A Planning and Control System for Self-Driving Racing Vehicles

Danilo Caporale, Adriano Fagiolini, Lucia Pallottino, Alessandro Settimi,
Andrea Biondo, Francesco Amerotti, Federico Massa, Stefano De Caro, Andrea Corti, and Luca Venturini

***Abstract*—Autonomous robots will soon enter our everyday life as self-driving cars. These vehicles are designed to behave according to certain sets of cooperative rules, such as traffic ones, and to respond to events that might be unpredictable in their occurrence but predictable in their nature, such as a pedestrian suddenly crossing a street, or another car losing control. As civilian autonomous cars will cross the road, racing autonomous cars are under development, which will require superior Artificial Intelligence Drivers to perform in structured but uncertain conditions. We describe some preliminary results obtained during the development of a planning and control system as key elements of an Artificial Intelligence driver for the competition scenario.**

***Index Terms*—Autonomous robots, self-driving vehicles, racing, robotics challenge.**

I. Introduction

Challenges are one of the main drives of robotics development, that call for integration of recent research results in real world applications. A notable example is the field of self-driving vehicles, where the DARPA Grand Challenge [1] and Urban Challenge [2] have pushed the robotics community to build autonomous cars for unstructured or urban scenarios. Following these challenges, interest in the development of hardware and software technologies for self-driving cars has increased, and are likely to be the first widely adopted Artificial Intelligence (AI) robots to appear in everyday life, but to date there is still uncertainty on how to guarantee operational safety for these robots. There is still debate on which should be the way to develop safe autonomous driving software as widely adopted deep learning algorithms are unpredictable in their nature and each AI driver might react differently in emergency situations [3]. 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 safe level 4 and 5 autonomous behavior [4] for these vehicles. Technical realization of such complex systems poses some questions that might find an answer in dedicated scenarios. In this work we consider a the racing track environment, as new competitions exist or are being developed where full scale (Roborace [1]) or small scale (F1tenth [2]) self-driving cars compete in a controlled, structured but uncertain scenario, with little or no risk for human drivers or pedestrians.

This work reports preliminary results on the development of an AI driver, for full scale electric autonomous vehicles

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

Fig. 1. Roborace DevBot - an electric self-driving racing car that can be driven by a human or fully autonomously

within the Roborace championship, some of which have been obtained using a Roborace DevBot (Figure 1). An optimal trajectory planning strategy is described for maximizing the vehicle's average travelling speed along the track. The resulting optimal paths can be improved when taking into account car conditions that change during the race, due e.g. to tyre consumption. A more complete description of the vehicle dynamics may be adopted considering different optimization schemes, as the one presented in [5] for motorcycles.

II. Racing track characteristics

A discussion is due to highlight the differences and possible benefits of developing an AI driver for a racing track scenario.

Obviously, during car race tests or events there are no people inside or outside the vehicle, hence the focus can be reduced to the third law of Asimov (self preservation of the robot). The road conditions are ideal, e.g. regular road surface, good visibility. The track can be delimited by walls, grass, cobble, all these characteristics being detectable with LIDAR or camera sensors. The track is not completely closed but may present safety or alternative path ways that have to be taken into account in the mapping and trajectory planning phase, see Figure 2 for an example[3].

III. Racing car functionalities

The input-output mapping for a self-driving vehicle control system has the typical structure of Figure 3. An autonomous car is equipped with both proprioceptive sensors, such as GPS, IMU, wheel encoders, optical speed sensor; and exteroceptive sensors, such as LIDAR, computer vision cameras, radars. The outputs are given as a steering command $\delta \in [-1, 1]$ rad and a throttle command $F_t \in [-1, 1]$ N. In this way, the car dynamics and low level controls are exposed to the controller

[3]http://www.autodromoimola.it/

Fig. 2. Example of alternative pathways that can be found in a racing track. Courtesy of Autodromo Internazionale di Imola Enzo e Dino Ferrari.

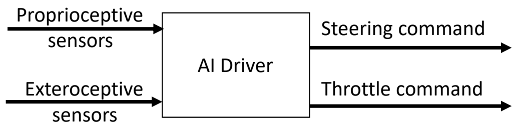


Fig. 3. AI driver input-output scheme

as a black box, replicating the way a human driver interacts with the vehicle. It is important to point out that very limited information about the vehicle dynamics can be used to drive the car at very high accelerations in a racing scenario, as will be explained later.

The AI driver has then to be able to perform a certain set of operations, such as mapping, localization, racing line optimization, path following, lap time minimization, obstacle avoidance, and some of these are covered in the next sections.

A. Mapping

Knowledge of the environment is fundamental to plan an optimal, collision free trajectory. The pose of several track features, such as fixed obstacles (walls, barriers, cones...) which can be measured by LIDARs or camera sensors, can be retrieved both offline or online, with the car manually driven or in autonomous exploration mode. The simplest approach that can be followed is to gather data while the robot is manually driven offline, then post-process the data to generate an occupancy grid map. Typical mapping algorithms require odometry (from wheel encoders or GPS receiver), vision (LIDARs and cameras), IMUs. Several software solutions can be used for this task, such as Google Cartographer [6], RGBDSLAMv2 [7] just to cite a few. These software solutions were designed to work both indoor and outdoor, hence they are robust w.r.t. GPS signal losses.

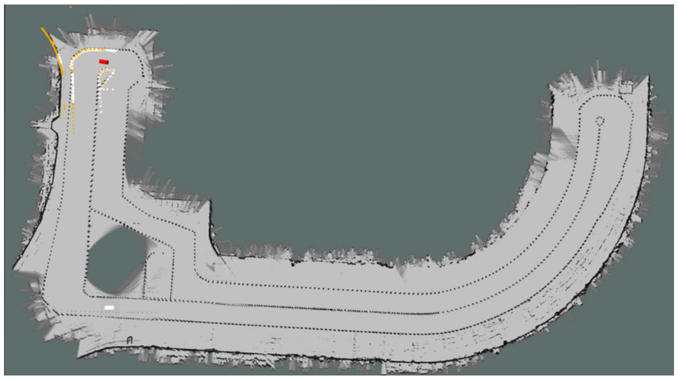


Fig. 4. Map estimation of a test track. Red: estimated pose of the car; Orange: current LIDAR scans; White: Points from the LIDAR scan that matched a submap.

B. Localization

Once a known map of the environment is available, the next step is to localize the car in such map and/or with respect to a planned trajectory. Localization, intended as an estimation of the car's pose $\xi = (x, y, \phi)$, with $\phi =$ yaw angle, can be split in two phases: first, a scan matching or feature matching algorithm uses camera LIDAR data to provide a first rough estimate of the vehicle position within the map, Fig.4. The robot pose is updated roughly by using odometry and IMU data, and can be further improved by using models of vehicle kinematics or dynamics, e.g. by means of an extended Kalman filter (EKF), to provide an accurate pose estimation of the vehicle. Other useful inputs for the EKF are speed and accelerations in vehicle frame, as measured by proprioceptive sensors.

C. Trajectory planning

The goal of the trajectory planning phase is to determine the way by which the vehicle should travel along a given track, in order to minimize the required lap time. A first step is to perform path length minimization [8] or traveling speed maximization [9]. The general consideration to bear in mind is that the maximum speed that a vehicle can sustain without slipping is strictly related to the instantaneous curvature σ of the path it is following. This is expressed by a relation of the form

$$m\, V_{max}^2 = \frac{F_{c,max}}{\sigma}\,, \tag{1}$$

being m the mass of the vehicle and $F_{c,max}$ the maximum lateral force tires can sustain, which intuitively shows that a smaller path curvature allows a higher maximum speed. Therefore, the above mentioned two strategies are in conflict since, to minimize the path length, it is necessary to have high curvature. An optimal trade-off between these two strategies [10] can indeed be obtained by considering a trajectory F satisfying the convex combination

$$F^2 = (1 - \epsilon) \cdot \Gamma^2 + \epsilon \cdot \Lambda^2\,, \tag{2}$$

where Λ and Γ are the solutions of the length and the path curvature minimization, respectively, while ϵ is a vector of

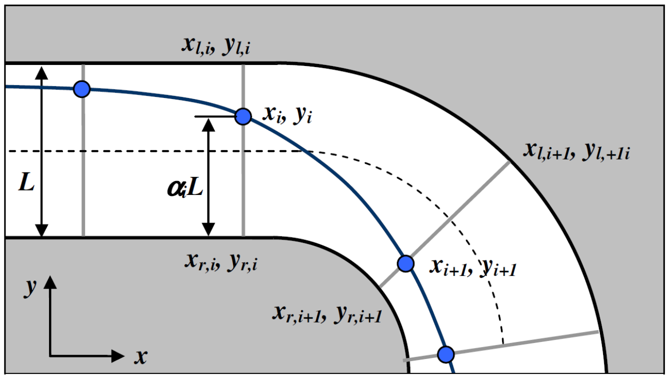


Fig. 5. Admissible positions space

weights. Since the requested solution needs to be constrained inside the track, the trajectory planner has to solve a constrained minimization. Given the left and right side track bound based reference frames

$$F_l = \left\{ O_l = \begin{pmatrix} x_l \\ y_l \end{pmatrix}; \quad \hat{i}_l = \begin{pmatrix} c_\theta \\ s_\theta \end{pmatrix}; \quad \hat{j}_l = \begin{pmatrix} -s_\theta \\ c_\theta \end{pmatrix} \right\},$$

$$F_r = \left\{ O_r = \begin{pmatrix} x_r \\ y_r \end{pmatrix}; \quad \hat{i}_r = \begin{pmatrix} c_\theta \\ s_\theta \end{pmatrix}; \quad \hat{j}_r = \begin{pmatrix} -s_\theta \\ c_\theta \end{pmatrix} \right\},$$

the space of admissible positions is

$$P_G = \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x_r \\ y_r \end{pmatrix} + \alpha \begin{pmatrix} x_l - x_r \\ y_l - y_r \end{pmatrix}, \tag{3}$$

being $\alpha \in [0, 1]$ a scalar parameter. Therefore, a sub-optimal solution to the path length minimization problem can be calculated by approximating the total trajectory length as the sum of n line segments, yielding the cost function

$$\Lambda^2 = \sum_{i=1}^{n} \lambda_i^2, \tag{4}$$

in which λ_i the i-th segment length, that can be easily expressed as the Euclidean distance between $P_{G,i}$ and $P_{G,i+1}$. Having denoted with $P_{x,i}$ and $P_{y,i}$ the component of $P_{G,i}$ along the x and y axes, the i-th segment length is

$$\lambda_i = \sqrt{(\Delta P_{x,i})^2 + (\Delta P_{y,i})^2}, \tag{5}$$

in which, having defined $\Delta x_i = x_{l,i} - x_{r,i}$ and $\Delta y_i = y_{l,i} - y_{r,i}$,

$$\begin{aligned} \Delta P_{x,i} &= \begin{pmatrix} \Delta x_{i+1} & -\Delta x_i \end{pmatrix} \begin{pmatrix} \alpha_{i+1} \\ \alpha_i \end{pmatrix} + \text{const}, \\ \Delta P_{y,i} &= \begin{pmatrix} \Delta y_{i+1} & -\Delta y_i \end{pmatrix} \begin{pmatrix} \alpha_{i+1} \\ \alpha_i \end{pmatrix} + \text{const}. \end{aligned} \tag{6}$$

Moreover, the path length cost function in (4) can be expressed in a quadratic form. More precisely, eq. (5) can be rewritten as

$$\Lambda^2 = \sum_{i=1}^{n} \Delta P_{x,i}^T \Delta P_{x,i} + \Delta P_{y,i}^T \Delta P_{y,i}. \tag{7}$$

Substituting eq. (6) in the previous equation, defining the vector

$$\bar{\alpha}_{\mathbf{i}} = \begin{pmatrix} \alpha_{i+1} \\ \alpha_i \end{pmatrix}, \tag{8}$$

and H_Λ, B_Λ two suitable constant matrices, the quadratic cost function is completely described as

$$\Lambda^2 = \sum_{i=1}^{n} \bar{\alpha}_i^T H_{\Lambda,i} \bar{\alpha}_i + B_{\Lambda,i} \bar{\alpha}_i + \text{const}. \tag{9}$$

The curvature $\sigma(s)$ of a given function parametrized by the distance traveled along the track s is defined as the rate change of the angle of the tangent to the function

$$\tan(\theta) = \frac{dy}{dx}, \tag{10}$$

$$\sigma(s) = \frac{d\theta}{ds} = \frac{\ddot{y}\dot{x} - \ddot{x}\dot{y}}{\dot{x}^2 + \dot{y}^2}. \tag{11}$$

Since the track has been discretized into line segment, the total curvature cost function is

$$\Gamma^2 = \sum_{i=1}^{n} \sigma_i, \tag{12}$$

being σ_i the curvature of the path evaluated at the i-th point. Therefore, the derivatives in eq. (11) have to be computed numerically. A convenient way to calculate these derivatives is to use a Taylor expansion series, defining $\dot{x}_i$ and $\ddot{x}_i$ as

$$\begin{aligned} \dot{x}_i &= \frac{1}{\Delta_{i-1,i} + \Delta_{i,i+1}} \left[\Delta_{i,i+1} \frac{\Delta^x_{i-1,i}}{\Delta_{i-1,i}} + \Delta_{i-1,i} \frac{\Delta^x_{i.i+1}}{\Delta_{i,i+1}} \right], \\ \ddot{x}_i &= \frac{2}{\Delta_{i-1,i} + \Delta_{i,i+1}} \left[\frac{\Delta^x_{i,i+1}}{\Delta_{i,i+1}} - \frac{\Delta^x_{i-1,i}}{\Delta_{i-1,i}} \right], \end{aligned} \tag{13}$$

in which $\Delta^x_{i,i+1} = x_{i+1} - x_i$ and $(\Delta_{i,i+1})^2 = (\Delta^x_{i,i+1})^2 + (\Delta^y_{i,i+1})^2$. Similar relations can be obtained for $\dot{y}$ and $\ddot{y}$. Substituting eq. (13) in (11), the total curvature cost function is obtained

$$\begin{aligned} \Gamma^2 = \sum_{i=1}^{n} & \left[\frac{y_{i+1} - 2y_i + y_{i-1}}{\Delta^2} \right] \left[\frac{x_{i-1} - x_{i-1}}{2\Delta} \right] + \\ & - \left[\frac{x_{i+1} - 2x_i + x_{i-1}}{\Delta^2} \right] \left[\frac{y_{i-1} - y_{i-1}}{2\Delta} \right]. \end{aligned} \tag{14}$$

In order to find the minimum of both the cost function the Optimization toolbox of MATLAB has been used. Moreover, to achieve the minimum lap time, it's required to find an optimal speed profile. As seen in eq. (1) the maximum speed allowed is proportional to $\frac{1}{\sigma}$. Therefore, considering both the mass of the vehicle and the maximum centripetal force the tyres can perform at a constant k, and by the knowledge of the curvatures σ_i of the already computed trajectory Γ, an initial guess of the desired speed at i-th point is

$$v_i = \sqrt{\frac{k}{\sigma_i}}. \tag{15}$$

According to eq. (2) the weight vector ϵ has to be chosen so that it weights more the minimum path length whenever the maximum speed achievable by the vehicle is lower than the desired one and the minimum curvature path otherwise. In conclusion, an optimal speed profile can be computed using the curvature of the already calculated optimal trajectory F.

D. Model Predictive Control

In the following, the design of a model based control is reported. We refer to Figures 6 and 7 for the map (global) and vehicle (local) frame quantities, respectively.

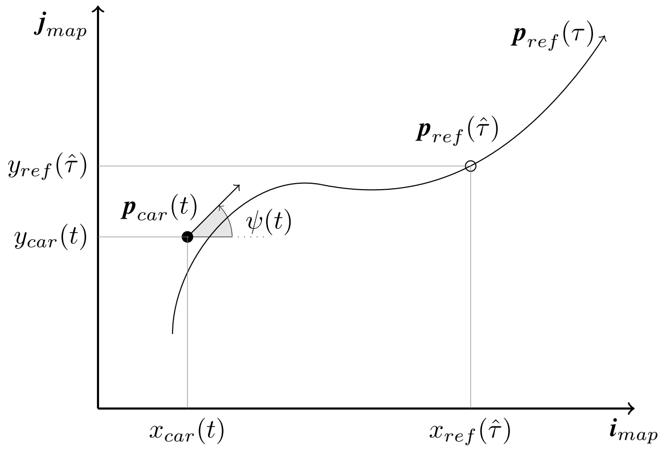


Fig. 6. Map frame

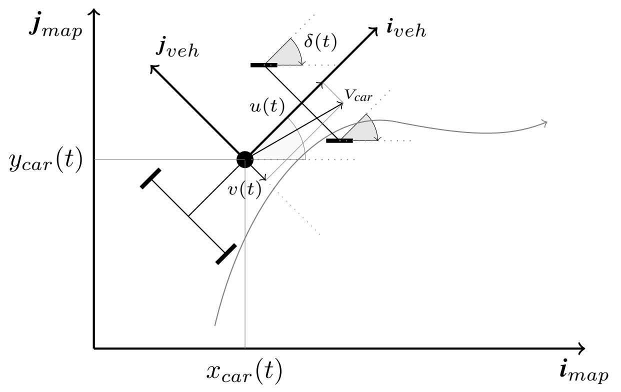


Fig. 7. Vehicle frame

The racing line reference trajectory in the map frame is $\boldsymbol{p}_{ref}(\tau) \in \mathbb{R}^2$, with

$$\boldsymbol{p}_{ref}(\tau) = \begin{cases} x_{ref}(\tau) \\ y_{ref}(\tau) \end{cases} \tag{16}$$

The vehicle state $\boldsymbol{x}(t)$ can be expressed as

$$\boldsymbol{x}(t) = \begin{cases} \boldsymbol{p}_{car}(t) = \begin{cases} x_{car}(t) & \textit{x-position of vehicle in map frame} \\ y_{car}(t) & \textit{y-position of vehicle in map frame} \end{cases} \\ \psi(t) & \textit{heading angle in map frame} \\ u(t) & \textit{longitudinal velocity in vehicle frame} \\ v(t) & \textit{lateral velocity in vehicle frame} \\ r(t) & \textit{heading angular rate in vehicle frame} \end{cases} \tag{17}$$

and the control input vector is

$$\boldsymbol{u}(t) = \begin{cases} \delta(t) & \textit{wheel steering angle} \\ F_t(t) & \textit{traction force}. \end{cases} \tag{18}$$

The nonlinear vehicle dynamics, considering a single track model [11], can be expressed as an autonomous system

$$\dot{\boldsymbol{x}}(\tau) = f(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau)) \tag{19}$$

We call Reference Horizon Generator (RHG), see Figure 8, the method by which we extract the portion of the racing line to be considered in the given prediction horizon T_{hor} given the current pose estimate of the vehicle in map frame $p_{car}(t)$. The beginning of the portion of the racing line to be considered is simply given by

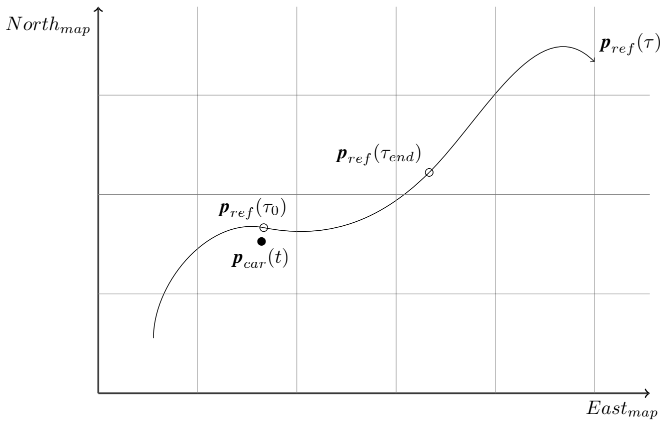
$$\tau_0 = \arg\min_{\tau} \left\| \boldsymbol{p}_{ref}(\tau) - \boldsymbol{p}_{car}(t) \right\|_2 \tag{20}$$

while the endpoint is given by

$$\tau_{end} = \tau_0 + T_{hor}. \tag{21}$$

Hence, the portion of the trajectory to be considered in the current time frame is

$$\boldsymbol{p}_{hor}(\hat{\tau}) = \boldsymbol{p}_{ref}(\tau - \tau_0), \quad \begin{matrix} \tau \in [\tau_0, \ \tau_{end}] \\ \hat{\tau} \in [0, \ T_{hor}] \end{matrix} \tag{22}$$



$North_{map}$
$\boldsymbol{p}_{ref}(\tau)$
$\boldsymbol{p}_{ref}(\tau_{end})$
$\boldsymbol{p}_{ref}(\tau_0)$
$\boldsymbol{p}_{car}(t)$
$East_{map}$

Fig. 8. Reference Horizon Generator

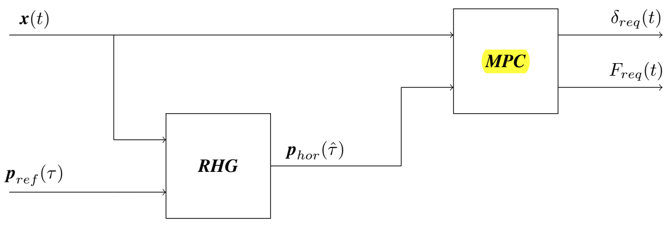
The state vector trajectory in the prediction horizon at instant t, assuming initial model state is the current measured state $\hat{\boldsymbol{x}}(0) = \boldsymbol{x}(t)$ and applying an input sequence $\hat{\boldsymbol{u}}(\hat{\tau})$, is

$$\hat{\boldsymbol{x}}(\hat{\tau}) = \hat{\boldsymbol{x}}(0) + \int_0^{\hat{\tau}} f(\hat{\boldsymbol{x}}(\tau), \ \hat{\boldsymbol{u}}(\tau)) \, d\tau, \quad \hat{\tau} \in [0, T_{hor}]. \tag{23}$$

The controller aims at finding the optimum input sequence $\boldsymbol{u}^*(\hat{\tau})$ such that the following cost function is minimized:

$$J = \int_0^{T_{hor}} \|\boldsymbol{p}_{hor}(\hat{\tau}) - \hat{\boldsymbol{p}}_{car}(\hat{\tau})\|_2 \, d\hat{\tau} \tag{24}$$

$$\boldsymbol{u}^*(\hat{\tau}) = \arg\min_{\boldsymbol{u}(\hat{\tau})} J \tag{25}$$



$\boldsymbol{x}(t)$
MPC
$\delta_{req}(t)$
$F_{req}(t)$
$\boldsymbol{p}_{ref}(\tau)$
RHG
$\boldsymbol{p}_{hor}(\hat{\tau})$

Fig. 9. Controller structure

E. Pure Steering Control

A different approach can be considered by decoupling the velocity and steering control, where the first can be obtained by means of a simple PID controller, whilst several approaches have been proposed in the literature for pure steering control. In this section we design a feedforward-feedback steering control architecture with the objective to minimize deviation from a desired path, based on the concept of center of percussion first introduced by Kritayakirana and Gerdes [12] for racing car control.

We consider the linearized single track model, whose dynamics is given by

$$\begin{pmatrix} \dot{\beta} \\ \dot{r} \\ \dot{e} \\ \Delta\dot{\Psi} \end{pmatrix} = \begin{pmatrix} \dfrac{F_{yf}+F_{yr}}{mu} - r \\ \dfrac{aF_{yf}-bF_{yr}}{I_z} \\ u(\beta+\Delta\Psi) \\ r-\dot{s}k \end{pmatrix}, \tag{26}$$

where m and I_z are the vehicle mass and moment of inertia, and s is the distance along the reference path. The error that was considered is the *lookahead error*, a weighted combination of the lateral error e and the heading error $\Delta\Psi$, given by

$$e_p = e + x_p\Delta\Psi\,. \tag{27}$$

The *lookahead error* can be considered as a linear approximation of the lateral error projection, at distance x_p, in front of the vehicle. Since e and $\Delta\Psi$ are state variables, the feedback steering control law can be expressed as a linear state feedback:

$$\begin{aligned} \delta_{FB} &= -k_p e_p \\ &= -k_p(e + x_p\Delta\Psi), \end{aligned} \tag{28}$$

where k_p is a proportional gain.

The feedforward steering was introduced to minimize the level of compensation required by the steering feedback. The objective in designing a feedforward steering is to estimate the steering angle required to traverse a path with a known curvature and velocity profile. Let us consider the second time derivative of e_p:

$$\ddot{e}_p = \frac{F_{yf}+F_{yr}}{m} - uk\dot{s} + x_p\frac{aF_{yf}-bF_{yf}}{I_z} - x_p(k\ddot{s}-\dot{s}\dot{k}) \tag{29}$$

Eliminating $\ddot{e}_p$ is a necessary condition for the vehicle to traverse a path with curvature k. Since it not possible to act directly on the rear lateral tyre force commanding the front steering angle, the terms x_p has be to chosen to eliminate

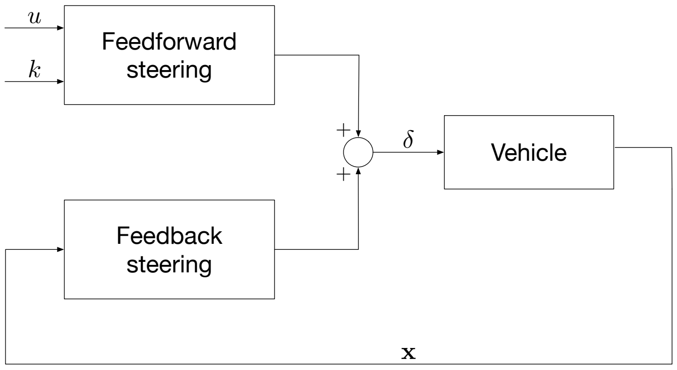


Fig. 10. Feedforward/feedback steering control

the dependence on $\ddot{e}_p$ from F_{yr}. To this purpose, we use the center of percussion as a convenient projection point, which is defined as

$$x_{cop} = \frac{I_z}{bm}. \tag{30}$$

In fact, substituting $x_p = x_{cop}$ it is possible to eliminate $\ddot{e}_p$ with the only front lateral tyre force

$$F_{yf} = \frac{mb}{L}(uk + x_{cop}(k\ddot{s} + \dot{k}\dot{s})). \tag{31}$$

Assuming steady-state conditions and small angles, simplified expressions of the lateral tyre forces that guarantees the vehicle to travel along a path of curvature k with a longitudinal velocity u are:

$$F_{yf}^{ss} = \frac{mb}{L}u^2k\,, \tag{32}$$

$$F_{yr}^{ss} = \frac{ma}{L}u^2k\,. \tag{33}$$

Assuming small tyre slip angles α_f and α_r, the rear and lateral tyre forces can be expressed by the linear relations:

$$F_{yf} = C_f\alpha_f\,, \tag{34}$$

$$F_{yr} = C_r\alpha_r \tag{35}$$

where C_r and C_f are the tyre cornering stiffness. From kinematic consideration, the feedforward steering able to keep the vehicle in a path of curvature k is equal to

$$\delta_{FFW} = Lk - \alpha_f^{FFW} + \alpha_r^{FFW}\,. \tag{36}$$

To obtain an expression of the feedforward steering dependent from k and u, the lateral tyre forces were used to substitute α_f and α_r to eq.(36). The resulting feedforward steering is

$$\delta_{FFW} = Lk + \frac{K_{ug}}{g}u^2k\,. \tag{37}$$

where K_{ug} is the understeer gradient, equal to

$$K_{ug} = \frac{F_{zr}}{C_r} - \frac{F_{zf}}{C_f}\,. \tag{38}$$

The terms F_{zf} and F_{zr} are respectively the front and rear vertical loads of the vehicle.

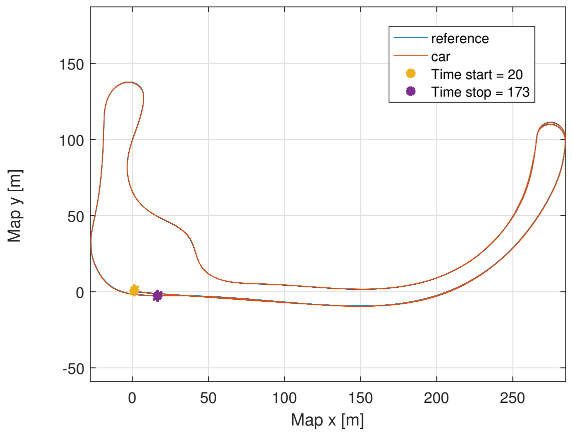


Fig. 11. Tracking of a racing line over two laps.

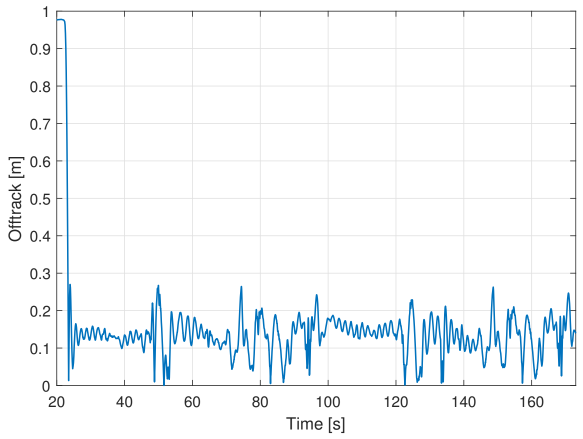


Fig. 12. Offtrack over two laps.

The resulting steering control can be computed as

$$\delta = \delta_{FB} + \delta_{FFW} \tag{39}$$

$$= \begin{bmatrix} 0 & 0 & -k_p & -k_p x_p \end{bmatrix} \mathbf{x} + \left(L + \frac{K_{ug} u^2}{g} \right) k\,, \tag{40}$$

being $\mathbf{x}$ the state vector

$$\mathbf{x} = \begin{bmatrix} \beta & r & e & \Delta\Psi \end{bmatrix}^T. \tag{41}$$

IV. Experimental Results

Once the map of the track has been obtained and a racing line with an optimal speed profile has been generated, the DevBot has been driven in autonomous mode to track such racing line, in the track shown in Figure 4. In Figure 11 tracking results over two laps are reported, with the lateral tracking error (*offtrack*) shown in Figure 12. A sensor fusion algorithm has been developed to provide accurate vehicle state as an input to the nonlinear MPC controller.

V. Conclusions and Future Work

In this work we have presented a racing scenario for fully-autonomous vehicles. We have shown how the problem can be approached and several solutions have been proposed based on control methods and vehicle dynamics results. Future AI racing drivers would have to rely more on artificial intelligence algorithms, that can be trained and evolve autonomously based on experience. The question remains on how to develop personalities for such drivers, e.g. how to tune the choice between aggressive or more gentle maneuvers, and achieve formal guarantees for safe behavior. Both these questions are well suited to be developed and tested in a racing scenario.

Acknowledgment

The authors would like to thank Roborace and all the members, advisors and supporters of Team Roborace Italia (http://www.roboteamitalia.it/) for making this work possible. In particular, the Autodromo di Imola Enzo e Dino Ferrari for the kind collaboration and useful insights.

References

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

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

[3] A. Nunes, B. Reimer, and J. F. Coughlin, "People must retain control of autonomous vehicles." *Nature*, vol. 556, no. 7700, p. 169, 2018.

[4] , *Taxonomy and Definitions for Terms Related to On-Road Motor Vehicle Automated Driving Systems*, jan 2014. [Online]. Available: https://doi.org/10.4271/J3016$_$201401

[5] A. Saccon, J. Hauser, and A. Beghi, "Trajectory exploration of a rigid motorcycle model," *IEEE Transactions on Control Systems Technology*, vol. 20, no. 2, pp. 424–437, 2012.

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

[7] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard, "3-d mapping with an rgb-d camera," *IEEE Transactions on Robotics*, vol. 30, no. 1, pp. 177–187, 2014.

[8] F. Braghin, F. Cheli, S. Melzi, and E. Sabbioni, "Race driver model," in *Computers & Structures*, vol. 86, no. 13-14, March 2007, pp. 1503–1516.

[9] M. Tipping, M. Hatton, and R. Herbrich, "Racing line optimization," in *US Patent*, March 2013.

[10] L. Cardamone, D. Loiacono, P. Lanzi, and A. Bardelli, "Searching for the optimal racing line using genetic algorithms," in *Computational Intelligence and Games (CIG)*, August 2010.

[11] M. Guiggiani, *The science of vehicle dynamics: handling, braking, and ride of road and race cars*. Springer Science & Business Media, 2014.

[12] K. Kritayakirana and J. C. Gerdes, "Using the centre of percussion to design a steering controller for an autonomous race car," *Vehicle System Dynamics*, vol. 50, no. sup1, pp. 33–51, 2012.