynamic mass model for speed profile optimization, and the dynamic single-track model for real time nonlinear model predictive control. Apart from the kinematic model, models are known as grey-box, hence some parameters need to be identified either via experiments (when possible) or via simulations on an accurate dynamic simulator provided by the car manufacturer. For a detailed presentation of vehicle dynamics modeling refer to [24]. A key role in the nonlinear dynamics of the car is played by the wheel-road contact, for which the Pacejka model [25] has been used to obtain a complete parametrization of the pneumatic in the region of interest.

With reference to the model of a cornering vehicle depicted in Fig. 3, let u and v be the longitudinal and lateral speeds of the vehicle, r the vehicle yaw rate, δ the steering angle and a_i, $i = 1, 2$, the front and rear axle distances from the center of mass G of the vehicle. Moreover, let m be the vehicle mass and Y_i, $i = 1, 2$, the ground contact forces at the front and rear axles.

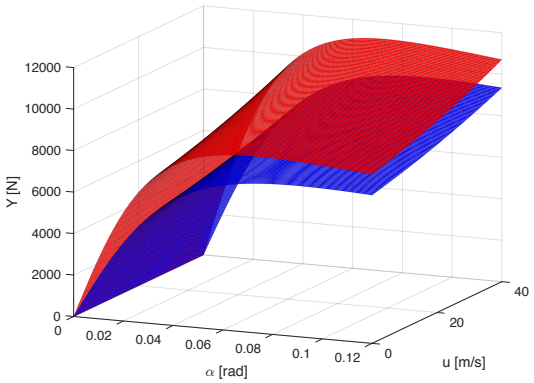
Under the assumption of *quasi-steady state conditions* (slowly varying longitudinal speed u and lateral acceleration a_y) the dynamics of the vehicle can be written as in (1).

$$\begin{aligned} ma_y &= Y_1 + Y_2 \\ 0 &= a_1 Y_1 + a_2 Y_2 \end{aligned} \tag{1}$$

For a front steering vehicle, the congruence equations give the apparent slip angles of the front and rear axles as $\alpha_1 = \delta - \arctan\left(\frac{v+ra_1}{u}\right)$ and $\alpha_2 = -\arctan\left(\frac{v-ra_2}{u}\right)$, respectively. The tyre force model for a single axle (front or rear) ground force is then given by (2)



(a)



(b)

Fig. 4. Axle characteristics obtained with the Pacejka Tyre Formula [26]. (a) A detail of the model fitting data from a real track compared with the data from the car simulator. (b) front (blue) and rear (red) axle characteristics as a function of longitudinal speed and apparent slip angle.

$$Y(\alpha,u) = A \sin\left(B \arctan\left(C\alpha - D\left(C\alpha - \arctan\left(C\alpha\right)\right)\right)\right)\lambda(u), \tag{2}$$

where $\lambda(u)$ is a second degree polynomial used to model aerodynamic effects as a function of the speed, A, B, C, D are coefficients commonly named, respectively, peak value, shape factor, stiffness factor and curvature factor that were computed with a least squares fitting algorithm. A single pole dynamics for the tyre transient has been considered. The resulting curves are shown in Fig. 4.

V. RACING LINE OPTIMIZATION

To provide a racing line to the vehicle using its identified model, an offline trajectory optimization algorithm has been developed in two steps. First, a minimum curvature path has been obtained based on the geometry of the track and the car. Then, a speed profile has been derived on this path using the simplified dynamics of the point mass model.

A. Optimal Curvature Path

Let s be the arc length of the track, $x(s)$ and $y(s)$ the coordinates on the 2D map of the points to be optimized. Let $x' = \frac{dx}{ds}$ and $y' = \frac{dy}{ds}$ be the spatial derivatives at point s. Then, the curvature of the path at point s is given

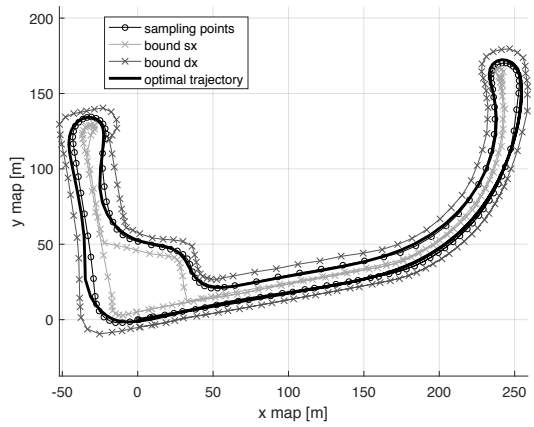


Fig. 5. Racing line in map frame p^*_{map}, with approaching path from zero speed at $(x, y) = (0, 0)$.

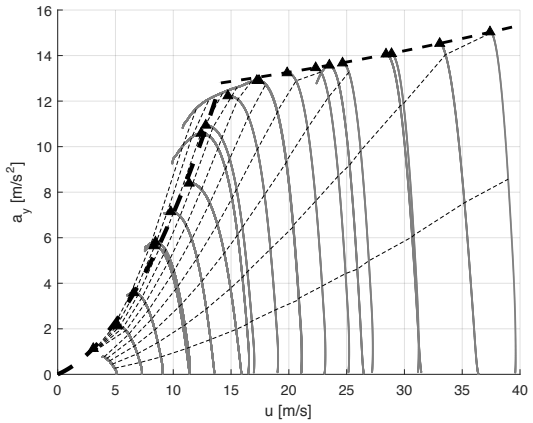


Fig. 6. Map of Achievable Performance [24] for the lateral acceleration. Continuous grey lines represent data from slow ramp steer maneuvers performed on the vehicle dynamic simulator. Thin dashed lines represent points at the same value of steering angle, increasing from the bottom to the top. The thick dashed line fits the peaks of the steer maneuver data and gives a conservative limit on the lateral acceleration as a function of the longitudinal speed.

by $k(s) = (x'y'' - y'x'')/\left(x'^2 + y'^2\right)^{3/2}$. The path is discretized in N_s segments, where the discretization step along the arc length s is chosen adaptively as a decreasing function of the mean line curvature, leading to a finer sampling where the track curvature is higher. The path of the racing line has been computed by solving the optimization problem in (3) over the fixed discretization s_i of the mean line of the track.

$$\begin{aligned} \min_{x,y} \quad & \sum_{i=1}^{N_s} k\big(x(s), y(s)\big) \\ \text{s.t.} \quad & \big(x(s), y(s)\big) \in \mathcal{C}_{\text{free}}\big(L(s), R(s)\big) \end{aligned} \tag{3}$$

$L(s)$ and $R(s)$ represent, respectively, the borders of the track and we refer to them for simplicity as *left* and *right* borders, coherently with the forward direction of the car. Since the presence of other vehicles and obstacles on the track is disregarded, the collision free configuration space $\mathcal{C}_{\text{free}}$ is solely defined by these margins. Thus, the constraint in (3) guarantees that each point of the racing line lies within the track borders. The result of this optimization can be interpolated with splines or clothoids to produce a smooth reference trajectory of optimal heading ψ^*_{map}, position $p^*_{\text{map}} = \begin{bmatrix} x^*_{\text{map}} & y^*_{\text{map}} \end{bmatrix}^T$, and curvature k^*, see Fig. 5.

B. Speed Profile Optimization

Once an optimal path is available, a reference speed profile is obtained considering the point mass model of the vehicle. Speed can be iteratively computed as $V(i+1) = \sqrt{V(i)^2 + 2a_T(i)\Delta s\sqrt{x'^2 + y'^2}}$, with a_T and Δs the acceleration and step along the arc length, respectively. $\Delta t\big(V(i)\big)$ is the interval during which the car is driving along the i-th segment of the racing line, and can be written as

$$\Delta t\big(V(i)\big) = \begin{cases} \frac{-V(i) + \sqrt{V(i)^2 + 2a_T(i)\Delta s\sqrt{x'^2+y'^2}}}{a_T(i)} & \text{if } a_T(i) \neq 0 \\ \frac{\Delta s\sqrt{x'^2+y'^2}}{V(i)} & \text{if } a_T(i) = 0 \end{cases}$$

Maximum accelerations and torques are the constraints of the problem. In particular, for the lateral acceleration we refer to the point mass model, hence $a_N = \frac{V}{R^2}$ where $R = \frac{1}{k}$ is the instantaneous curvature radius of the trajectory. The safety limit for the maximum lateral acceleration was derived from the Map of Achievable Performance $a_N - u$ as a function of the speed [24], see Fig. 6. The resulting constraints are

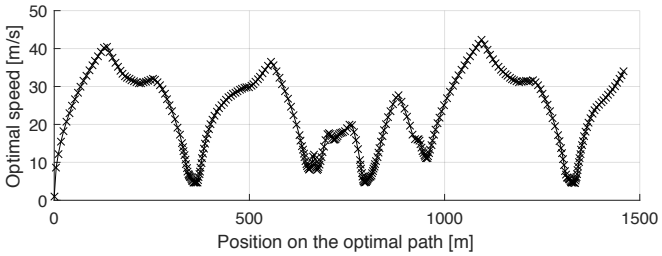


Fig. 7. Racing line optimal speed profile without constraints on the maximum speed.

$$\begin{aligned} a_T(i) < a_x^{\max} = \mu_x g - \tfrac{1}{2}\gamma C_x S V^2 \qquad & \text{max. long. acc.} \\ a_N(i) = a_y^{\max} \qquad & \text{max. lat. acc.} \\ \left(\tfrac{a_T(i)}{a_x^{\max}}\right)^2 + \left(\tfrac{a_N(i)}{a_y^{\max}}\right)^2 < 1 \qquad & \text{max. combined acc.} \\ a_x^{\max} = \tfrac{F_{\max}}{m} \end{aligned} \tag{4}$$

where: μ_x is the longitudinal friction coefficient that can be derived as the projection of the ground forces on the longitudinal direction of the car; g is the gravity acceleration; γ is the air density; C_x is the drag coefficient; S the frontal area of the car; a_T and a_N are the longitudinal (tangential) and lateral (normal) accelerations; $F_{\max}(u)$ is the maximum traction/braking force at ground level that the electric motors can exert, as a function of the vehicle speed. A nonlinear optimization problem can then be solved as in (5), where a first guess on the speed profile is provided as a function of the curvature of the track and the maximum speed.

$$\min_{V} \quad \sum_{i=1}^{N_s} \Delta t\big(V(i)\big) \qquad \text{s.t.constraints (4)} \tag{5}$$

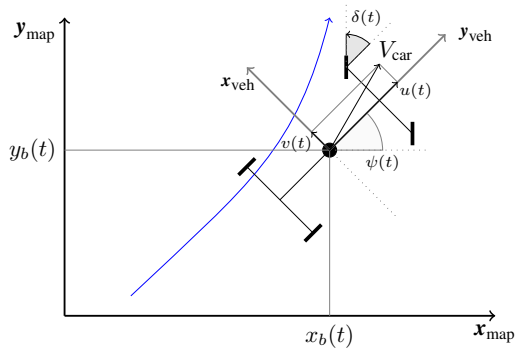


Fig. 8. Reference trajectory (in blue) and car coordinates in the vehicle frame for the single track model.

The resulting speed profile is given in Fig. 7. Finally, we can define the desired vehicle trajectory as $\boldsymbol{p}_{\text{ref}} = \begin{Bmatrix} \psi^*_{\text{map}} & p^*_{\text{map}} & k^* & V^* \end{Bmatrix}$. This reference trajectory of the vehicle constitutes a feed-forward input to the motion control system as detailed below.

VI. MOTION CONTROL

For the purpose of model based control we used the dynamic single track model of the car, see [24]. Front and rear axles slip angles α_i, $i = 1, 2$ are as in (1), and axle forces $Y_i(\alpha_i)$, $i = 1, 2$ as in (2). With reference to Fig. 3 and Fig. 8, let u and v be the longitudinal and lateral speed components of the vehicle in local frame, and V_{car} the magnitude of the speed vector. The dynamic equation of the single track model is then given in (6)

$$\begin{aligned} ma_y &= m(\dot{v} + ur) = Y_1(\alpha_1) + Y_2(\alpha_2) \\ J_z\dot{r} &= Y_1 a_1 - Y_2 a_2 \end{aligned} \tag{6}$$

where J_z is the moment of inertia of the vehicle. The kinematics of the vehicle can be obtained from (7)

$$\begin{aligned} \dot{x}_b &= u\cos\psi - v\sin\psi \\ \dot{y}_b &= u\sin\psi + v\cos\psi \\ \dot{\psi} &= r \\ \dot{u} &= a_x + vr \\ \dot{v} &= a_y - ur \\ \dot{r} &= \frac{Y_1 a_1 \cos\delta + X_1 a_1 \sin\delta - Y_2 a_2}{J_z} \end{aligned} \tag{7}$$

where ψ is the heading of the car, x_b and y_b its position in the map frame, see Fig. 8. The measured or estimated outputs used for control are $z = \begin{bmatrix} x_b & y_b & \psi & u & v & r \end{bmatrix}^T$, and the control signal is $\nu = \begin{bmatrix} \delta & F_t \end{bmatrix}^T$, with δ the steering command and F_t the total (signed) traction/braking force. In this way, the vehicle is controlled with the same interface that a human driver would use, with a different and possibly richer set of information coming from the vehicle sensors and the knowledge of the model parameters.

A. Real Time Model Predictive Control

We used Nonlinear Model Predictive Control (NMPC) to satisfy two key requirements for our control system: i) to include nonlinear tyre characteristics in the vehicle model; and ii) to predict the car behavior over a prediction horizon, exploiting its nonlinear model. Given a reference trajectory, let the tracking error at discrete time k be

$$e(k) = \begin{bmatrix} x^*(k) - x(k) \\ y^*(k) - y(k) \\ \psi^*(k) - \psi(k) \\ u^*(k) - u(k) \end{bmatrix} \tag{8}$$

The cost function considered is $J = \sum_{k=\tau_0}^{T_{\text{hor}}} e(k)^T Q e(k)$, where Q is a positive semidefinite weight matrix. The sampling time used for the execution of this controller on a real time target machine is $T_s = 0.004$ s, while the prediction horizon was chosen to be $T_{\text{hor}} = 0.5$ s.

Since the numerical solver employed finds local solutions of the NMPC problem, we provided a warm start solution at each sample time as a strategy to avoid failure in the computation of a control. The procedure employed is outlined in Algorithm 1, where by $\mathbf{F}(i)$ and $\boldsymbol{\delta}(j)$ we mean vectors of constant values over the whole prediction horizon. It is worth noting that, despite this algorithm is not computationally efficient since it explores all the possible solutions on the grid of constant control values, the prediction horizon considered is small enough to make the computation of $J_{i,j}(k)$ by forward integration of the model based error dynamics amenable to real time usage.

The Reference Horizon Generator (RHG) block in Fig. 9 selects in real time the portion of the racing line that belongs to the prediction horizon T_{hor}, starting with the current pose of the car, which is the problem of how to localize the car against the racing line. Letting τ be an index over the discretized reference vector $p_{\text{ref}}(\tau)$, we seek for the index τ^* on the reference path solving (9), where $\boldsymbol{p}_b(t) = \begin{bmatrix} x_b(t) & y_b(t) \end{bmatrix}^T$.

$$\tau^* = \arg\min_{\tau} \|\boldsymbol{p}_{\text{ref}}(\tau) - \boldsymbol{p}_b(t)\|_2 \tag{9}$$

By varying the prediction horizon T_{hor} and the error weight matrix Q it is possible to tune the behavior of the steering and force commands.

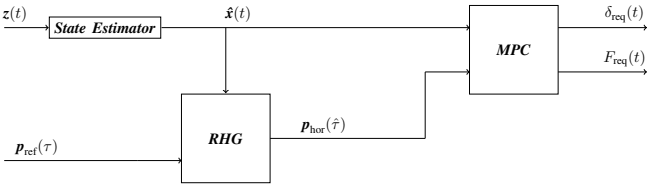


Fig. 9. The measured output of the system $z(t)$ is used by the State Estimator to obtain the estimated state $\hat{x}(t)$. The *RHG* $\boldsymbol{p}_{\text{ref}}(\tau)$ computes the desired trajectory on the prediction horizon $\boldsymbol{p}_{\text{hor}}(\hat{\tau})$, while the *MPC* block computes the controls for the car.

VII. MAPPING AND LOCALIZATION

In this work we used a combination of LIDARs raw data, IMU accelerations and GPS speed, in combination with an optimization based SLAM ROS package [27], to perform mapping.

Data: $u(k-1)$: control vector at previous sample time.
T_{hor} prediction horizon.
Result: $u_{\text{guess}}(k)$: warm start control vector at current sample time k for the MPC solver
Compute the cost $J(0,0)$ obtained with $u(k-1)$;
Compute a grid of $N \times M$ points in the minimum and maximum range of the control signals δ and F_T;
for $i \leftarrow 1$ **to** N **do**
  for $j \leftarrow 1$ **to** M **do**
    Apply $u(i,j) \leftarrow (\mathbf{F}(i), \boldsymbol{\delta}(j))$;
    Compute $J(i,j)$;
    if $J(i,j) < J_{\text{min}}$ **then**
      $J_{\text{min}} \leftarrow J(i,j)$;
      $u_{\text{guess}}(k) \leftarrow u(i,j)$;
    end
  end
end

Algorithm 1: Warm start algorithm for the MPC optimization problem.

To localize the car we developed an Extended Kalman Filter (EKF) based on the single track model of Equations (1), (6), (7), that receives as inputs the high rate measure updates from the vehicle sensors (IMU, wheel speed and optical speed sensor) as well as a low rate update from the LIDAR+IMU+GPS odometry. In this way it is easy to compute frequent updates of the vehicle state while correcting drift using LIDAR sensors input.

VIII. EXPERIMENTAL RESULTS

Some preliminary results of experiments conducted on the real vehicle are here reported. The vehicle used is the development version of the Robocar, namely the DevBot (see Fig. 1 in [10]). For these tests, a safety limit of $V_{\text{max}} = 50$ km/h was considered. Normalized longitudinal and lateral accelerations, are limited to $g_{\text{long,max}} = g_{\text{lat,max}} = 0.8$. Localization (offtrack) error with respect to the planned path is below 0.3m (see Fig. 12 in [10]). Despite being relatively small when compared with the size of the car (about 4 m long and 2 m wide), the use of different localization techniques or a better tuning of the localization algorithm can improve this result. A major issue we faced is that scan matching performed on PX2 overloads the ARM CPU, sometimes resulting in failures. For this reason we are working on a more efficient implementation.

Speed and acceleration tracking results are shown in Fig. 10. Both transient and steady state performance were satisfactory. Measured accelerations in vehicle frame are compared with the ones obtained from the racing line. One main observation is that for high values of the acceleration we see differences in the values achieved by the vehicle. This can be due to a conservative estimation of the acceleration limits in the handling map or neglected second order dynamics (suspension dynamics). Moreover, the single mass model used for planning the vehicle trajectory does not take into account the fact that the vehicle is not always aligned along the tangent to the path, hence resulting in an observed lower acceleration during some turns.

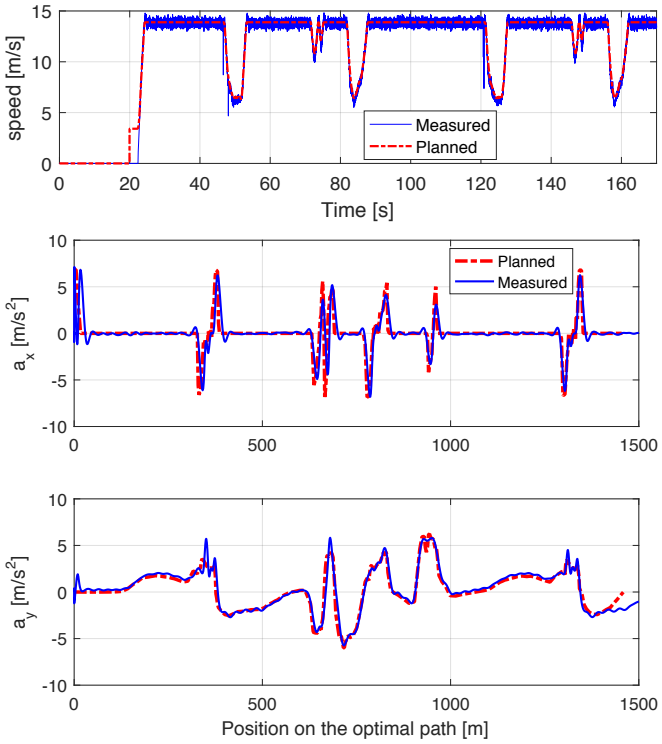


Fig. 10. From the top: speed tracking over two laps, detail of longitudinal and lateral acceleration tracking.

Finally, the trajectory planning algorithm was implemented with MATLAB fmincon function, hence convergence to the global optimal solution is not guaranteed. Despite this, for the purpose of this work a feasible trajectory was sufficient for operating the vehicle.

IX. CONCLUSIONS

In this work we reported some preliminary experiments conducted towards the development of a fully autonomous race driver for a full scale electric autonomous vehicle, namely the Roborace Robocar. The methods and the results reported are general enough to be applied to vehicles of smaller scale or other autonomous cars. Here we focused on the development of a modular architecture for the mapping, localization, planning and control of the vehicle. Future work will be devoted to globally optimal path planning and online re-planning in the multi-vehicle scenario.

ACKNOWLEDGMENT

The authors would like to thank all the members of Roboteam Italia[4] for the hard work, dedication and support provided to this project. A special thanks goes to Roborace team for the valuable support received during the preparation of this work.

[4]http://www.roboteamitalia.it

REFERENCES

[1] M. Buehler, K. Iagnemma, and S. Singh, *The 2005 DARPA grand challenge: the great robot race*. Springer, 2007, vol. 36.

[2] ——, *The DARPA urban challenge: autonomous vehicles in city traffic*. Springer, 2009, vol. 56.

[3] P. Gao, H.-W. Kaas, D. Mohr, and D. Wee, "Automotive revolution–perspective towards 2030 how the convergence of disruptive technology-driven trends could transform the auto industry," *Advanced Industries, McKinsey & Company*, 2016.

[4] *Taxonomy and Definitions for Terms Related to On-Road Motor Vehicle Automated Driving Systems*, January 2014. [Online]. Available: https://doi.org/10.4271/J3016_201401

[5] D. J. Hicks, "The safety of autonomous vehicles: Lessons from philosophy of science," *IEEE Technology and Society Magazine*, vol. 37, no. 1, pp. 62–69, 2018.

[6] M. d. l. I. Valls, H. F. C. Hendrikx, V. Reijgwart, F. V. Meier, I. Sa, R. Dubé, A. R. Gawel, M. Bürki, and R. Siegwart, "Design of an autonomous racecar: Perception, state estimation and system integration," *arXiv preprint arXiv:1804.03252*, 2018.

[7] T. Drage, J. Kalinowski, and T. Braunl, "Integration of drive-by-wire with navigation control for a driverless electric race car," *IEEE Intelligent Transportation Systems Magazine*, vol. 6, no. 4, pp. 23–33, 2014.

[8] V. A. Laurense, J. Y. Goh, and J. C. Gerdes, "Path-tracking for autonomous vehicles at the limit of friction," in *American Control Conference (ACC), 2017*. IEEE, 2017, pp. 5586–5591.

[9] T. Lin, E. Tseng, and F. Borrelli, "Modeling driver behavior during complex maneuvers," in *American Control Conference (ACC), 2013*. IEEE, 2013, pp. 6448–6453.

[10] D. Caporale, A. Fagiolini, L. Pallottino, A. Settimi, A. Biondo, F. Amerotti, F. Massa, S. De Caro, A. Corti, and L. Venturini, "A planning and control system for self-driving racing vehicles," in *2018 IEEE 4th International Forum on Research and Technology for Society and Industry (RTSI)*. IEEE, 2018, pp. 1–6.

[11] A. E. Sallab, M. Abdou, E. Perot, and S. Yogamani, "Deep reinforcement learning framework for autonomous driving," *Electronic Imaging*, vol. 2017, no. 19, pp. 70–76, 2017.

[12] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to End Learning for Self-Driving Cars," *arXiv:1604.07316 [cs]*, Apr. 2016, arXiv: 1604.07316. [Online]. Available: http://arxiv.org/abs/1604.07316

[13] 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, 2016.

[14] A. Liniger, "Path planning and control for autonomous racing," Ph.D. dissertation, ETH Zurich, 2018.

[15] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[16] Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli, and J. P. How, "Real-time motion planning with applications to autonomous urban driving," *IEEE Transactions on Control Systems Technology*, vol. 17, no. 5, pp. 1105–1118, 2009.

[17] T. Lipp and S. Boyd, "Minimum-time speed optimisation over a fixed path," *International Journal of Control*, vol. 87, no. 6, pp. 1297–1311, 2014.

[18] L. Consolini, M. Laurini, M. Locatelli, and A. Minari, "A solution of the minimum-time velocity planning problem based on lattice theory," *arXiv preprint arXiv:1809.01959*, 2018.

[19] G. Perantoni and D. J. Limebeer, "Optimal control for a formula one car with variable parameters," *Vehicle System Dynamics*, vol. 52, no. 5, pp. 653–678, 2014.

[20] D. Limebeer and M. Massaro, *Dynamics and optimal control of road vehicles*. Oxford University Press, 2018.

[21] N. Dal Bianco, E. Bertolazzi, F. Biral, and M. Massaro, "Comparison of direct and indirect methods for minimum lap time optimal control problems," *Vehicle System Dynamics*, pp. 1–32, 2018.

[22] A. Liniger and J. Lygeros, "A non-cooperative game approach to autonomous racing," *arXiv preprint arXiv:1712.03913*, 2017.

[23] C. Chatzikomis, A. Sorniotti, P. Gruber, M. Bastin, R. M. Shah, and Y. Orlov, "Torque-vectoring control for an autonomous and driverless electric racing vehicle with multiple motors," *SAE International Journal of Vehicle Dynamics, Stability, and NVH*, vol. 1, no. 2017-01-1597, pp. 338–351, 2017.

[24] M. Guiggiani, *The science of vehicle dynamics*. New York, NY: Springer International Publishing, 2018.

[25] H. Pacejka, *Tire and vehicle dynamics*. Elsevier, 2005.

[26] H. B. Pacejka and E. Bakker, "The magic formula tyre model," *Vehicle system dynamics*, vol. 21, no. S1, pp. 1–18, 1992.

[27] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2D LIDAR SLAM," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1271–1278.