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

Danilo Caporale†, Alessandro Settimi‡, Federico Massa‡, Francesco Amerotti‡, Andrea Corti‡, Adriano Fagliolini‡, 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.

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), middle scale (Formula SAE) and small scale (FITenth) 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
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 uncertainties, 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:

- \( V_{\text{max}} \): 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 \( \delta \). 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.
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, \( \delta \) 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).

\[
ma_y = Y_1 + Y_2 \quad 0 = a_1Y_1 + a_2Y_2
\]  

(1)

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

\[
Y(\alpha, u) = A \sin \left( B \arctan \left( C\alpha - D \left( C\alpha - \arctan (C\alpha) \right) \right) \right) \lambda(u),
\]

(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
by \( k(s) = \frac{\left(x'y'' - y'x''\right)}{\left(x'^2 + y'^2\right)^{3/2}} \). 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.

\[
\begin{align*}
\min_{x,y} & \quad \sum_{i=1}^{N_s} k(x(s_i), y(s_i)) \\
\text{s.t.} & \quad (x(s_i), y(s_i)) \in C_{\text{free}}(L(s), R(s))
\end{align*}
\]

\( L(s) \) 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 \( C_{\text{free}} \) 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 \( \psi_{\text{map}}^* \), position \( p_{\text{map}}^* = [x_{\text{map}}^*, y_{\text{map}}^*]^T \), 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 \( \Delta s \) the acceleration and step along the arc length, respectively. \( \Delta t(V(i)) \) is the interval during which the car is driving along the \( i \)-th segment of the racing line, and can be written as

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

Maximum accelerations and torques are the constraints of the problem. In particular, for the lateral acceleration we refer to the point mass model, hence \( a_N = \frac{V}{\rho} \), where \( \rho = \frac{1}{\kappa} \) 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

\[
\begin{align*}
& a_T(i) < a_{T\text{max}} = \mu_x g - \frac{1}{2} \gamma C_x SV^2 \\
& a_N(i) = a_{N\text{max}} \\
& \left(\frac{a_T(i)}{a_{T\text{max}}}\right)^2 + \left(\frac{a_N(i)}{a_{N\text{max}}}\right)^2 < 1
\end{align*}
\]

where: \( \mu_x \) 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; \( \gamma \) 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; \( F_{\text{max}}(u) \) 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.

\[
\begin{align*}
\min_{V} & \quad \sum_{i=1}^{N_s} \Delta t(V(i)) \\
\text{s.t. constraints} & \quad (4)
\end{align*}
\]
Finally, we can define the desired vehicle trajectory as 

\[ P_{\text{ref}} = \{ \psi^*_{\text{map}}, p^*_{\text{map}}, k^* \}. \]

This reference trajectory of the vehicle constitutes a feed-forward input to the motion control system as detailed below.

\[ J. \text{ MOTION CONTROL} \]

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 \( \alpha_i \), \( i = 1, 2 \) 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 \( V_{\text{car}} \) the magnitude of the speed vector. The dynamic equation of the single track model is then given in (6)

\[ \begin{align*}
    m a_y &= m(\dot{\psi} + ur) = Y_1(\alpha_1) + Y_2(\alpha_2) \\
    J_z \dot{\psi} &= Y_1 a_1 - Y_2 a_2
\end{align*} \quad (6) \]

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

\[ \begin{align*}
    \dot{x}_b &= u \cos \psi - v \sin \psi \\
    \dot{y}_b &= u \sin \psi + v \cos \psi \\
    \dot{\psi} &= r \\
    \dot{u} &= a_x + vr \\
    \dot{v} &= a_y - ur \\
    \dot{r} &= \frac{Y_1 a_1 \cos \delta + X_1 a_1 \sin \delta - Y_2 a_2}{J_z}
\end{align*} \quad (7) \]

where \( \psi \) 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 \( z = [x_b \; y_b \; \psi \; u \; v \; r]^T \), and the control signal is \( \nu = [\delta \; F_i]^T \), with \( \delta \) the steering command and \( F_i \) 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

\[ e(k) = \begin{bmatrix}
    x^*(k) - x(k) \\
    y^*(k) - y(k) \\
    \psi^*(k) - \psi(k) \\
    u^*(k) - u(k)
\end{bmatrix} \quad (8) \]

The cost function considered is \( J = \sum_{k=0}^{T_{\text{hor}}} e(k)^T Q e(k) \), 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 \( F(i) \) and \( \delta(j) \) 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 \( J_{ij}(k) \) 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 \( T_{\text{hor}} \), starting with the current pose of the car, which is the problem of how to localize the car against the racing line. Letting \( \tau \) be an index over the discretized reference vector \( P_{\text{ref}}(\tau) \), we seek for the index \( \tau^* \) on the reference path solving (9), where \( P_b(t) = [x_b(t) \; y_b(t)]^T \).

\[ \tau^* = \arg \min_{\tau} \| P_{\text{ref}}(\tau) - P_b(t) \|_2 \quad (9) \]

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

\[ \text{VII. MAPPING AND LOCALIZATION} \]

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.

Fig. 8. Reference trajectory (in blue) and car coordinates in the vehicle frame for the single track model.
Data: \( u(k - 1) \): control vector at previous sample time. 
\( T_{\text{hor}} \): 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 \( \delta \) and \( F_r \);

for \( i \leftarrow 1 \) to \( N \)
for \( j \leftarrow 1 \) to \( M \)
    Apply \( u(i, j) \leftarrow (F(i), \delta(j)) \);
    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

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 (4), (5), (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.

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