This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2023.3330431

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2023.0322000

Fast Planning and Tracking of Complex Autonomous Parking Maneuvers with Optimal Control and Pseudo-Neural Networks

EDOARDO PAGOT[1]*, MATTIA PICCININI[1]*, (Member, IEEE), ENRICO BERTOLAZZI[1], and FRANCESCO BIRAL[1]

[1]Department of Industrial Engineering, University of Trento, 38123 Trento, Italy (e-mail: edoardo.pagot@unitn.it; mattia.piccinini@unitn.it; enrico.bertolazzi@unitn.it; francesco.biral@unitn.it)

*These authors contributed equally to this work.

Corresponding author: Mattia Piccinini (e-mail: mattia.piccinini@unitn.it).

ABSTRACT This paper presents a framework to plan and execute autonomous parking maneuvers in complex parking scenarios. We formulate a minimum-time optimal control problem for trajectory planning, using an indirect optimal control approach. A novel smooth penalty function is devised for collision avoidance with optimal control, and an effective technique is adopted to compute an initial solution guess. The trajectory planning tasks are solved with low computational times, and a dense mesh is used to discretize the domain of the optimal control problems, resulting in accurate collision-free solutions. The planned parking maneuvers are tracked with an original pseudo-neural feedforward-feedback steering controller, which outperforms other techniques from the literature, and a feedback longitudinal controller, to drive a realistic 14-degree-of-freedom vehicle simulator. We validate the planning and tracking algorithms in challenging narrow parking scenarios, including reverse, parallel and angle parking, and unstructured environments. The framework robustness is assessed by changing the vehicle mass, the road adherence conditions, and by introducing measurement noise with realistic sensor models. A video of the trajectory planning and tracking results is available as supplementary material.

INDEX TERMS Autonomous parking, optimal control, trajectory optimization, trajectory tracking.

I. INTRODUCTION

Current developments in intelligent vehicle technology focus on several aspects of autonomous driving, including autonomous parking and valet parking. Valet parking is the ability to autonomously cruise for a free parking spot, while autonomous parking refers more specifically to the execution of a parking maneuver. This paper focuses on planning and executing autonomous parking maneuvers.

The expected benefits of autonomous parking and valet parking are several. There are technical advantages, such as the ability to move and park in narrower spaces, compared to human-driven parking. Moreover, an autonomous system can find the fastest and shortest parking path, and it can minimize the time to search for a parking spot—which is anything but a minor matter. In the New York City region alone, vehicles cruising for a free parking space travel more than 70,000 miles every day. This equals to a daily emission of 29 metric tons of CO_2 [1]. Hence, autonomous parking systems can considerably impact greenhouse emissions. There are also advantages related to passenger comfort: for example, a passenger can exit the vehicle and let it look for a parking spot by itself. The vehicle could be parked in narrow spots where the driver is precluded from opening the doors, or where drivers with reduced mobility would not be able to dismount from the vehicle.

An autonomous parking framework usually consists of three main functional parts: (a) environment mapping and on-line vehicle state estimation; (b) trajectory planning, to compute a feasible path and velocity profile; (c) motion tracking, to execute the planned trajectory. In this work, we focus on trajectory planning and tracking (points (b) and (c)), but we finally analyze the robustness of our framework to measurement noise, in a realistic mapping and state estimation simulation scenario.

The following subsection critically analyses the strengths and limitations of related literature papers, after which the main paper contributions are outlined.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2023.3330431

A. RELATED WORK

In the field of path and trajectory planning for automated parking, the main techniques that emerge in the literature [2] are search-based, sampling-based, and optimal control-based. The search methods, like the many variants of A* and Hybrid A* [3], [4], and the sampling RRT*-like techniques [5], [6] tend to suffer from curse of dimensionality issues, when the parking spaces are narrow and a fine map resolution is required [2], [7]. Moreover, the search A*-like methods are generally used for *path planning*, and the computed path may be far from the global optimum. Conversely, optimal control-based methods can be used for *trajectory planning*, i.e., to compute both a path and a velocity profile. The computed trajectory is at least locally optimal, according to the chosen objective function, constraints, and vehicle dynamics model.

In recent years, several authors used optimal control (OC) to solve trajectory planning problems for autonomous parking. The authors of [8] used optimal control and direct collocation to compute minimum-time parking maneuvers in generic scenarios with multiple obstacles; however, their computational times were far from real-time. The previous work was then extended in [9], in which a look-up table of pre-computed OC solutions was created and provided as guess for the minimum-time optimal control problem. The framework was successfully tested on narrow parking spots; however, only the parallel parking scenario was studied, and the computational times were in the order of 40-180 seconds, thus far-fetched for a real-time application. In [10], the authors obtained the parking maneuvers from the solution of a combined minimum-time and minimum-space optimal control problem (OCP). The OCP was initialized by means of standard maneuvers from a look-up table. They tuned and tested their framework on reverse parking only, and the computational times were in the order of 100 seconds. In [11], a multi-stage Hybrid A* was developed to compute a coarse path in unstructured parking scenarios, and the solution was refined by solving an optimal control problem in an obstacle-free corridor. Similarly, [12] used an enhanced Hybrid A* method to compute an obstacle-free corridor, in which an OCP was solved for valet parking navigation problems. The authors of [13] presented an iterative framework for autonomous parking, based on the direct optimal control method. They reported promising computational times lower than 1 second; however, the framework was validated on quite trivial scenarios, that were closer to a navigation problem than to a parking one. The authors of [14] developed a machine learning (ML) framework based on a Monte Carlo tree search, able to learn optimal parking maneuvers. The ML algorithm was trained with the solutions of minimum-time optimal control problems solved with direct collocation, which were computed offline; the computational time for the offline solution of the OCPs is not provided by the authors, and the framework was tested only on parallel parking scenarios.

Recently, some papers used reinforcement learning [15], [16] for trajectory planning in autonomous parking. However, they showed only examples of parallel parking scenarios.

Finally, a remarkable application of optimal control for autonomous parking is [17], where the authors solved a minimum-time OCP with a convexification technique, in order to obtain an exact and smooth formulation of the original, non-convex OCP. They employed the Hybrid A^* algorithm [3] to compute solution guesses for the optimal control problem, obtaining average computational times lower than 2-3 seconds, and testing their framework on both reverse and parallel parking scenarios. However, the parking spots were relatively large, and the resulting parking maneuvers were quite simple, being always composed of only two driving segments, the first in the forward direction, and the second in the rearward direction. Moreover, to reduce the computational time, the discretization grid of the OCP was quite coarse: as pointed out by the authors, some collisions with the obstacles are visible in the plotted OCP solutions, in the time frame between two consecutive discretization grid points.

To summarize, the main difficulties that hinder accurate and computationally fast OC trajectory planning for autonomous parking are typically related to: (1) the need to use a dense discretization planning grid; (2) the need to plan complex maneuvers, with many segments in the forward and rearward direction; (3) the non-convexity of the planning domain; (4) the presence of many local minima, and hence the need for a good solution guess.

Regarding the closed-loop motion tracking and execution of parking or low-speed maneuvers, various low-level controllers can be found in the literature [18]–[21]. For example, in [6] a pure-pursuit steering controller [22] was used for path tracking, and a PI longitudinal controller for speed tracking. The authors in [23] used a short-horizon MPC to track a given parking trajectory. They linearized a kinematic vehicle model around the reference trajectory, obtaining a quadratic problem to be solved online. A sliding mode path tracking controller was used in [24], in combination with a PI speed tracking regulator. Other authors used prescribed performance function-based controllers for trajectory tracking with collision avoidance [25], [26]. A Lyapunov-based feedback controller was presented [27], for both speed and path tracking. In [28], an integrated longitudinal-lateral neural network controller for autonomous parking was developed. However, their approach requires a very large training dataset, since the neural network structure is generic, and they show only parallel parking examples.

B. RESEARCH QUESTION

The research question of our paper is the following:
Given a high-fidelity vehicle simulator, develop a framework for online fast trajectory planning with optimal control and low-level tracking of complex automated parking maneuvers. The framework must operate in a variety of real-life parking scenarios, including narrow parallel, reverse and angle parking spots, and unstructured environments. The trajectory

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2023.3330431

tracking algorithms must be sufficiently robust to unknown variations in the vehicle's parameters and road adherence, and to measurement noise.

To the best of our knowledge, no existing literature paper answers the previous research question in its entirety. More precisely, the works mentioned in Section I-A are limited by at least one of the following factors:

1) Real-time optimal control planning and tracking of accurate parking maneuvers was not performed in both narrow parallel/reverse/angle parking spots and unstructured environments. Specifically, the authors of [17] solved their OCPs with acceptable computational times, but they used a quite coarse discretization mesh, which yielded local obstacle collisions in narrow parking spots. The authors of [11] dealt with generic parking scenarios, but they solved the OCP in an obstacle-free corridor, and without very narrow parking spots.
2) Many authors, like [9]–[11], [17], focused mainly on trajectory planning, but they did not properly analyze the maneuver execution phase, with high-fidelity multibody vehicle simulators. Moreover, they did not consider model mismatches and sensor noise.
3) Most of the optimal control-based papers used direct collocation approaches, while indirect optimal control methods —based on the Pontryagin's Maximum Principle (PMP)— are still unexplored in the field of autonomous parking.
4) Some papers, like [28], used neural network controllers for motion tracking, but they required very large training datasets, and they focused on a limited set of parking scenarios.

C. PAPER CONTRIBUTION AND STRUCTURE

Our paper aims to answer the research question of Section I-B, through the following contributions:

1) We present a novel autonomous parking framework that solves minimum-time optimal control problems (OCPs) by means of an indirect method. Our framework is able to deal with narrow parking spots and generic unstructured parking scenarios, computing complex maneuvers with many forward/reverse driving segments, in a single optimization. In comparison with the direct collocation literature methods, we employ an indirect optimal control method, implemented by our software tool PINS. Using PINS and an effective OCP formulation, low computational times are obtained even with a dense discretization time grid, that yields accurate solutions near the obstacles.
2) We devise a novel method to deal with collision avoidance in the optimal control problems, using penalty functions described as smooth three-dimensional functions. Our formulation allows the creation of arbitrary scenarios, by combining multiple rectangular obstacles of different sizes.
3) A novel scheme is adopted to generate the solution guess for the OCP, combining a Hybrid A* algorithm and an optimal control tracking problem.
4) An original pseudo-neural physics-driven steering controller is illustrated, which enables an accurate closed-loop tracking of the optimal control trajectories on a complex 14-degree-of-freedom vehicle simulator.
5) The parking framework (planning and tracking) is validated on a wide range of complex and non-conventional maneuvers. Moreover, we analyze the framework's robustness to changes in vehicle parameters and road adherence, and to localization noise due to sensors.

This paper is organized as follows. Section II presents an overview of the trajectory planning and control framework, and of the vehicle simulator to be controlled. Section III outlines the optimal control problem formulation for trajectory planning, and the warm-start strategy to generate a good guess for the problem. Section IV deals with the trajectory tracking controllers, with a special focus on the pseudo-neural feedforward-feedback steering controller. The main results are described in Section V: we first analyze the planned and executed maneuvers in a wide range of challenging parking scenarios. Then, we evaluate the framework's robustness to variations in the vehicle parameters and road adherence, and to vehicle's localization errors due to measurement noise. Finally, in Section VI we outline our conclusions and the future work.

II. FRAMEWORK OVERVIEW

In this section, we give an overview of the presented autonomous parking framework. The trajectory planning and tracking scheme is provided in Fig. 2. The framework plans and executes parking maneuvers with a custom-designed vehicle simulator (VS), which accurately reproduces the dynamics of a sedan car.

A. VEHICLE SIMULATOR

With the target to reproduce the dynamical behavior of a real sedan car, the VS is implemented as a high-fidelity 14-degree-of-freedom multibody model. The Pacejka MF-6.2 formulation [29] is adopted for the tire forces and moments, and the VS can be used on three-dimensional roads [30]. The vehicle suspensions are modeled with kinematic and compliance maps. An efficient formulation is obtained for the VS multibody model through symbolic manipulation, so that the VS supports real-time hardware-in-the-loop (HIL) simulations.

The VS is integrated into the driving simulator of Fig. 1, in the Autonomous Driving Laboratory at the University of Trento (Italy). The VS employed in this paper was also used in [31], to validate an artificial race driver, with minimum-time maneuvers. The reader is referred to [32, Sections V-VI] for further implementation details about the VS. The VS simulates a front-wheel-drive (FWD) sedan car, with an

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2023.3330431



FIGURE 1. Vehicle driving simulator in the Autonomous Driving Laboratory of the University of Trento (Italy). The simulator uses real-time hardware to implement the presented VS model.

Parameter	Symbol	Value
Vehicle mass	m	1300 kg
Center of mass (CoM) height	h_G	0.285 m
Front axle distance from the CoM	L_1	1.225 m
Rear axle distance from the CoM	L_2	1.375 m
Wheelbase	L	2.60 m
Yaw inertia	I_z	1400 kg m^2
Track width (front)	W_1	1.80 m
Track width (rear)	W_2	1.80 m
Steering ratio	τ_d	20

TABLE 1. Main parameters of the vehicle simulator.

electric motor and an open mechanical differential. The main parameters of the VS are reported in Table 1.

In this work, the internal structure and the parameters of the VS are assumed to be unknown, so that the VS is used as a black-box model. Moreover, we will test the robustness of our autonomous parking framework by varying some parameters of the VS (vehicle mass and road adherence), without changing/re-training the models for trajectory planning and control.

B. TRAJECTORY PLANNING AND CONTROL OVERVIEW

Given the desired initial and final vehicle pose, and the positions of the obstacles, an optimal control problem (OCP) plans the parking trajectory (path and speed profile). A feedforward steering controller is implemented with a novel pseudo-neural physics-driven structure. Such a pseudo-neural controller computes the steering angle δ_{ff_k} for the current time step k, using a window of present and future optimal trajectory curvature values $[\rho_k, \ldots, \rho_{k+q}]$. A feedback steering proportional controller computes a steering angle δ_{fb_k}, to compensate the path (x, y) and yaw angle (θ) tracking errors. The total steering wheel angle is computed as $\delta_{D_k} = \delta_{\mathrm{ff}_k} + \delta_{\mathrm{fb}_k}$, and it is an input for the VS. Finally, a feedback longitudinal PI controller is used to track the optimal forward speed v_x profile, computing the requested pedal p, with $-1 \leq p \leq 1$ ($p < 0$ for braking, $p > 0$ for throttle).

The optimal control problem is solved only once, before the maneuver is executed, but we will show that the computational times are low enough for a real-world application. During the maneuver execution, the optimal control solution is extracted and fed to the feedforward and feedback tracking controllers, which operate in real-time at 1 kHz. The vehicle simulator is also executed at 1 kHz.

In Section V-D, we will test the planning and control scheme of Fig. 2 in a realistic localization and state estimation simulation framework, to evaluate the scheme's robustness to measurement errors.

III. TRAJECTORY PLANNING

This section describes the OCP employed for trajectory planning, the novel penalty function for collision avoidance, and the original method to generate a guess solution for the OCP.

A. VEHICLE MODEL FOR OPTIMAL CONTROL

In the optimal control problem formulation, we adopt a kinematic model of the vehicle motion:

$$\begin{cases} \dot{x}(t) = v_x(t)\cos(\theta(t)) & \text{(1a)} \\ \dot{y}(t) = v_x(t)\sin(\theta(t)) & \text{(1b)} \\ \dot{\theta}(t) = v_x(t)\tan(\delta(t))/L & \text{(1c)} \\ \dot{v}_x(t) = a_x(t) & \text{(1d)} \\ \tau_\delta \dot{\delta}(t) = \delta_{\mathrm{dot}}(t) & \text{(1e)} \end{cases}$$

where the notation $\dot{\mathrm{x}}$ indicates the derivative of the quantity x with respect to time t. Referring to Fig. 3, x and y are the coordinates of the origin of the vehicle's moving frame —centered on the vehicle's rear axle—, with respect to the ground frame X-Y. The angle θ is the orientation of the vehicle's moving frame, with respect to the ground frame. v_x is the vehicle forward velocity, while δ is the Ackermann steering angle (at the center of the front axle). The model states are $x = [x, y, \theta, v_x, \delta]$, while the controls are $u = [a_x, \delta_{\mathrm{dot}}]$, where a_x is the vehicle forward acceleration and δ_{dot} is the angular velocity of the steering wheel. The parameter τ_δ in (1e) is the transmission ratio of the steering mechanism. A reasonable limit on δ_{dot} is imposed in order to consider the maximum angular speed reachable by real-world electric steering wheel actuators. Finally, L is the vehicle wheelbase.

We underline that the steering angle δ computed by the OCP with the kinematic model (1c,1e) is *not* directly used to control the vehicle simulator, whose steering dynamics is more complex and nonlinear. Section IV-A will present a feedforward steering controller, which is designed and trained to learn a more accurate model of the steering dynamics, to track the trajectory curvature profile returned by the OCP.

B. OBSTACLE FORMULATION FOR OPTIMAL CONTROL

We introduce a novel penalty function to deal with collision avoidance in the optimal control problem. The penalty

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2023.3330431

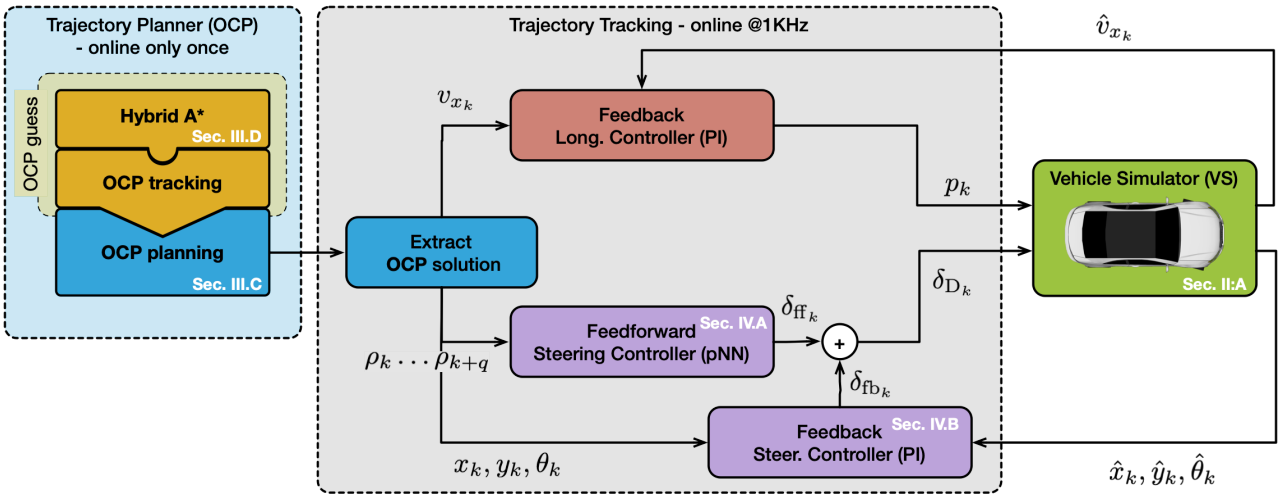


FIGURE 2. **Overview of the trajectory planning and tracking framework. The OCP plans the parking trajectory only once, with low computational times, and the maneuver is then tracked by the feedforward and feedback controllers at a 1 kHz rate. In each control block, we write the paper section that describes it.**

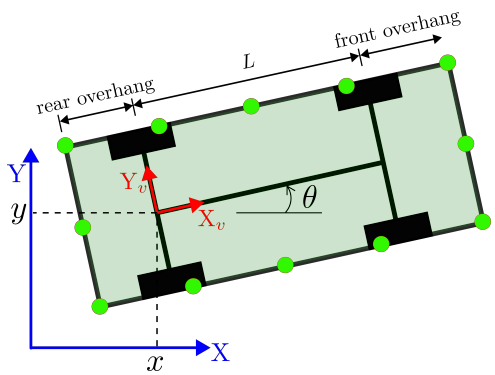


FIGURE 3. **Schematic of the ego vehicle: the quantities $\{x, y\}$ and θ represent the position and orientation of the vehicle's moving frame X_v-Y_v (in red), with respect to the ground frame X-Y (in blue). The green dots are the collision checking points used to evaluate the obstacle penalties. The vehicle wheelbase is L. Notice that the front and rear overhangs of the vehicle are considered in the model.**

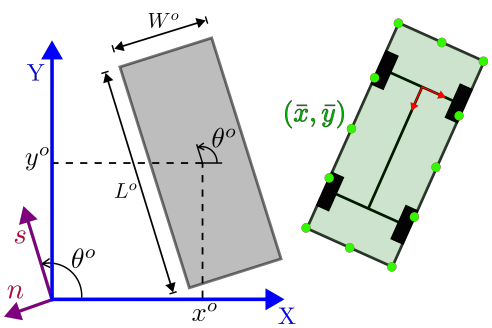


FIGURE 4. **In the optimal control problem, an obstacle (depicted in grey) is modeled using its absolute position $[x^o, y^o]$ and orientation θ^o, with respect to the X-Y ground frame, and its width W^o and length L^o. The penalty function $P(\cdot)$ in (2) takes as input the obstacle geometry and one of the collision checking points $[\bar{x}, \bar{y}]$ (green dots), belonging to the ego vehicle and expressed in the X-Y ground frame. The reference frame s-n is rotated by the obstacle orientation angle θ^o, and is used to ease collision checking with generically oriented obstacles.**

function enforces the geometry of a given obstacle: referring to Fig. 4, an obstacle (shown in grey) is defined in terms of its absolute position $[x^o, y^o]$, orientation θ^o, width W^o and length L^o. The arguments of the penalty function are the obstacle geometry parameters and the coordinates $[\bar{x}, \bar{y}]$ of an arbitrary point. As will be explained next, the points $[\bar{x}, \bar{y}]$ passed as arguments of the penalty function belongs to a set of several collision checking points defined along the ego vehicle's perimeter (green dots in Fig. 4). The penalty function is named $P(\cdot)$, and is expressed as:

$$P(\bar{x}, \bar{y}, x^o, y^o, \theta^o, W^o, L^o) = Q(\bar{s}, L^o, s^o) \cdot Q(\bar{n}, W^o, n^o) \quad (2)$$

$$\text{with} \quad \begin{cases} [\bar{s}, \bar{n}]^T = \mathbf{R}(\theta^{\mathbf{o}}) \cdot [\bar{x}, \bar{y}]^T & (3a) \\ [s^o, n^o]^T = \mathbf{R}(\theta^{\mathbf{o}}) \cdot [x^o, y^o]^T & (3b) \end{cases}$$

To consider the obstacle orientation angle θ^o, equations (3a,3b) are used to perform a change of reference frame: the collision checking point $[\bar{x}, \bar{y}]$ and the obstacle center $[x^o, y^o]$, defined in the ground frame X-Y, are now expressed in the rotated s-n frame (Fig. 4). In (3a,3b), the notation $[\cdot]^T$ indicates the transpose of a vector, while the rotation matrix $\mathbf{R}(\theta^{\mathbf{o}})$ is given by:

$$\mathbf{R}(\theta^{\mathbf{o}}) = \begin{bmatrix} \cos(\theta^o) & \sin(\theta^o) \\ -\sin(\theta^o) & \cos(\theta^o) \end{bmatrix} \quad (4)$$

The penalty $P(\cdot)$ in (2) is built by multiplying two smooth functions, named $Q(\cdot)$ and plotted in Fig. 5a. The $Q(\cdot)$ functions use as arguments the transformed coordinates of the collision checking point $[\bar{s}, \bar{n}]$ and of the obstacle center $[s^o, n^o]$, computed with (3a,3b). As shown in Fig. 5a, the value of the function $Q(\bar{s}, L^o, s^o)$ increases if $s^o - \frac{L^o}{2} \leq \bar{s} \leq s^o + \frac{L^o}{2}$, i.e., if the collision checking point is inside the obstacle shape, along the s direction.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2023.3330431

The function $Q(\cdot)$ is computed as the first-order derivative of a smooth *Clip* function, and is written as[1]:

$$Q(\bar{s},\, L^o,\, s^o) = \frac{1}{2\,h_t}\left(\bar{s} - s_0 + \frac{L^o}{2}\right)\frac{1}{\sqrt{1+\frac{1}{h_t{}^2}\left(\bar{s} - s_0 + \frac{L^o}{2}\right)^2}} + \\ - \frac{1}{2\,h_t}\left(\bar{s} - s_0 - \frac{L^o}{2}\right)\frac{1}{\sqrt{1+\frac{1}{h_t{}^2}\left(\bar{s} - s_0 - \frac{L^o}{2}\right)^2}} \tag{5}$$

where h_t is a tunable smoothing parameter, whose effect is shown in Fig. 5a. The function $Q(\cdot)$ in (5) is computed along the s direction: by changing its arguments, the function is also evaluated along the n direction, and used in (2). Fig. 5b shows an example of the obstacle penalty function $P(\cdot)$, obtained with (2) and $[x^o, y^o] = [0, 0]$ m, $\theta^o = 0$ deg, $W^o = 2$ m, $L^o = 3$ m. Note that, when $\theta^o = 0$ deg, the obstacle s-n directions are aligned with the ground reference frame X-Y (Fig. 4). The penalty function in (2) models the rectangular bounding box of an obstacle. However, combining multiple penalties allows us to deal with obstacles of generic shapes, as will be shown in the angle parking example of Section V-A.

Finally, for collision checking we define 12 points on the rectangular bounding box of the ego vehicle (green dots in Fig. 3 and Fig. 4): the 4 corners of the box, 3 points equally spaced along the right and left sides of the vehicle, and the 2 middle points of the front and rear edges of the box. To consider the i-th obstacle, located in $[x_i^o, y_i^o]$ with orientation θ_i^o and dimensions W_i^o, L_i^o, we augment the cost function of the OCP (Section III-C) with the sum of the values of the penalty $P(\bar{x}(t), \bar{y}(t), x_i^o, y_i^o, \theta_i^o, W_i^o, L_i^o)$, evaluated in the time-varying coordinates $[\bar{x}(t), \bar{y}(t)]$ of each of the 12 points of the ego vehicle. The choice of 12 points for collision checking ensures that the bounding box of the ego car does not collide with the obstacles. The value of each penalty is zero when the corresponding collision checking point is outside the obstacle bounding box, and greater than zero if the collision checking point is inside the obstacle area. Local collisions of the ego bounding box with an obstacle could still happen in between two collision checking points; however, in our results we never encountered this event. Indeed, we started our experiments with only 4 points, and the number of 12 collision checking points represents the lower bound for which no collisions occurred, in all the test scenarios of this paper.

C. OPTIMAL CONTROL PROBLEM FORMULATION

Exploiting the devised collision avoidance penalty function, we solve the following optimal control problem for trajectory planning of parking maneuvers:

$$\min_{u\in\mathcal{U}} \; w_T t_f + \int_0^{t_f} \ell_{\text{obst}}(t)\,\mathrm{d}t \tag{6a}$$

$$\text{s.t.} \;\; \dot{x}(t) = f(x(t), u(t)), \tag{6b}$$

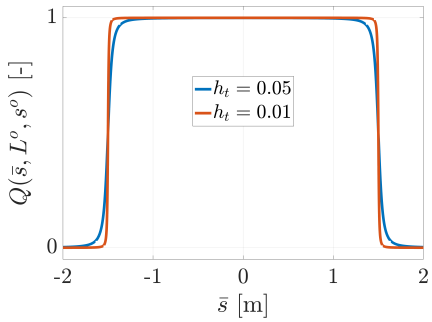
$$|\delta(t)| \le \delta_{\max},\; |v_x(t)| \le v_{x_{\max}}, \tag{6c}$$

$$|\delta_{\text{dot}}(t)| \le \delta_{\text{dot}_{\max}},\; |a_x(t)| \le a_{x_{\max}}, \tag{6d}$$

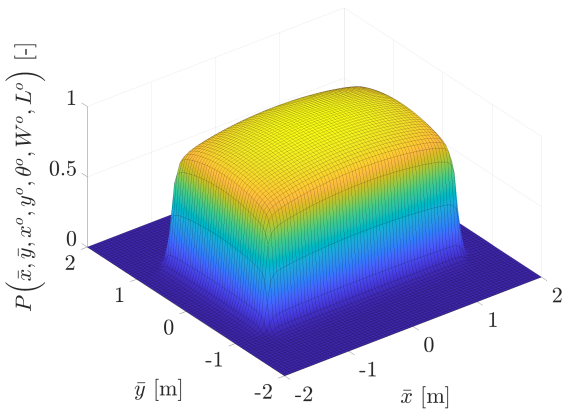
$$x(0) = x_{\text{ini}},\; y(0) = y_{\text{ini}},\; \theta(0) = \theta_{\text{ini}},\; v_x(0) = 0, \tag{6e}$$

$$x(t_f) = x_f, y(t_f) = y_f, \theta(t_f) = \theta_f, v_x(t_f) = 0 \tag{6f}$$

We remark that the problem (6a)-(6f) is *not* formulated to track a pre-calculated path, but rather to freely plan a collision avoidance trajectory. The cost function (6a) has two main purposes: minimize the maneuver time t_f, and avoid collisions. The former goal is weighted by the tunable parameter w_T. Collision avoidance is performed with the integral term in (6a), where $\ell_{\text{obst}}(t)$ has the following expression:



(a)



(b)

FIGURE 5. **(a) Example of $Q(\bar{s}, L^o, s^o)$ function (5), used for collision avoidance along the obstacle longitudinal s direction. The plotted example uses $s^o = 0$ m and $L^o = 3$ m: the function $Q(\cdot)$ increases if $s^o - \frac{L^o}{2} \le \bar{s} \le s^o + \frac{L^o}{2}$, i.e., if the collision checking point is inside the obstacle shape, along the s direction. (b) Example of the three-dimensional obstacle penalty function (2), with parameters $x^o = 0$ m, $y^o = 0$ m, $\theta^o = 0$ deg, $W^o = 2$ m, $L^o = 3$ m. The value of the penalty is zero when the collision checking point coordinates $[\bar{x}, \bar{y}]$ are outside the obstacle bounding box, and greater than zero when the coordinates $[\bar{x}, \bar{y}]$ are inside the obstacle area.**

[1]We can tune the maximum value of the function $Q(\cdot)$ by means of an additional scaling function, here not reported for brevity.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2023.3330431

$$\ell_{\text{obst}}(t) = \sum_{i=1}^{n_{\text{obs}}} \sum_{j=1}^{n_{\text{pts}}} P\big(x_j(t), y_j(t), x_i^o, y_i^o, \theta_i^o, W_i^o, L_i^o\big) \quad (7)$$

$\ell_{\text{obst}}(t)$ in (7) is the sum of the values of the novel penalty function $P(\cdot)$, computed for all the $n_{\text{pts}} = 12$ collision checking points of the ego vehicle (Fig. 3), and for all the n_{obs} obstacles. In (7), $[x_j(t), y_j(t)]$ are the coordinates of the j-th collision checking point of the ego vehicle; $[x_i^o, y_i^o, \theta_i^o]$ are the coordinates and orientation of the i-th obstacle, whose width and length are W_i^o and L_i^o. The expression of the penalty function $P(\cdot)$ is given by (2). The relative weight between the two objectives in the cost function (6a) is tuned so that no collisions occur, in all the 75 maneuvers on which the framework is tested. Also, the absolute weight on the final-time objective w_T is large enough to enforce parking maneuver times that are well inside the range of human drivers' parking times.

The dynamical system (6b) is the kinematic vehicle model (1). Box constraints (6c,6d) are imposed for the states $\{\delta, v_x\}$ and the controls $\{\delta_{\text{dot}}, a_x\}$. Initial and final conditions are set for the vehicle pose and velocity with the constraints (6e,6f). Since we employ an indirect optimal control method, such initial and final conditions are strictly enforced.

Notice that, in the optimal control formulation (6a)-(6f), the final maneuver time t_f is not known a priori, while being one of the quantities we want to minimize. In order to solve the OCP, we rewrite it in its *free-time* formulation, by substituting the time t with a new independent variable, named ζ:

$$t = \zeta\, t_f, \qquad \zeta \in [0, 1], \quad (8)$$

The new independent variable ζ ranges in the fixed interval $[0, 1]$. The final maneuver time t_f is now treated as a new state variable, which is constant in time, i.e. $\frac{\mathrm{d}t_f}{\mathrm{d}t} = 0$. The extended state of the dynamical system is redefined as $x = [x, y, \theta, v_x, \delta, t_f]$. Let us indicate with h' the derivative of the generic quantity h with respect to the new independent variable ζ. Using the chain rule of differentiation, we can write:

$$x' = \frac{\mathrm{d}x}{\mathrm{d}\zeta} = \frac{\mathrm{d}x}{\mathrm{d}t}\frac{\mathrm{d}t}{\mathrm{d}\zeta} = \frac{\mathrm{d}x}{\mathrm{d}t} t_f \quad (9)$$

Exploiting the relation (9) and adding the new state variable $t_f(\zeta)$, whose derivative with respect to ζ is zero, the system (1) is rewritten as:

$$\begin{cases} x'(\zeta) = t_f(\zeta)\, v_x(\zeta) \cos\big(\theta(\zeta)\big) \\ y'(\zeta) = t_f(\zeta)\, v_x(\zeta) \sin\big(\theta(\zeta)\big) \\ \theta'(\zeta) = t_f(\zeta)\, v_x(\zeta) \tan(\delta(\zeta))/L \\ v_x'(\zeta) = t_f(\zeta)\, a_x(\zeta) \\ \tau_\delta \delta'(\zeta) = t_f(\zeta)\, \delta_{\text{dot}}(\zeta) \\ t_f'(\zeta) = 0 \end{cases} \quad (10)$$

and the original OCP (6a)-(6f) is reformulated as:

$$\min_{u \in \mathcal{U}} \; w_T t_f + \int_0^1 \ell_{\text{obst}}(\zeta)\, t_f \, \mathrm{d}\zeta \quad (11a)$$

$$\text{s.t.} \;\; (10),$$

$$|\delta(\zeta)| \le \delta_{\max}, \; |v_x(\zeta)| \le v_{x_{\max}}, \quad (11b)$$

$$|\delta_{\text{dot}}(\zeta)| \le \delta_{\text{dot}_{\max}}, \; |a_x(\zeta)| \le a_{x_{\max}}, \quad (11c)$$

$$x(0) = x_{\text{ini}}, \; y(0) = y_{\text{ini}}, \; \theta(0) = \theta_{\text{ini}}, \; v_x(0) = 0, \quad (11d)$$

$$x(1) = x_f, \; y(1) = y_f, \; \theta(1) = \theta_f, \; v_x(1) = 0 \quad (11e)$$

Note that the final time t_f is now a state variable, thus it can be explicitly minimized in the target function (11a).

The OCP is formulated and solved using the software suite PINS [33]–[37], which is based on the indirect method for optimal control. A comprehensive comparison of direct and indirect methods for the solution of optimal control problems can be found in [37]. The comparison showed that PINS has comparable performances to a state-of-the-art direct method implementation. Starting from the continuous-time OCP (11a)-(11e), PINS builds a two-point boundary value problem, which is then discretized. We use a dense time grid with 500 points for the discretization. The choice of PINS is justified because the indirect optimal control method implemented by the proprietary software and solver, developed by our research group, allow us to solve the OCPs with low computational times on the dense mesh grid. The reader can refer to [31], [38]–[41] for other recent examples of minimum-time nonlinear OCPs solved in real-time and offline using PINS.

D. TRAJECTORY GUESS GENERATION

We adopt an effective *Hybrid A* + OCP tracking* strategy to compute a solution guess for the optimal control problem. The proposed guess method combines the Hybrid A* algorithm [3] and an optimal control tracking problem. First, a kinematically feasible parking path is generated through the Hybrid A* algorithm. Second, the path provided by the Hybrid A* is tracked by solving the following optimal control problem:

$$\min_{u \in \mathcal{U}} w_T t_f + \int_0^1 \Big[w_P \Big(\big(x(\zeta) - x_A(\zeta)\big)^2 + \big(y(\zeta) - y_A(\zeta)\big)^2 \Big) + w_\theta \big(\theta(\zeta) - \theta_A(\zeta)\big)^2 \Big] t_f \, \mathrm{d}\zeta \quad (12)$$

$$\text{s.t.} \;\; (10), (11b), (11c), (11d), (11e)$$

where $\{x_A, y_A, \theta_A\}$ are the coordinates and yaw angle of the Hybrid A* path, and $\{w_T, w_P, w_\theta\}$ are tunable weights. Note that the tracking OCP (12) uses the same vehicle model (10) and constraints (11b)-(11e) as the main OCP (11a)-(11e). The solution of the tracking OCP (12) is then used as a guess for the free-trajectory main optimal control problem (11a)-(11e).

As will be shown in Section V-B, our solution guess method (Hybrid A* + OCP tracking) outperforms the Hybrid

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2023.3330431

A* alone, leading to collision-free trajectories and decreasing the overall computational times. More precisely, combining the Hybrid A* with the tracking OCP (12), we obtain a solution guess for *all* the variables of the free-trajectory OCP (11a)-(11e), while the Hybrid A* alone would provide only a sequence of positions and yaw angles.

IV. TRAJECTORY TRACKING

In this section, we present the low-level trajectory tracking controllers. Referring to Fig. 2, we implement the following low-level controllers: a feedback longitudinal controller for speed-tracking, and a feedforward + feedback steering controller for path tracking.

The main novelty introduced in this paper for trajectory tracking is the feedforward steering controller, to which we dedicate the next section. The feedback steering controller consists of two proportional regulators, which will be described in Section IV-B. Finally, the feedback longitudinal controller is a simple PI regulator, and its description will be omitted since it is well established in the literature.

A. FEEDFORWARD STEERING CONTROLLER

To track the path computed by high-level OCP, we devise a novel feedforward steering controller. The controller has a pseudo-neural physics-driven formulation (pNN), which uses few learnable parameters to accurately model the nonlinear steering dynamics. In comparison with other papers using neural networks for motion tracking of parking maneuvers [28], the specific structure devised for pNN requires smaller training datasets to produce accurate predictions. The structure of pNN is partly inspired by the neural models that we presented in [31], [42], [43], but the model of this paper is designed to capture the nonlinear steering dynamics at low speed.

Referring to Fig. 6, pNN computes in real-time the steering wheel angle δ_{ff_k}, where $k \in \mathbb{N}$ is the index of the current time step, using the following inputs:

1) a window of $q+1$ future trajectory curvature values $\boldsymbol{\rho}_k = [\rho_k, \ldots, \rho_{k+q}]$ computed by the high-level OCP;
2) a window of steering angles $[\delta_{\mathrm{ff}_{k-r}}, \ldots, \delta_{\mathrm{ff}_{k-1}}]$ computed by pNN in the past r time steps.

The trajectory curvature ρ used as input for pNN is given by Ω/v_x, with $\Omega = \frac{\mathrm{d}\theta}{\mathrm{d}t}$ and v_x being respectively the yaw rate and the forward speed computed by the high-level OCP.

pNN is a discrete-time dynamical system, with sampling time $T_{\mathrm{s_{NN}}} = 0.05$ s, formulated as:

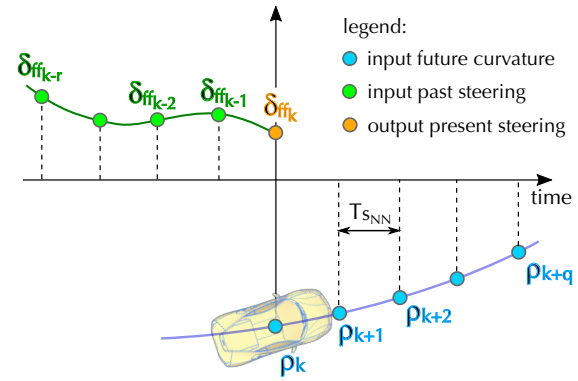


FIGURE 6. **Inputs and output of the feedforward pseudo-neural steering controller (pNN). Inputs: window of future optimal curvature values $\boldsymbol{\rho_k} = [\rho_k, \ldots, \rho_{k+q}]$ computed by the OCP, and window of past steering angles $[\delta_{\mathbf{ff}_{k-r}}, \ldots, \delta_{\mathbf{ff}_{k-1}}]$ computed by pNN. Output: feedforward steering angle $\delta_{\mathbf{ff}_k}$ for the current time step k.**

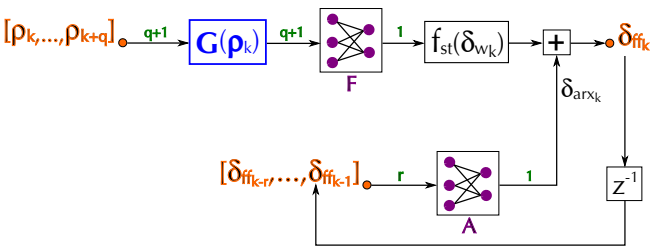


FIGURE 7. **Structure of the feedforward pseudo-neural steering controller (pNN), aimed at modeling the nonlinear steering dynamics at low speed. The quantities colored in green are the sizes of the propagated signals.**

$$\begin{aligned}\delta_{\mathrm{ff}_k} &= f_{\mathrm{st}}\left(G(\boldsymbol{\rho}_k)\cdot\mathbf{F}^T\right) + \left[\delta_{\mathrm{ff}_{k-r}}, \quad \ldots, \quad \delta_{\mathrm{ff}_{k-1}}\right]\mathbf{A}^T = \\ &= f_{\mathrm{st}}\left(\left[g(\rho_k), \quad \ldots, \quad g(\rho_{k+q})\right]\begin{bmatrix}\mathrm{F}_1\\ \vdots \\ \mathrm{F}_{q+1}\end{bmatrix}\right) + \\ &\quad + \left[\delta_{\mathrm{ff}_{k-r}}, \quad \ldots, \quad \delta_{\mathrm{ff}_{k-1}}\right]\begin{bmatrix}\mathrm{A}_1\\ \vdots \\ \mathrm{A}_r\end{bmatrix}\end{aligned} \tag{13}$$

where the notation $\mathbf{F}^T$ denotes the transpose of the vector $\mathbf{F}$. Fig. 7 graphically depicts the internal structure (13) of pNN.

Let us now analyze the role of each block in Fig. 7 and equation (13), starting from the functional layer $G(\cdot)$.

1) Functional Layer G

The vector functional layer $G(\boldsymbol{\rho}_k) = [g(\rho_k), \ldots, g(\rho_{k+q})]$ in (13) is designed to learn a steady-state relation between the trajectory curvature ρ and the steering angle δ_{w} at the front wheels[2]. The vector $G(\boldsymbol{\rho}_k)$ has $q+1$ entries, representing steering angle δ_{w} predictions, for the present and future time steps.

[2] δ_{w} is defined as the average of the steering angles at the front wheels.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2023.3330431

The function $g(\cdot)$ is the scalar version of $G(\cdot)$, and is designed to model a vehicle characteristic curve, which we call *curvature diagram*. Such a diagram is shown in Fig. 8b, with a solid blue line, and it can be obtained experimentally with a simple sinusoidal steering maneuver, like the one in Fig. 8a. The curvature diagram captures the deviations between $\tan(\delta_{\mathrm{w}})$ and the kinematic steering angle ρL, with L being the vehicle wheelbase. The shape of the diagram depends on the steering maps[3] (i.e., the nonlinear relations between the steering wheel angle δ_D and δ_{w}), on the characteristics of the tires and suspensions, and on other factors. The shape of the curvature diagram is nearly independent of the vehicle speed v_x (in the parking range $v_x \lesssim 5$ km/h).

Let us model the curvature diagram with a polynomial, as a function of the trajectory curvature ρ:

$$\tan(\delta_{\mathrm{w}}) - \rho L = h(\rho) = h_1\rho + h_2\rho^3 + h_3\rho^5 \qquad (14)$$

where $h(\rho)$ is the approximating function, shown with a green line in Fig. 8b, and $\{h_1, h_2, h_3\}$ are optimization variables. Polynomials of 5th order (with only odd powers) were found to be accurate enough to model the curvature diagram.

The function $g(\rho)$ appearing in (13) is obtained by solving (14) for the steering angle δ_{w}:

$$\delta_{\mathrm{w}} = \arctan\big(h(\rho) + \rho L\big) = g(\rho) \qquad (15)$$

The function $g(\rho)$ is a static map, and relates ρ with δ_{w} in steady-state conditions.

2) Fully-Connected Layer F

As previously described, the functional layer $G(\boldsymbol{\rho}_k)$ returns a vector of steady-state front wheels steering angle predictions, for a given vector of present and future trajectory curvatures $\boldsymbol{\rho}_k$. The entries of $G(\boldsymbol{\rho}_k)$ are then linearly combined by the fully-connected layer weights $\mathbf{F} = [\mathrm{F}_1, \ldots, \mathrm{F}_{q+1}]$, to model part of the transient dynamic response of the system, and improve the steady-state predictions. During training, the knowledge of the future curvature $[\rho_k, \ldots, \rho_{k+q}]$ is used to learn the dynamic behavior of the vehicle, and the possible delays between a steering input and the corresponding trajectory curvature output.

The scalar output of $G(\boldsymbol{\rho}_k)\cdot\mathbf{F}^T$ in (13) is a first estimate of the front wheels steering angle δ_{w_k}, at the present time step k.

3) Functional Layer f_{st}

The functional layer $f_{\mathrm{st}}(\cdot)$ in Fig. 7 receives as input a steering angle δ_{w_k} at the front wheels —computed by the preceding layers— and maps it in the corresponding steering wheel angle δ_{ff_k}. More precisely, the function $f_{\mathrm{st}}(\cdot)$ is an identified static steering map[4], i.e., a lookup table to relate δ_{w_k} and δ_{ff_k}. Note that $f_{\mathrm{st}}(\cdot)$ is only an estimate of the real steering map, which also depends on the suspension deformations. Such minor effects are learned partly by the proposed neural model, and partly compensated by the feedback steering controller.

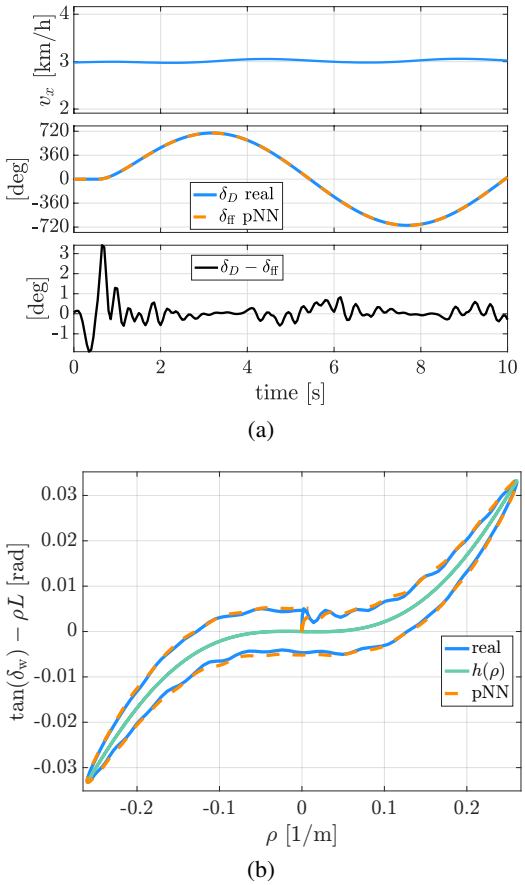


FIGURE 8. Testing the feedforward pseudo-neural steering controller (pNN) with a sine-steer maneuver at constant speed: (a) comparison of the real steering wheel angle δ_D with the prediction δ_{ff} of pNN, and (b) resulting curvature diagram.

4) Fully-Connected Layer A

Finally, in (13) we use the fully-connected layer weights $\mathbf{A} = [\mathrm{A}_1, \ldots, \mathrm{A}_r]$ to learn a linear combination of past steering angles $[\delta_{\mathrm{ff}_{k-r}}, \ldots, \delta_{\mathrm{ff}_{k-1}}]$, computed by pNN in the previous $r = 15$ time steps. The weights in $\mathbf{A}$ provide a further accuracy improvement, to model the steering dynamics. The unit delay z^{-1} in Fig. 7 shows that the output of pNN is recursively fed back and re-used, in an auto-regressive way.

The output of the controller pNN is $\delta_{\mathrm{ff}_k} = f_{\mathrm{st}}(\delta_{\mathrm{w}_k}) + \delta_{\mathrm{arx}_k}$, with $\delta_{\mathrm{arx}_k} = [\delta_{\mathrm{ff}_{k-r}}, \ldots, \delta_{\mathrm{ff}_{k-1}}]\mathbf{A}^T$.

[3]The steering maps of the vehicle simulator also depend on the deformations of the suspensions.

[4]We estimate the steering map $\delta_{\mathrm{ff}} = f_{\mathrm{st}}(\delta_{\mathrm{w}})$ with offline static tests, by identifying a lookup table relating δ_{w} with δ_{ff}. The angle δ_{w} is measured as $(\delta_{11} + \delta_{12})/2$, with $\{\delta_{11}, \delta_{12}\}$ being the front right and front left wheel steering angles. Note that $\{\delta_{11}, \delta_{12}\}$ do not need to be measured online, but only one time, offline, to estimate the steering map.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2023.3330431

TRAINING THE FEEDFORWARD STEERING CONTROLLER

We now provide some details about the training and parameterization of the feedforward pseudo-neural steering controller pNN.

Regarding the inputs of pNN, the lengths $q+1=15$ and $r=15$ of, respectively, the future curvature window and the past steering window are optimized with experiments, to cover a sufficient portion of the vehicle dynamic response. The pNN model has a total number of $3+(q+1)+r=3+15+15=33$ learnable parameters.

The model is trained with supervised learning. The vehicle simulator we want to control is used in an open-loop scheme to generate the training dataset: a single sinusoidal steering maneuver —like the one in Fig. 8a— is performed at a nearly constant vehicle speed (4 km/h), and the resulting vehicle telemetry is employed to train the pNN model. Similar maneuvers —with different sine steering frequency and vehicle speed— are carried out and used for the validation and test sets. The training loss function is the mean square error (MSE) between the steering wheel angles δ_{ff} computed by pNN and the recorded steering values of the training set. Tensorflow is adopted to formulate and train pNN, using the Keras API and the Nadam optimization method [44].

We underline that, given the specific structure designed for pNN and its low number of learnable parameters, the training dataset required to produce accurate predictions is smaller than in other more traditional neural network approaches [28].

Fig. 8a shows the results obtained with pNN on a test set, not used during training. The steering angle δ_{ff} computed by pNN is very close to the recorded angle δ_D, and the predicted curvature diagram (Fig. 8b) is almost overlapped with the real one. The root mean squared (RMS) prediction error $\delta_{\text{ff}}-\delta_D$ on the test set is 0.487 deg.

B. FEEDBACK STEERING CONTROLLER

The feedback steering controller consists of two proportional regulators. The first controller aims at decreasing the path tracking error[5], namely the Euclidean distance between the (estimated) vehicle position $[\hat{x}_k, \hat{y}_k]$ at time step k and the desired position $[x_k, y_k]$. The second controller reduces the difference between the (estimated) yaw angle $\hat{\theta}_k$ and the target value θ_k.

The feedback controllers are tuned after training the feedforward steering controller pNN. Since the internal structure and the parameters of the vehicle to be controlled are assumed to be unknown, we tune the feedback steering controllers with a model-free approach. Similarly to what we did in [31], the M2 iterative learning tuning (ILT) method of [45] is adopted, due to its ability to deal with custom cost functions and generic plant models. Using the ILT, the proportional gain of a feedback controller is iteratively updated using an approximate gradient method, where the gradient of the cost function is estimated from the results of the previous and current iterations.

The path tracking feedback controller is tuned first, while the yaw angle tracking feedback controller is tuned in a second phase. The cost function of the ILT is a weighted sum of the mean squared values of the tracking error, the control magnitude δ_{fb}, and its variations $\Delta\delta_{\text{fb}}$. In each learning iteration, the vehicle simulator is controlled in closed-loop, to track a reference parking maneuver computed with optimal control. In this way, new data are collected, the proportional controller gain is updated, and a new iteration is performed, until certain exit conditions are met.

V. RESULTS

A. TEST SCENARIOS

The presented autonomous parking framework is tested in four challenging scenarios. To evaluate the robustness of the trajectory planning and control algorithms, a grid of starting positions is used for each scenario:

1) Reverse parking scenario, Fig. 9a: the maneuvers are solved on a grid of 28 starting positions.
2) Parallel parking scenario, Fig. 9b: the maneuvers are solved on a grid of 21 starting positions. A nearby obstacle limits the maneuvering space on the right side of the vehicle.
3) Generic parking scenario, Fig. 9c: the maneuvers are solved on a grid of 14 starting positions. The vehicle must execute a reverse parking maneuver: the parking spot is quite far and an obstacle is interposed between the starting and final positions.
4) Angle parking scenario, Fig. 9d: the maneuvers are solved on a grid of 12 starting positions. The vehicle must reverse into an angle parking spot. The maneuvering space on the right and on the left of the vehicle is limited by a combination of obstacles. The complex obstacles in this scenario are created by rotating and superimposing several rectangular obstacles.

Overall, we use $28+21+14+12=75$ parking maneuvers to test our framework.

Each side of the grey obstacles in Fig. 9 has a safety margin (0.10 m) with respect to the true obstacle dimensions. We underline that the safety margin is imposed only to consider potential errors deriving from the obstacle size estimation, localization system and path tracking controllers. As will be shown next, no safety margin would be required for the parking maneuvers planned with optimal control, since in the test scenarios presented in this work such maneuvers never lead to obstacle collisions.

B. TRAJECTORY PLANNING SOLUTIONS WITH OPTIMAL CONTROL

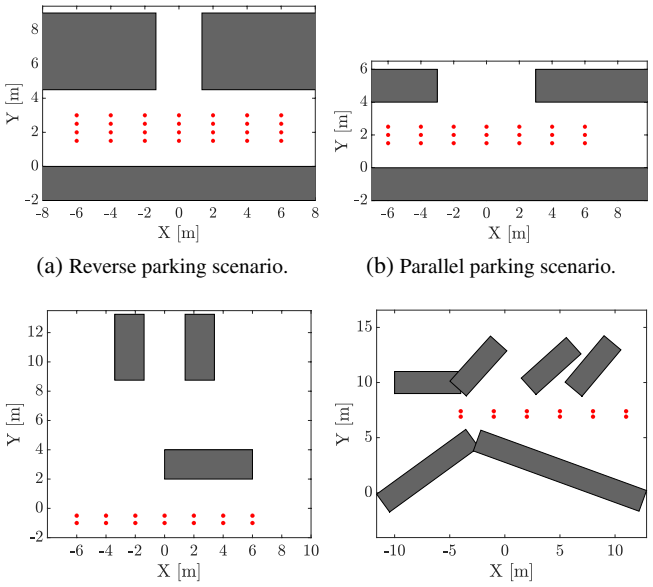
In this section, we show the trajectories planned with optimal control, for the four test scenarios previously described.

Fig. 10, 11, 12 and 13 depict some of the planned maneuvers, respectively for the reverse parking, parallel

[5]A low-pass filter is used to eliminate the high-frequency content of the path tracking error, to avoid rapid steering variations.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

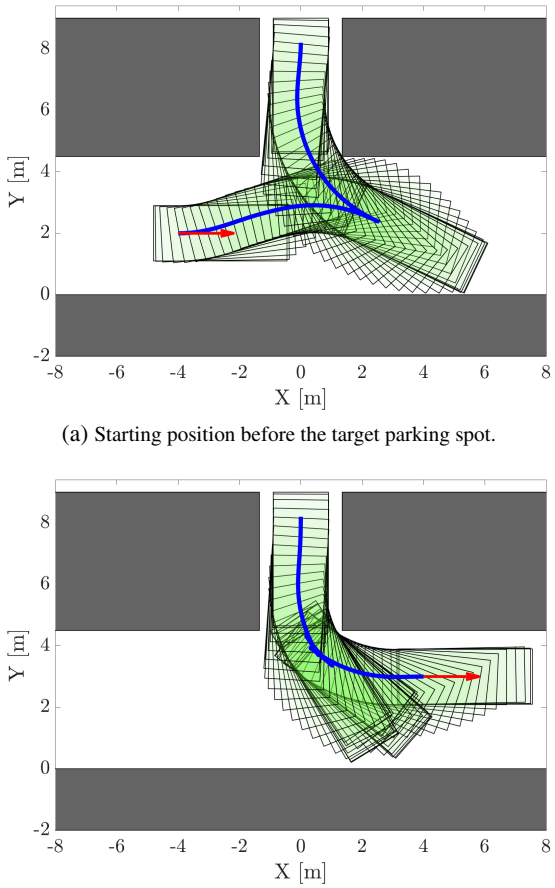
This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2023.3330431



(a) Reverse parking scenario. (b) Parallel parking scenario.

(c) Generic parking scenario. (d) Angle parking scenario.

FIGURE 9. Grid of starting positions (red dots) used to test the automated parking framework, for each of the four parking scenarios. The total number of parking maneuvers is 28 + 21 + 14 + 12 = 75.



(a) Starting position before the target parking spot.

(b) Starting position after the target parking spot.

FIGURE 10. Reverse parking maneuvers, plotted for 2 of the 28 starting positions in Fig. 9a. The blue line represents the path of the vehicle reference point (center of the rear axle). The red arrow indicates the vehicle's forward direction at the starting point. Frames of the vehicle motion are shown every 10 discretization grid points of the OCP solution.

parking, generic parking and angle parking scenarios. For the sake of brevity, in each scenario, the parking maneuvers are plotted for only two of the many starting positions of Fig. 9. In each plot of Fig. 10, 11, 12 and 13, we show in green a sequence of 50 frames of the vehicle motion. The discretization time grid in the OCP contains 500 points, and it is therefore 10 times denser than the plotted sequence of frames.

The presented framework is able to plan complex maneuvers, composed of several segments with different travel directions. For example, the angle parking maneuvers in Fig. 13a and 13b have 4 segments, with the travel direction sequence forward-rearward-forward-rearward, while the reverse parking maneuver in Fig. 10b consists of 5 segments.

No collisions occur with the obstacles, in all the 75 testing scenarios. Such an accurate collision-free trajectory optimization is made possible by the dense mesh with which the OCPs are discretized, and by the relatively large number of collision-checking points. In contrast, the authors of [17] obtained simpler parking maneuvers, always composed of 2 segments, and their trajectories sometimes had local obstacle collisions between adjacent mesh points, due to a quite coarse discretization.

The OCPs are solved on a 2019 MacBook Pro, equipped with a 2,6 GHz 6-Core Intel Core i7 processor. Table 2 reports the computational (CPU) times to calculate the solution guesses (Section III-D) and solve the OCPs. On average, the CPU time to plan the parking maneuvers is around 1.5-2 s, and it is below 4.5 s for all the 75 tested maneuvers.

The main factors and novelties that contribute to decreasing the CPU times are:

- The formulation of the obstacle smooth penalty functions (Section III-B);
- The effectiveness of the solution guess strategy (Section III-D);
- The effectiveness of the PINS solver (Section III-C).

Table 3 compares the CPU times obtained with our *Hybrid A* + OCP tracking* guess strategy (Section III-D), against the Hybrid A* alone. The latter is used as a solution guess for optimal control by other authors, like [17]. However, the Hybrid A* returns only a feasible (often non-time-optimal) path, without speed and steering angle profiles. Conversely, our guess strategy uses the optimal control problem (12) to track the Hybrid A* path, and the resulting solution is then used as a guess for all the states and controls of the main OCP (11a)-(11e). Table 3 compares the mean CPU times obtained with our guess strategy and with the Hybrid A*, using the 75 parking maneuvers of the 4 scenarios (Fig. 9). Our guess method decreases by more than 20% the mean CPU times for trajectory planning (1.80 s against 2.30 s).

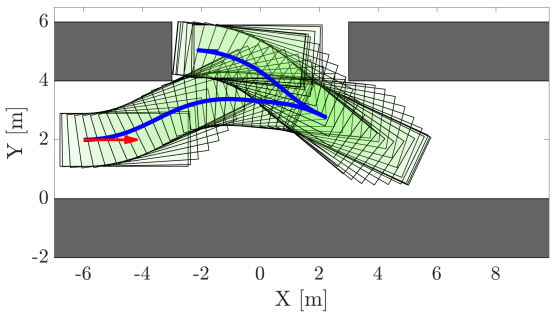
Assuming an automotive-grade hardware 5 times slower than the hardware employed in this paper, the scaled-up computational times to plan the parking maneuvers would

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

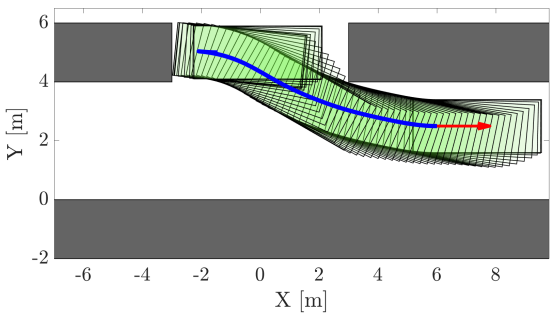
This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2023.3330431

	Reverse Parking	Parallel Parking	Generic Parking Scenario	Angle Parking
mean CPU time solve **guess**	1.01 s	0.61 s	0.62 s	1.04 s
max CPU time solve **guess**	2.04 s	0.79 s	1.15 s	2.03 s
mean CPU time solve **OCP**	1.10 s	1.08 s	0.54 s	1.03 s
max CPU time solve **OCP**	3.01 s	2.95 s	0.86 s	2.51 s
mean CPU time solve **guess + OCP**	**2.11** s	**1.69** s	**1.16** s	**2.07** s
max CPU time solve **guess + OCP**	**4.45** s	**3.57** s	**1.82** s	**4.23** s

TABLE 2. **CPU times of the optimal control problems on the four parking scenarios, using the grid of starting positions of Fig. 9.**



(a) Starting position before the target parking spot.



(b) Starting position after the target parking spot.

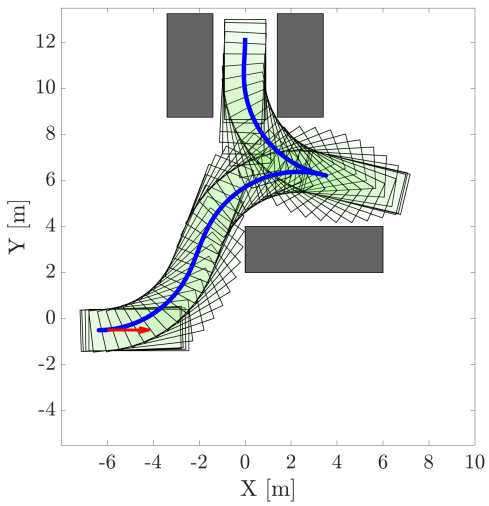
FIGURE 11. **Parallel parking maneuvers, plotted for 2 of the 21 starting positions in Fig. 9b. The blue line represents the path of the vehicle reference point (center of the rear axle). The red arrow indicates the vehicle's forward direction at the starting point. Frames of the vehicle motion are shown every 10 discretization grid points of the OCP solution.**

still be acceptable for a commercial application.

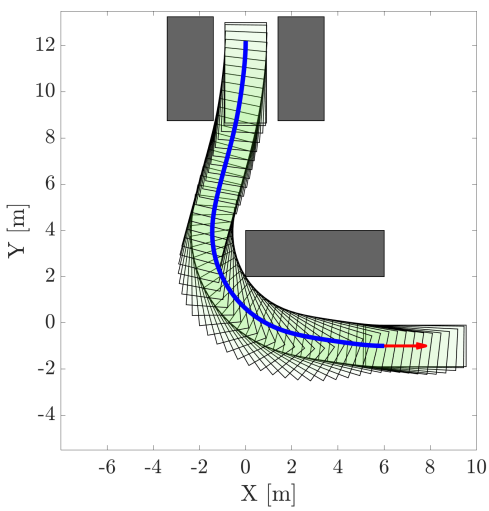
C. TRAJECTORY TRACKING WITH THE VEHICLE SIMULATOR

Following the scheme in Fig. 2, the parking trajectories planned with optimal control are tracked by low-level controllers (Section IV), to drive the high-fidelity vehicle simulator described in Section II-A. The trajectory tracking framework is implemented in a Matlab&Simulink environment.

A video demonstration of the planned and tracked parking maneuvers is available as supplementary material. The video was generated with the CARLA graphic environment [46].



(a) Starting position before the obstacle.

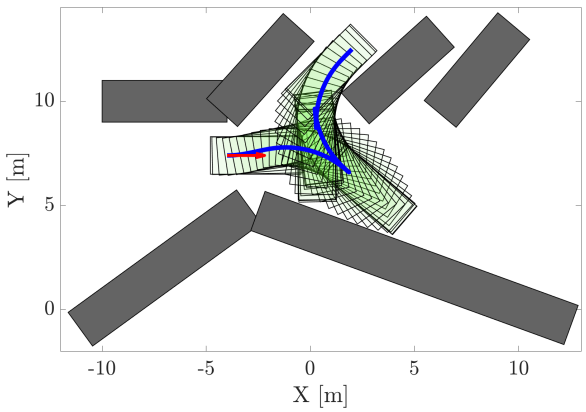


(b) Starting position after the obstacle.

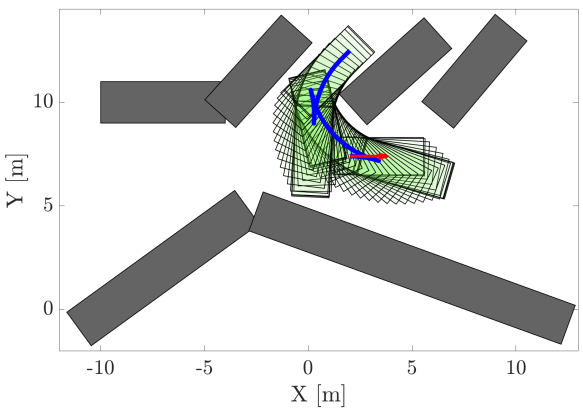
FIGURE 12. **Generic parking scenario, plotted for 2 of the 14 starting positions in Fig. 9c. The blue line represents the path of the vehicle reference point (center of the rear axle). The red arrow indicates the vehicle's forward direction at the starting point. Frames of the vehicle motion are shown every 10 discretization grid points of the OCP solution.**

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2023.3330431



(a) Starting position before the target parking spot.



(b) Starting position after the target parking spot.

FIGURE 13. Angle parking maneuvers, plotted for 2 of the 12 starting positions in Fig. 9d. The blue line represents the path of the vehicle reference point (center of the rear axle). The red arrow indicates the vehicle's forward direction at the starting point. Frames of the vehicle motion are shown every 10 discretization grid points of the OCP solution.

	Hybrid A*	Hybrid A* + OCP tracking
mean CPU time solve **guess**	0.50 s	0.83 s
mean CPU time solve **guess + OCP**	2.30 s	**1.80** s

TABLE 3. Comparison of the proposed solution guess method, named *Hybrid A* + OCP tracking*, against the benchmark *Hybrid A alone. The table compares the mean CPU times obtained on the four scenarios of Fig. 9 (75 parking maneuvers). Our method provides an improved guess to the OCP (11a)-(11e) for trajectory planning, whose overall mean computational time is decreased by more than 20% (1.80 s against 2.30 s).**

	Pure pursuit	Clothoid-based	pNN + fb
RMS(path track err)	0.179 m	0.015 m	**0.006** m
max(path track err)	0.815 m	0.164 m	**0.048** m
RMS($\lvert\theta - \hat{\theta}\rvert$)	6.48 deg	1.10 deg	**0.19** deg
max($\lvert\theta - \hat{\theta}\rvert$)	20.52 deg	5.02 deg	**0.74** deg

TABLE 4. Tracking performance on the four parking scenarios (75 parking maneuvers, whose starting conditions are shown in Fig. 9), with different low-level steering controllers. The proposed pNN+fb steering controller outperforms the literature benchmarks, in all the path and yaw angle (θ) tracking error indicators. Considering the 0.10 m safety margin on the obstacle bounding boxes, pNN+fb is the only controller that never leads to collisions.

However, we remark that we do *not* close the loop on the simple vehicle model of the CARLA simulator; on the contrary, we employ our high-fidelity vehicle simulator, illustrated in Section II-A.

Literature Benchmarks for Steering Control

The performance of the proposed feedforward-feedback pseudo-neural steering controller (Sections IV-A and IV-B) is compared with other two literature lateral controllers, namely the clothoid-based technique of [21] and the pure pursuit controller [22].

For the clothoid-based method [21], at each time step a clothoid curve is built, with the library developed by [47], to connect the current vehicle pose with a point on the target path, using a look-ahead distance of 2 m. The steering wheel angle computed by the clothoid-based controller at the time step k is based on a kinematic relation:

$$\delta_{D_k} = f_{\text{st}}\Big(\arctan(\rho_{c_k} L)\Big) \tag{16}$$

where ρ_{c_k} is the curvature value of the clothoid at its starting point, L is the vehicle wheelbase, and $f_{\text{st}}(\cdot)$ is the vehicle (identified) steering map, described in Section IV-A.

For the pure pursuit controller, the look-ahead distance is tuned with experiments and set to 2.5 m.

Analysis of the Trajectory Tracking Results

Table 4 compares the path and yaw angle tracking performance of the proposed feedforward-feedback steering controller (here named pNN+fb) with the clothoid-based and pure pursuit literature techniques. The comparison focuses on the root mean square (RMS) tracking errors and their maximum values, computed using the four parking scenarios and the 75 different starting positions shown in Fig. 9. The path tracking error (PTE) is computed as $\sqrt{(x_k - \hat{x}_k)^2 + (y_k - \hat{y}_k)^2}$, with $[\hat{x}_k, \hat{y}_k]$ being the real vehicle coordinates at time step k, and $[x_k, y_k]$ being the coordinates of the closest point on the optimal path. Similarly, the yaw angle tracking error is $\theta_k - \hat{\theta}_k$.

The proposed pNN+fb steering controller outperforms the literature benchmarks in all the path and yaw angle tracking error indicators. Considering that the maximum path tracking and yaw angle tracking errors obtained with the pNN+fb

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2023.3330431

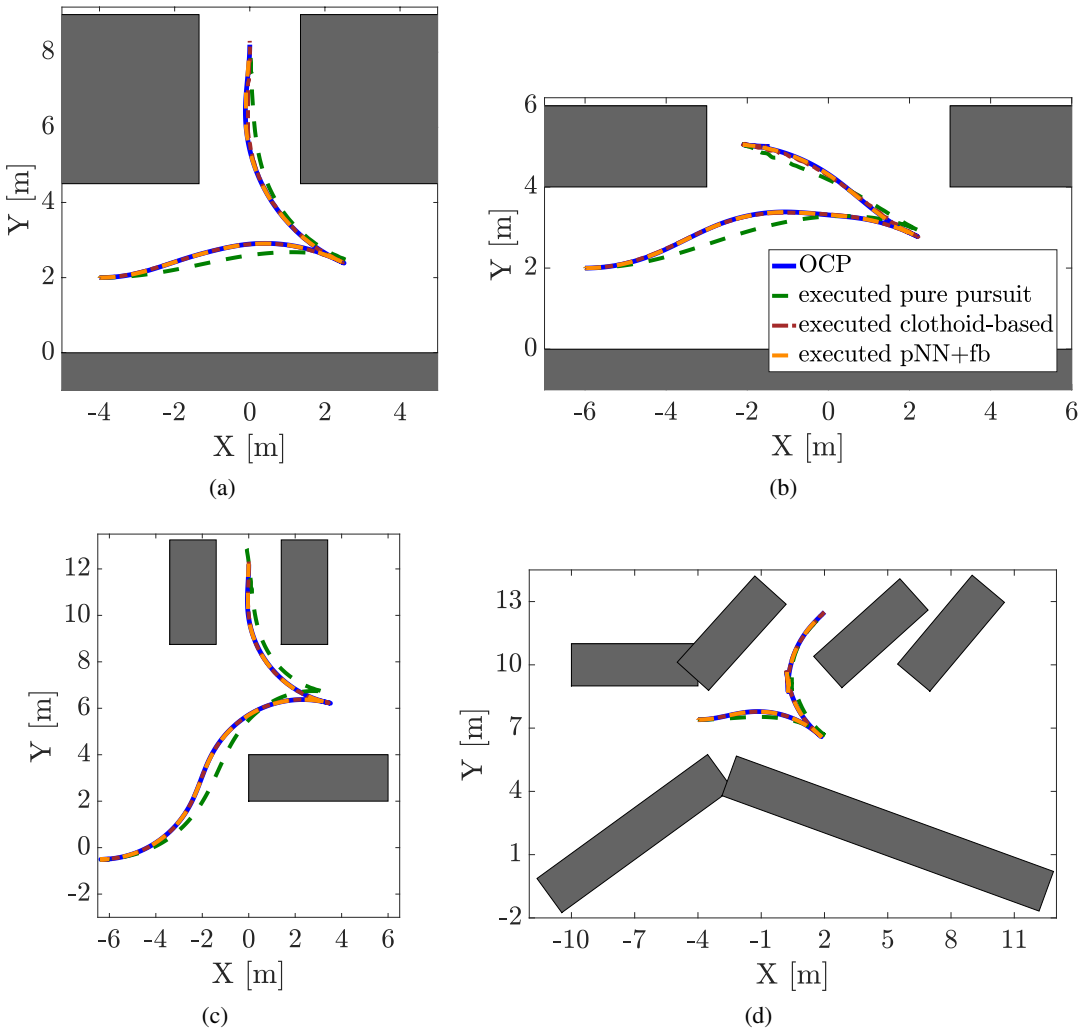


FIGURE 14. Comparison of the path tracking performance of the proposed feedforward-feedback pseudo-neural steering controller (pNN+fb) with the clothoid-based and pure pursuit literature techniques. The figure plots 4 of the 75 parking maneuvers considered in this paper: (a) reverse parking, (b) parallel parking, (c) generic parking scenario, (d) angle parking. The plots show the paths of the origin of the vehicle's reference frame, namely the center of the vehicle's rear axle.

controller are 0.048 m and 0.74 deg respectively, and that the obstacle bounding boxes are defined with a safety margin of 0.10 m (Section V-A), then no collisions due to errors in the tracking of the planned optimal path occur when the pNN+fb is used.

Fig. 14 shows some of the path-tracking results, in the four parking scenarios. Fig. 15 plots the forward speed v_x and steering wheel angle δ_D profiles, together with the path and yaw angle tracking errors, for the parking maneuver of Fig. 14a. Note that the pNN+fb controller delivers a smooth steering angle δ_D. Conversely, the clothoid-based and pure pursuit controllers sometimes yield oscillating or saturated steering angles, which result in worse tracking accuracy and decrease passenger comfort. Finally, the feedback longitudinal controller provides adequate speed-tracking performance.

The obtained results show that the presented framework is suitable for a real application, where planning and executing accurate maneuvers is fundamental, and the computational times should be reasonably low.

D. ROBUSTNESS ANALYSES

1) Robustness to Vehicle and Adherence Parameters

The framework's robustness is here analyzed by increasing the vehicle mass and reducing the road adherence, without changing the parameters of the trajectory planning and tracking controllers. Specifically, the vehicle mass is increased by 220 kg, and a wet road is simulated, decreasing by 40% the cornering and longitudinal tire stiffnesses [29], as well as the peak of the lateral and longitudinal tire forces. The changed vehicle and adherence parameters affect the curvature diagram (Fig. 8b) and the lateral dynamics, thus potentially decreasing the accuracy of the feedforward steering controller pNN. Nonetheless, the feedback steering controller compensates for part of the unmodeled dynamics, preserving good tracking accuracy. As reported in Table 5,

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2023.3330431

	Reverse Parking	Parallel Parking	Generic Parking Scenario	Angle Parking
RMS(path track err)	0.007 m	0.006 m	0.008 m	0.007 m
max(path track err)	0.051 m	0.043 m	0.038 m	0.055 m
RMS($\lvert\theta - \hat{\theta}\rvert$)	0.20 deg	0.23 deg	0.15 deg	0.22 deg
max($\lvert\theta - \hat{\theta}\rvert$)	0.65 deg	0.78 deg	0.51 deg	0.75 deg

TABLE 5. Analyzing the effect of an unknown increase in the vehicle mass (by 220 kg) and road adherence reduction (by 40%) on the trajectory tracking performance. The table reports the path and yaw angle tracking errors on the four parking scenarios (75 parking maneuvers), obtained with the proposed pNN+fb steering controller. The results do not degrade with respect to the nominal case (Table 4), thus indicating good robustness of the automated parking framework to variations of the tyre-road adherence and/or vehicle physical properties.

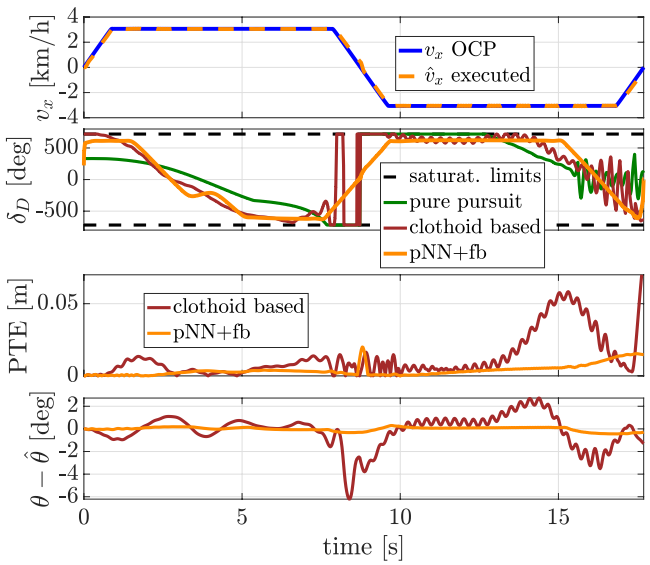


FIGURE 15. Trajectory tracking in the reverse parking scenario of Fig. 14a: speed (v_x) and steering angle (δ_D) profiles, path tracking error (PTE) and yaw angle tracking error. The plots of the path and yaw tracking errors show only the two best controllers, excluding the pure pursuit. The proposed pNN+fb controller delivers a smooth steering angle δ_D, while the clothoid-based and pure pursuit controllers sometimes yield oscillating or saturated steering angles, which result in worse tracking accuracy and decrease passenger comfort.

the path and yaw angle tracking results do not degrade with respect to the nominal case (Table 4), for the four parking scenarios and the 75 starting positions, which indicates good robustness of the automated parking framework to variations of the tyre-road adherence and/or vehicle physical properties.

2) Robustness to Sensor Noise

In this section, realistic sensor models are integrated into a state estimation setup, to evaluate the robustness of the planning and control framework to measurement noise. We remark that proposing an effective choice of sensors and localization techniques for autonomous parking is not in the scope of this paper. The state estimation methods in this section are used only to evaluate the results in the presence of measurement noise.

We define our parking scenarios using the *Automated Driving Toolbox* in Matlab & Simulink R2022b. Exploiting the Toolbox, synthetic point clouds are generated from a lidar sensor mounted on the ego vehicle, using a realistic statistical model of the sensor detections. Simultaneous localization and mapping (SLAM) is performed with the Toolbox, using the lidar scans. The SLAM algorithm returns an estimate of the ego vehicle pose $[\hat{x}, \hat{y}, \hat{\theta}]$. In addition, an inertial measurement unit (IMU) model is used within the *imuSensor* function in Matlab, to measure the longitudinal and lateral accelerations $\hat{a}_x$, $\hat{a}_y$ and the yaw rate $\hat{\Omega} = \dot{\hat{\theta}}$. Finally, wheel encoder models are adopted to measure the angular rates of the four wheels, from which an estimate of the vehicle forward speed $\hat{v}_x$ is retrieved using the estimation algorithm described in [48, Chapter 5]. An extended Kalman filter (EKF) is implemented to fuse the available measurements, using a kinematic prediction model.

Fig. 16 shows the results obtained in the reverse parking scenario, with the state estimation setup previously described. The SLAM algorithm returns an estimate of the vehicle pose every 0.2 s (green dots in Fig. 16a), while the sampling time of the IMU and wheel encoders is 0.01 s. The EKF estimates the vehicle state (pose, velocities and accelerations) every 1 ms, and the estimates are fed to the tracking controllers. The RMS value of the path tracking error (PTE, Fig. 16b) between the desired and executed paths is 0.029 m, while its maximum value is 0.089 m. Considering the 0.10 m safety margin used to define the obstacle bounding boxes (Section V-A), no collisions occur throughout the maneuver execution. The estimates of forward speed v_x and yaw rate Ω provided by the EKF are sufficiently close to the actual values (Fig. 16b), which results in a good tracking of the desired speed. The feedback steering controller computes steering corrections δ_{fb} (Fig. 16b, dark-blue line of the bottom plot) to compensate for part of the path and yaw angle estimated tracking errors. Note that the overall steering wheel angle $\delta_{\mathrm{ff}} + \delta_{\mathrm{fb}}$ (orange line) remains sufficiently smooth, which is important for passengers' comfort. A rigorous stability analysis for the feedback steering controller is hindered by the nonlinear and black-box nature of the vehicle simulator, and by the presence of a nonlinear feedforward steering controller. Nonetheless, we remark that the feedback steering controller adds minor control corrections δ_{fb} (Fig. 16b, bottom plot), since most of the control action is provided by the feedforward steering controller pNN. The feedback steering controller considerably reduces the path and yaw angle tracking errors, in all the tested parking scenarios, even in the presence of unknown changes in the vehicle

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2023.3330431

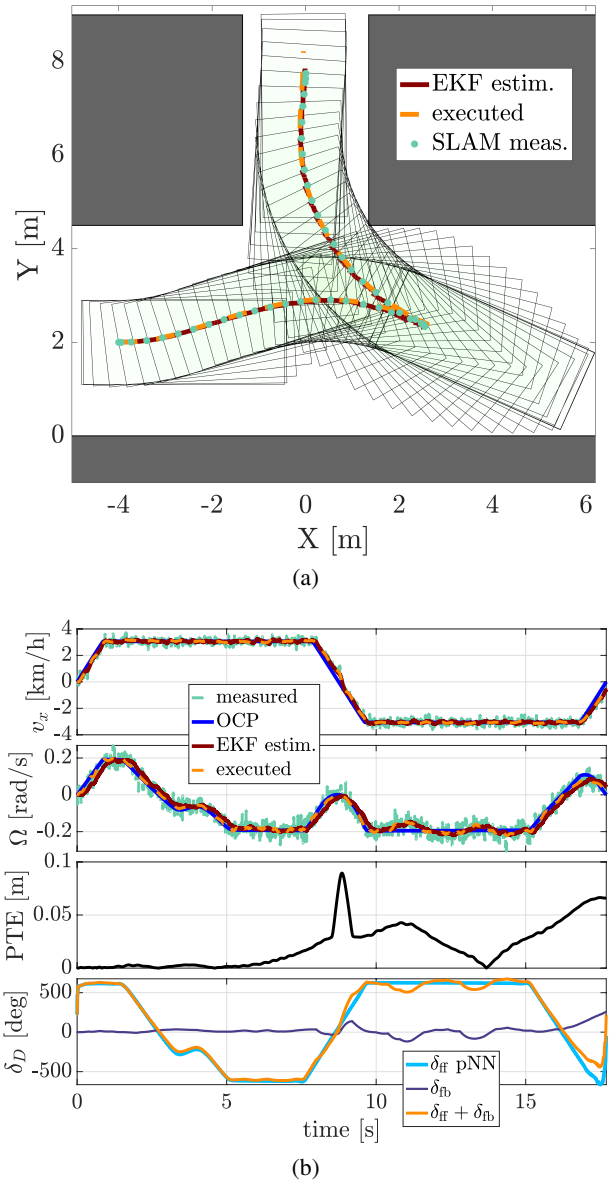


FIGURE 16. Executing a parking maneuver in the reverse parking scenario, in the presence of measurement noise: (a) comparison of the executed path with the raw SLAM results and the EKF estimates, and (b) forward speed (v_x), yaw rate, path tracking error (PTE) and steering wheel angle (δ_D) profiles.

and adherence parameters. Thus, from a practical viewpoint, the feedback action improves the closed-loop tracking performance, and helps compensate for model mismatches and disturbances.

VI. CONCLUSIONS

In this paper, we presented a framework for fast planning and execution of complex autonomous parking maneuvers. Using an indirect optimal control approach, a minimum-time parking problem was formulated, and a novel smooth penalty function was adopted for collision avoidance with optimal control. We proposed an effective warm-start strategy for the optimal control problems, based on the creation of robust guess solutions by means of an optimal tracking of the path given by the Hybrid A* algorithm. A dense mesh was used to discretize the domain of the OCPs for trajectory planning, which yields accurate collision-free maneuvers.

We introduced a novel pseudo-neural feedforward-feedback steering controller, which, in comparison with other literature techniques, yields better accuracy and smoother steering angle profiles. Using the proposed steering controller and a feedback longitudinal velocity regulator, the planned maneuvers were executed in real-time with a 14-degree-of-freedom vehicle simulator.

The framework was validated using narrow reverse, parallel and angle parking spots, and more generic parking scenarios. The results show that our framework can plan and execute a wide range of parking maneuvers in different challenging scenarios, moving close to the obstacles and avoiding collisions. In addition, our optimal control formulation and solution method can compute maneuvers with multiple segments of forward and reverse driving. The framework is sufficiently robust to a change in the vehicle mass and in the tire-ground adherence. Moreover, collision-free maneuvers were executed in the presence of synthetic measurement noise, generated with realistic sensor models. Finally, the computational times are sufficiently low, and the framework is promising for a real-world automotive application.

Future work will be focused on testing the framework with a modified commercial vehicle, to implement a valet and autonomous parking application.

ACKNOWLEDGMENT

The authors would like to thank Alice Plebe for her help with the preparation of the final video, and the critical reviews of the manuscript. The authors are also grateful to Matteo Larcher, for the development of the vehicle simulator.

REFERENCES

[1] P. Ramaswamy, "Iot smart parking system for reducing green house gas emission," in *2016 international conference on recent trends in information technology (ICRTIT)*. IEEE, 2016, pp. 1–6.
[2] B. Li, L. Fan, Y. Ouyang, S. Tang, X. Wang, D. Cao, and F.-Y. Wang, "Online competition of trajectory planning for automated parking: Benchmarks, achievements, learned lessons, and future perspectives," *IEEE Transactions on Intelligent Vehicles*, pp. 1–6, 2022.
[3] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Path planning for autonomous vehicles in unknown semi-structured environments," *The International Journal of Robotics Research*, vol. 29, no. 5, pp. 485–501, 2010.
[4] J. Petereit, T. Emter, C. W. Frey, T. Kopfstedt, and A. Beutel, "Application of hybrid a* to an autonomous mobile robot for path planning in unstructured outdoor environments," in *ROBOTIK 2012; 7th German Conference on Robotics*, 2012, pp. 1–6.
[5] S. Karaman and E. Frazzoli, "Optimal kinodynamic motion planning using incremental sampling-based methods," in *49th IEEE Conference on Decision and Control (CDC)*, 2010, pp. 7681–7687.
[6] Y. Dong, Y. Zhong, and J. Hong, "Knowledge-biased sampling-based path planning for automated vehicles parking," *IEEE Access*, vol. 8, pp. 156 818–156 827, 2020.
[7] B. Li, L. Li, T. Acarman, Z. Shao, and M. Yue, "Optimization-based maneuver planning for a tractor-trailer vehicle in a curvy tunnel: A weak reliance on sampling and search," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 706–713, 2022.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2023.3330431

[8] B. Li and Z. Shao, "A unified motion planning method for parking an autonomous vehicle in the presence of irregularly placed obstacles," *Knowledge-Based Systems*, vol. 86, pp. 11–20, 2015.

[9] B. Li, K. Wang, and Z. Shao, "Time-optimal maneuver planning in automatic parallel parking using a simultaneous dynamic optimization approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 11, pp. 3263–3274, 2016.

[10] C. Chen, B. Wu, L. Xuan, J. Chen, T. Wang, and L. Qian, "A trajectory planning method for autonomous valet parking via solving an optimal control problem," *Sensors*, vol. 20, no. 22, p. 6435, 2020.

[11] W. Sheng, B. Li, and X. Zhong, "Autonomous parking trajectory planning with tiny passages: A combination of multistage hybrid a-star algorithm and numerical optimal control," *IEEE Access*, vol. 9, pp. 102 801–102 810, 2021.

[12] J. Lian, W. Ren, D. Yang, L. Li, and F. Yu, "Trajectory planning for autonomous valet parking in narrow environments with enhanced hybrid a* search and nonlinear optimization," *IEEE Transactions on Intelligent Vehicles*, pp. 1–12, 2023.

[13] B. Li, T. Acarman, Y. Zhang, Y. Ouyang, C. Yaman, Q. Kong, X. Zhong, and X. Peng, "Optimization-based trajectory planning for autonomous parking with irregularly placed obstacles: A lightweight iterative framework," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[14] S. Song, H. Chen, H. Sun, M. Liu, and T. Xia, "Time-optimized online planning for parallel parking with nonlinear optimization and improved monte carlo tree search," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2226–2233, 2022.

[15] J. Zhang, H. Chen, S. Song, and F. Hu, "Reinforcement learning-based motion planning for automatic parking system," *IEEE Access*, vol. 8, pp. 154 485–154 501, 2020.

[16] S. Song, H. Chen, H. Sun, and M. Liu, "Data efficient reinforcement learning for integrated lateral planning and control in automated parking system," *Sensors*, vol. 20, no. 24, 2020.

[17] X. Zhang, A. Liniger, and F. Borrelli, "Optimization-based collision avoidance," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 3, pp. 972–983, 2020.

[18] 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.

[19] N. Amer, H. Zamzuri, K. Hudha, and Z. Kadir, "Modelling and control strategies in path tracking control for autonomous ground vehicles: A review of state of the art and challenges," *Journal of Intelligent & Robotic Systems*, vol. 86, pp. 1–30, 05 2017.

[20] Y. Kebbati, N. Ait-Oufroukh, D. Ichalal, and V. Vigneron, "Lateral control for autonomous wheeled vehicles: A technical review," *Asian Journal of Control*, vol. n/a, no. n/a.

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

[22] V. Sukhil and M. Behl, "Adaptive lookahead pure-pursuit for autonomous racing," *CoRR*, vol. abs/2111.08873, 2021. [Online]. Available: https://arxiv.org/abs/2111.08873

[23] Z. Zhang, L. Xie, and H. Su, "Trajectory tracking control for autonomous parking using reduced-horizon model predictive control," in *2020 4th CAA International Conference on Vehicular Control and Intelligence (CVCI)*, 2020, pp. 401–405.

[24] J. Zhang, Z. Shi, X. Yang, and J. Zhao, "Trajectory planning and tracking control for autonomous parallel parking of a non-holonomic vehicle," *Measurement and Control*, vol. 53, no. 9-10, pp. 1800–1816, 2020.

[25] B. S. Park, "Prescribed performance function based control for trajectory tracking of nonholonomic mobile robots with collision avoidance," in *2017 IEEE Conference on Control Technology and Applications (CCTA)*, 2017, pp. 1013–1018.

[26] S. He, S.-L. Dai, Z. Zhao, T. Zou, and Y. Ma, "Ude-based distributed formation control for msvs with collision avoidance and connectivity preservation," *IEEE Transactions on Industrial Informatics*, pp. 1–12, 2023.

[27] P. Zips, M. Böck, and A. Kugi, "Fast optimization based motion planning and path-tracking control for car parking," *IFAC Proceedings Volumes*, vol. 46, no. 23, pp. 86–91, 2013, 9th IFAC Symposium on Nonlinear Control Systems.

[28] J. Moon, I. Bae, and S. Kim, "An inverse vehicle model for a neural-network-based integrated lateral and longitudinal automatic parking controller," *Electronics*, vol. 8, no. 12, 2019.

[29] H. Pacejka, *Tire and Vehicle Dynamics Ed. 3*. Elsevier Science, 2012.

[30] D. Stocco and E. Bertolazzi, "Acme: A small 3d geometry library," *SoftwareX*, vol. 16, p. 100845, 2021.

[31] M. Piccinini, S. Taddei, M. Larcher, M. Piazza, and F. Biral, "A physics-driven artificial agent for online time-optimal vehicle motion planning and control," *IEEE Access*, vol. 11, pp. 46 344–46 372, 2023.

[32] D. Stocco, M. Larcher, and E. Bertolazzi, "A novel approach for real-time tire/ground contact modeling," submitted.

[33] E. Bertolazzi, F. Biral, and M. Da Lio, "Symbolic–numeric indirect method for solving optimal control problems for large multibody systems," *Multibody System Dynamics*, vol. 13, no. 2, pp. 233–252, 2005. [Online]. Available: https://doi.org/10.1007/s11044-005-3987-4

[34] E. Bertolazzi, F. Biral, and M. Da Lio, "Symbolic-numeric efficient solution of optimal control problems for multibody systems," *Journal of Computational and Applied Mathematics*, vol. 185, no. 2, pp. 404–421, 2006, special Issue: International Workshop on the Technological Aspects of Mathematics. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377042705001238

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

[36] 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.

[37] N. D. Bianco, E. Bertolazzi, F. Biral, and M. Massaro, "Comparison of direct and indirect methods for minimum lap time optimal control problems," *Vehicle System Dynamics*, vol. 57, no. 5, pp. 665–696, 2019. [Online]. Available: https://doi.org/10.1080/00423114.2018.1480048

[38] M. Piccinini, M. Larcher, E. Pagot, D. Piscini, L. Pasquato, and F. Biral, "A predictive neural hierarchical framework for on-line time-optimal motion planning and control of black-box vehicle models," *Vehicle System Dynamics*, vol. 61, no. 1, pp. 83–110, 2023.

[39] E. Pagot, M. Piccinini, and F. Biral, "Real-time optimal control of an autonomous rc car with minimum-time maneuvers and a novel kineto-dynamical model," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 2390–2396.

[40] R. Lot and N. D. Bianco, "Lap time optimisation of a racing go-kart," *Vehicle System Dynamics*, vol. 54, no. 2, pp. 210–230, 2016.

[41] N. D. Bianco, R. Lot, and M. Gadola, "Minimum time optimal control simulation of a gp2 race car," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 232, no. 9, pp. 1180–1195, 2018.

[42] M. Da Lio, R. Donà, G. P. R. Papini, F. Biral, and H. Svensson, "A mental simulation approach for learning neural-network predictive control (in self-driving cars)," *IEEE Access*, vol. 8, pp. 192 041–192 064, 2020.

[43] M. D. Lio, M. Piccinini, and F. Biral, "Robust and sample-efficient estimation of vehicle lateral velocity using neural networks with explainable structure informed by kinematic principles," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–15, 2023.

[44] T. Dozat, "Incorporating nesterov momentum into adam," 2016.

[45] J.-X. Xu and D. Huang, "Optimal tuning of pid parameters using iterative learning approach," in *2007 IEEE 22nd International Symposium on Intelligent Control*, 2007, pp. 226–231.

[46] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

[47] E. Bertolazzi and M. Frego, "Interpolating clothoid splines with curvature continuity," *Mathematical Methods in the Applied Sciences*, vol. 41, no. 4, pp. 1723–1737, 2018. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/mma.4700

[48] S. Savaresi and M. Tanelli, *Active Braking Control Systems Design for Vehicles*, 01 2011.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2023.3330431



EDOARDO PAGOT received the M.S. degree in mechatronics engineering in 2018 and the Ph.D. degree cum laude in mechatronics engineering in 2023, both from the University of Trento, Italy. He is currently a research fellow within the Department of Industrial Engineering at the University of Trento.

His research interests include multi-body dynamics modeling and optimal control (online and offline) in the field of vehicle dynamics, for both intelligent transportation and Motorsport sectors.



MATTIA PICCININI received a B.Sc. in industrial engineering and an M.Sc. in mechatronics engineering, both cum laude and both from the University of Trento, Italy, respectively in 2017 and 2019. He is currently a Ph.D. student at the University of Trento, Italy. From March to June 2022, he was a visiting Ph.D. student at the Universität der Bundeswehr, Munich, Germany.

His research focuses on motion planning, control and state estimation methods for autonomous vehicles.



ENRICO BERTOLAZZI received the master's degree cum laude in mathematics from the University of Trento, Italy. He is currently an Associate Professor in Numerical Analysis with the Department of Industrial Engineering at the University of Trento. His research interests are mainly in the field of numerical analysis and include the development of numerical algorithms for the numerical solution of optimal control problems mainly with indirect methods.



FRANCESCO BIRAL received the master's degree in mechanical engineering from the University of Padova, Italy, and the Ph.D. degree in mechanism and machine theory from the University of Brescia, Italy, in 2000 for his work on minimum lap time of racing vehicles with the use of optimal control. He is currently an Associate Professor with the Department of Industrial Engineering at the University of Trento.

His research interests include symbolic and numerical multi-body dynamics and optimization, constrained optimal control, mainly in the field of vehicle dynamics with special focus on intelligent vehicles and optimal maneuver for racing vehicles. He has 15 years experience in the development and validation of ADAS and AD functions, both for cars and PTWs, gained in several European and industrial funded research projects.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4