
This is a repository copy of *Fast tube model predictive control for driverless cars using linear data-driven models*.

White Rose Research Online URL for this paper:
https://eprints.whiterose.ac.uk/194181/

Version: Accepted Version

---

**Article:**
Vicente, B.A.H., Trodden, P.A. orcid.org/0000-0002-8787-7432 and Anderson, S.R. orcid.org/0000-0002-7452-5681 (2022) Fast tube model predictive control for driverless cars using linear data-driven models. IEEE Transactions on Control Systems Technology, 31 (3). pp. 1395-1410. ISSN 1063-6536

https://doi.org/10.1109/tcst.2022.3224089

---

© 2022 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other users, including reprinting/ republishing this material for advertising or promotional purposes, creating new collective works for resale or redistribution to servers or lists, or reuse of any copyrighted components of this work in other works. Reproduced in accordance with the publisher's self-archiving policy.

**Reuse**
Items deposited in White Rose Research Online are protected by copyright, with all rights reserved unless indicated otherwise. They may be downloaded and/or printed for private study, or other acts as permitted by national copyright laws. The publisher or other rights holders may allow further reproduction and re-use of the full text version. This is indicated by the licence information on the White Rose Research Online record for the item.

**Takedown**
If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing eprints@whiterose.ac.uk including the URL of the record and the reason for the withdrawal request.


eprints@whiterose.ac.uk
https://eprints.whiterose.ac.uk/

# Fast Tube Model Predictive Control for Driverless Cars using Linear Data-Driven Models

Bernardo A. Hernandez Vicente, Paul A. Trodden and Sean R. Anderson

***Abstract*—Model predictive control (MPC) has been widely applied to different aspects of autonomous driving, typically employing nonlinear physically derived models for prediction. However, feedback control systems inherently correct for model errors, thus in many applications it is sufficient to use a linear time-invariant (LTI) model for control design, especially when using robust control methods. This philosophy of approach appears to have been neglected in current driverless car research, and is the research gap that we aim to address here. Namely, instead of deriving meticulous nonlinear physical models of vehicle dynamics, and solving a correspondingly complex optimal control problem, we identify a low-order data-driven LTI model and handle its uncertainty via robust linear MPC methods. We develop a two-step control scheme for driverless cars based on tube MPC (TMPC), which introduces structural robustness, ensuring constraint compliance despite modelling error in the data-driven prediction model. Furthermore, we employ fast optimisation methods designed to exploit the special structure of the linear MPC problem. We evaluate the proposed control scheme using a vehicle model identified from real-world data, and simulations in IPGCarmaker, where the model of the vehicle under control is inherently nonlinear and uses detailed 3D physics. Our results show that an LTI model can be effectively employed for the task of lane-keeping, that TMPC can prevent lane departure and possible collisions due to model uncertainty, and that linear models allow for several algorithmic improvements that can decrease computation time by an order of magnitude compared to naive MPC implementations.**

***Index Terms*—Autonomous driving, linear MPC, data-driven MPC, lane-keeping, robust control, fast MPC.**

## I. Introduction

Fully autonomous driving has the potential to greatly reduce travel time, emissions and accidents, which is why efforts to deploy (semi) autonomous vehicles in the past two decades have steeply increased [1], [2]. The design of a reliable control system is of paramount importance to guarantee safety of the vehicle and its passengers [3], and several control techniques have been proven successful at some aspect of autonomous driving [4]–[7]. Among them, model predictive control (MPC) is appealing due to its intrinsic handling of multivariable systems, constraints, and its inherent robustness [8]. Indeed, MPC has been shown to successfully handle varied autonomous driving requirements [9]–[11].

MPC controllers for driverless cars fall into different classes depending on the goal of the controller, which in turn defines the type of vehicle model used to make predictions. If the objective is to design high-level path planner–followers then several authors [9], [12], [13] rely on kinematic models which ignore the transient dynamics and provide a simple framework to design controllers for a range of vehicles. Since they act in higher layers, these MPC controllers usually require low-level controllers to drive the physical vehicle actuators. The simplicity of kinematic models, even if they are nonlinear as in [13] or time-varying as in [9], allows for the design of MPC controllers with strong theoretical guarantees despite the presence of constraints.

Dynamic models are usually required for the purpose of low-level vehicle control, and most authors employ nonlinear physically derived dynamic models for use in either nonlinear MPC (NMPC) or linear time-varying MPC (LTV-MPC) schemes. In [14], for example, nonlinear models of the lateral dynamics are used in NMPC to perform lane change manoeuvres, whilst in [11], [15] NMPC is used with a nonlinear four-wheel dynamic model to solve the tracking problem. A similar architecture is found in [10], [16] where NMPC is used for kinematic path planning with obstacle avoidance. In order to reduce the complexity of the MPC problem, yet capture the entire driving envelope, several authors use LTV models, including lateral-only dynamics [17], [18], and lateral–longitudinal dynamics [19]. Parameter estimation modules are included in [18], [19] to track model variations; however, the changing nature of the MPC's prediction model results in that, even for a fixed reference, theoretical guarantees are entirely forgone.

Although prevalent, the use of nonlinear dynamical models—and furthermore physically derived models—poses several challenges in the design and deployment of MPC controllers for driverless cars. A nonlinear prediction model results in the optimisation associated with the MPC controller being a nonlinear (nonconvex) program [20], which may pose unachievable computational demands for its on-line solution. Moreover, the accuracy of the prediction model plays a major role on the performance and theoretical guarantees of MPC controllers [21] and, although inherent robustness due to feedback may be present, uncertainties in nonlinear models may result in large prediction errors if unaccounted for [18]. NMPC and LTV-MPC techniques with explicitly built-in robustness provide mitigation for the latter, and have been devised for the purpose of autonomous driving [22], [23]; however, they are tailored to specific model structures and are more complex than their linear counterparts, both in the design and implementation stages. Furthermore, most robust MPC controllers require some knowledge about the magnitude of the modelling uncertainty. For physically derived models this implies generating a measure of confidence for each model parameter, which may require

Bernardo A. Hernandez Vicente is with the Department of Mechanical Engineering, University of Concepcion, Chile (e-mail: behernandez@udec.cl). Paul A. Trodden and Sean R. Anderson are with the Department of Automatic Control and Systems Engineering, The University of Sheffield, Sheffield S1 3JD, U.K. (e-mail: bahernandezv@gmail.com; {p.trodden; s.anderson} @sheffield.ac.uk).

large-scale testing (for a single vehicle) [22] and extensive a-priori knowledge.

While nonlinear physical models are the current prevailing choice in MPC for driverless cars, linear time-invariant (LTI) *identified* models have a track record of successful use in a wide range of control applications, particularly in the industrial and process control domains. Much of this success is built on the fact that LTI models are often sufficiently accurate for control purposes even if the true dynamics are nonlinear, which itself may be a consequence of the tacit observation that feedback control is *linearizing* [24]. However, the use of data-driven models in autonomous driving applications is uncommon [25], [26], particularly in the context of MPC [27]. Physically derived linear models share many drawbacks of their nonlinear counterparts when it comes to assessing their fidelity, particularly since linearisation may skew a-priori knowledge. Data-driven LTI models, on the other hand, are built to fit the available data, hence their accuracy depends on model design choices rather than (the lack of) a-priori knowledge [28]. Moreover, we have recently demonstrated that vehicle dynamics can be well described using data-driven LTI models in comparison to nonlinear and linearised physical models [29].

In this paper, therefore, we propose a novel two-step MPC controller for the purpose of lane-keeping, fitted with an LTI data-driven model for predicting vehicle dynamics. Our proposed controller performs both high-level path-planning using spatial-based optimal control and low-level vehicle dynamics control using linear tube MPC (TMPC) [30]. Our architecture is similar in structure to that in [10], [11], [16], but with several key differences. First and foremost, we use the approach in [29] to identify an LTI dynamic model of the vehicle in velocity space for the MPC predictions. By using LTI models for MPC predictions we are able to tackle some of the challenges previously discussed. We take modelling error explicitly into account via built-in robustness in the form of TMPC, and we develop the TMPC optimisation algorithm for low-level vehicle control based on the fast MPC principles described in [31], which we apply here to TMPC.

To demonstrate the effectiveness of our proposed method we conduct simulation experiments on dynamic models derived from real-world car data and also using simulation in IPGCarMaker [32]. These experiments show that (i) the LTI data-driven models are sufficiently accurate for the task of path-following in normal driving conditions and may provide an alternative to nonlinear solutions; (ii) the use of TMPC (as opposed to MPC) can prevent departure from the lane, and potentially avoid collisions; (iii) the use of fast TMPC can decrease computation time by an order of magnitude, which provides a good baseline for future deployment on target hardware.

The rest of the paper is organized as follows. The objective and scope of the controller are discussed in Section II, followed by a description of the kinematic and data-driven models employed. Section III describes our proposed controller in detail, including a brief account of TMPC. In Sections IV–VI we present the modelling results and discuss in detail the performance of our proposed controller, both with respect to the control objectives and its computational capabilities. We finalise with some directions for future work in Section VII.

## II. Control architecture and scope

In this paper we aim to solve the lane-keeping problem for an autonomous road vehicle. More precisely, we seek to design a control system that provides low-level control actions in order to keep the vehicle as close to the lane centre as possible, while travelling at a desired speed and respecting comfort and safety constraints normally associated with road driving.

We propose a two-step controller to achieve this aim. The first step generates references for the vehicle in the velocity space, using a simple kinematic model in a curvilinear coordinate frame. The second step provides tracking of this reference, using a linear robust MPC controller fitted with an LTI data-driven model of the vehicle's dynamics. The latter is also expressed in the velocity space, meaning that its outputs are the velocities (linear and angular) of the vehicle's centre of gravity (CoG). The choice of velocity space follows previous work showcasing its prediction capabilities [29].

The architecture of our proposed control system is depicted in Figure 1. We assume we obtain measurements for the vehicle's velocity, yaw rate, and global position, which are preprocessed to obtain the position of the car in a curvilinear frame of reference. We also assume that some information about the road's shape is known a-priori, which is a common assumption in path-following algorithms [9], [10], [13], [15]. In the remainder of this section we introduce the models that are going to be used by both steps of our controller, and the methods by which they are obtained.

### A. Curvilinear kinematic model

The first step of our controller employs a simple kinematic model of the vehicle's CoG described in a curvilinear reference frame with its axes oriented alongside the lane's centre line (also known as Frenet reference frame). This allows for a simple description of the distance to centre of the lane, which is one of our main regulation objectives and hence will drive the reference generation.

The reference frame is depicted in Figure 2, where $s(t)$ is the distance travelled along the curve, $y_d(t)$ is the perpendicular distance to the curve, and the pair $(V(t), \psi(t))$ describe the vehicle's CoG velocity in magnitude and orientation. The variable $\gamma(s)$ is the path's curvature (or heading) with respect to some fixed reference frame $(X, Y)$, and we assume it is perfectly known.

Define $\omega(t)$ as the vehicle's yaw rate, and $\dot{}$ and $'$ as derivatives with respect to $t$ and $s$ respectively. In the curvilinear frame of reference the kinematic model of the vehicle is described by

$$\dot{s}(t) = \frac{V(t)\cos(\psi(t) - \gamma(s(t)))}{1 - y_d(t)\gamma'(s(t))} \tag{1a}$$

$$\dot{y_d}(t) = V(t)\sin(\psi(t) - \gamma(s(t))) \tag{1b}$$

$$\dot{\psi}(t) = \omega(t). \tag{1c}$$

In this paper, as in [10], we use Bezier curves to obtain an analytical description for $\gamma(\cdot)$ and its derivative. Note however that, given the nature of our data, we test our path-following algorithm against real world roads, hence we employ Bezier curves of different orders as necessary.

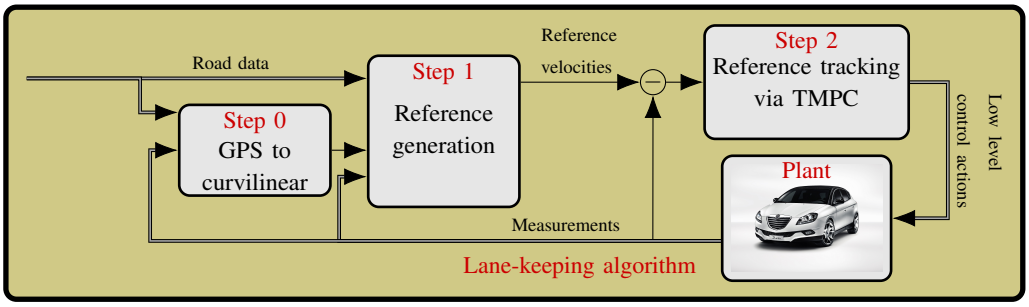


Figure 1. Architecture of the two-step lane-keeping controller. The first step generates velocity references, which are then tracked via TMPC in the second step

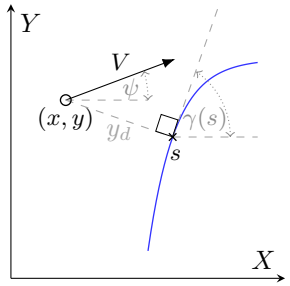


Figure 2. Curvilinear reference frame employed for reference generation.

### B. Data-driven LTI model

In the following we describe our selection of data-driven model and the estimation procedure (the data employed is depicted in Section IV-A). We propose to use a black-box linear model in state space form, that is

$$\dot{x}(t) = Ax(t) + Bu(t) + \mu(t) \tag{2a}$$
$$y(t) = Cx(t) + \nu(t) \tag{2b}$$

where $x \in \mathbb{R}^n$ is the state, $u \in \mathbb{R}^m$ is the input, $y \in \mathbb{R}^p$ is the output, $\nu \in \mathbb{R}^p$ is the measurement noise, $\mu \in \mathbb{R}^n$ is the process noise and the matrices $A, B, C$ are of appropriate dimension. The output vector is composed of the variables we are interested in regulating to reference values, which in this case are linear velocity in the plane $V(t)$ and yaw rate $\dot{\psi}(t)$. As inputs we choose a subset of the low-level inputs available for autonomous driving: pedals (gas and brake), engine torque and steering wheel. We also include road slope as an input in the modelling stage to reduce model uncertainty, but we do not use it for regulation in the MPC context.

The states of the model (contained in $x$) do not necessarily have any physical meaning and are defined via a subspace identification method to best fit the available data: the number of states, $n$, is a design variable, and should be kept small to avoid overfitting. Finally, the process noise $\mu$ represents the uncertainty in our model, and allows to capture the effects the nonlinearities on the system. In a physical modelling context this term can be parametrised as is done in [17]. Our data-driven approach does not allow for such parametrisation, thus we asses the impact of $\mu$ in prediction via simulation.

The model in (2) is continuous time, but for the purpose of estimating its parameters we use uniformly sampled data with sampling frequency $F_s$, total samples $N_d$ and sampling times $t_k$ with $k \in [0, N_d]$; the sampling details are described in Section IV-A. We estimate the state space matrices in (2) in two steps. First we employ a closed-form subspace identification method [33] to obtain a set of matrices $A_\circ$, $B_\circ$ and $C_\circ$ that produce a simulated output, say $\hat{y}$, that best fits the data in a least-squares sense [33]. The second step consists of a refinement of the parameters in $A_\circ$, $B_\circ$ and $C_\circ$ via the following nonlinear optimisation:

$$\min_{\theta} \sum_{k=1}^{N_d} \left(y(t_k) - \hat{y}(t_k, \theta)\right)^\top \Upsilon \left(y(t_k) - \hat{y}(t_k, \theta)\right), \tag{3}$$

where $\theta$ is a vector that contains all the parameters in $A_\circ$, $B_\circ$ and $C_\circ$, and $\Upsilon$ is a weight used to normalize the different outputs. The optimisation (3) is nonlinear in the parameters, despite (2) being a linear model, because no model is given to the noise $\nu$ [33].

The performance of the data-driven models is assessed via a normalised fit metric associated to the cost in (3):

$$\mathcal{F} = 100\left(1 - {||\boldsymbol{y}_i - \hat{\boldsymbol{y}}_i||_2}/{||\boldsymbol{y}_i - \bar{\boldsymbol{y}}_i||_2}\right),$$

where the subindex $i$ implies elementwise operations, the bold variables represent the entire time series and the bar denotes a mean value. A value of $\mathcal{F} = 100\%$ indicates a perfect fit. We also analysed the variance accounted for metric (VAF)

$$\mathcal{V} = 100\left(1 - {\mathrm{var}(\boldsymbol{y}_i - \hat{\boldsymbol{y}}_i)}/{\mathrm{var}(\boldsymbol{y}_i)}\right),$$

for which a value of $\mathcal{V} = 100\%$ indicates that the model is able to explain the entire variance of the measurements. This, however, is not expected since we do not include a model for measurement noise and hence cannot expect to explain it.

**Remark 1.** TMPC for linear models is a mature technique that enjoys an array of theoretical guarantees provided the model meets certain conditions. In particular, we require the pair $(A, B)$ to be stabilisable and the noise/disturbances affecting the model to be bounded. The closed-form subspace identification method is implemented with a constraint that forces $A$ to be stable, as we do not expect instability given our input/output pairs, but we can only guarantee boundedness of the uncertainty in a probabilistic sense, which we discuss in Section IV-C.

### C. Nonlinear physics-based model

In order to properly frame our contribution, we will also discuss the performance of our controller with respect to

nonlinear MPC, fitted with a nonlinear physical model for prediction (as is prevalent in the literature [15], [22], [34]). To represent the nonlinear dynamics of the vehicle, we implement a coupled (longitudinal–lateral) four-wheeled model as described in [29] (referred to as NLPM henceforth). The key features of the NLPM are one-way coupling from longitudinal to lateral dynamics and the use of the magic formula for tyre force estimation. The unknown parameters of the NLPM are fitted following the procedure described in Section II-B.

## III. TMPC FOR ROBUST PATH-FOLLOWING

### A. Reference generation

The first step of our controller defines the references in the velocity space. To do so we employ the kinematic model in (1) to predict the vehicle's movement and optimise its velocity and yaw rate in order to minimise its deviation from the path and adhere to the required velocity specifications. At this stage, $V(t)$ and $\dot{\psi}(t)$ act as inputs for the kinematic model, so to optimise them we propose the following MPC-like discrete optimisation problem to be solved in a receding horizon fashion at each sampling time $t$:

$$\begin{aligned} &\mathbb{P}_{N_c}(V(t), \omega(t), s(t), y_d(t), \psi(t)): \\ &\min_{\boldsymbol{V}(t), \boldsymbol{\omega}(t)} J_{N_c}\left(\boldsymbol{V}(t), \boldsymbol{\omega}(t), V(t), \omega(t), s(t), y_d(t), \psi(t)\right) \end{aligned} \tag{4}$$

where $\boldsymbol{V}(t)$ and $\boldsymbol{\omega}(t)$ are the predicted velocity and yaw rate profiles made at sampling instant $t$, that is $\boldsymbol{V}(t) = [V(t+T_s|t), V(t+2T_s|t), \ldots, V(t+N_gT_s|t)]^\top$, $N_g$ is the prediction horizon, $T_s$ the sampling time and the notation $(k|t)$ indicates prediction $k$ made at time $t$. Optimisation (4) is subject to constraints that include a forward Euler discretisation of the kinematic equations (1), initial conditions for the states $s(t)$, $y_d(t)$, $\psi(t)$ and initial condition for the inputs $V(t)$, $\omega(t)$. The latter explains why $V(t|t)$ is missing from $\boldsymbol{V}(t)$, since it is defined by measurements and not available for optimisation.

The kinematic model is nonlinear, hence (4) is a nonlinear program (NLP). To reduce computational complexity, we employ the concept of move-blocking MPC [35]. The prediction horizon is evenly divided in $N_c$ blocks during which the optimisation variables $V(k|t)$ and $\omega(k|t)$ remain constant. Formally, given $i \in [1, N_c]$ and $\bar{N} = {}^{N_g}/_{N_c}$, then $(V(k|t), \omega(k|t)) = \left(\bar{V}_i, \bar{\omega}_i\right)$ for all $k \in \left[t + (1 + (i-1)\bar{N})T_s, t + i\bar{N}T_s\right]$.

The cost function in (4) is defined as

$$\begin{aligned} &J_g\left(\boldsymbol{V}(t), \boldsymbol{\omega}(t), V(t), \omega(t), s(t), y_d(t), \psi(t)\right) = \\ &\sum_{k=t}^{t+N_gT_s} \left(Q_y y_d^2(k|t) + Q_{\dot{y}} \dot{y}_d^2(k|t) + Q_{\dot{s}} \left(\dot{s}(k|t) - V_g\right)^2\right) \end{aligned} \tag{5}$$

where $Q_y$, $Q_{\dot{y}}$ and $Q_{\dot{s}}$ are weights. In the cost function (5), the first term penalises lateral deviations from the path, the second term promotes the vehicle's alignment with the path, and the third term penalises deviations from a target velocity $V_g$. Finally, the optimised variables are defined by $\boldsymbol{V}^\star(t) = [V^\star(t+T_s|t), V^\star(t+2T_s|t), \ldots, V^\star(t+N_gT_s|t)]^\top$. In what follows, we refer to (4)–(5) as a receding horizon nonlinear optimal control problem (RHNOCP).

Given a fixed horizon $N_g$, dividing it into fewer blocks reduces the complexity of the RHNOCP, however it may result in poor performance and lose of manoeuvrability. Consider the extreme case where $N_c = 1$, if large changes in the path's curvature are present over the prediction horizon, a single value of yaw rate applied across the entire horizon could result in inaccurate tracking of the path. A small number of blocks should be paired with short prediction horizons, however this could result in sharp corners being first encountered in the planning horizon only when the vehicle is in close proximity to them, not allowing enough time for the vehicle to slow down appropriately. To tackle these issues and allow for reduced complexity of the optimisation, we propose to modulate the target velocity $V_g$ with respect to the change in curvature of the oncoming road further from the prediction capabilities of the optimisation horizon. The modulated velocity is defined by

$$V_f(s) = \begin{cases} e^{-\alpha(s)/\bar{\alpha}} V_g, & e^{-\alpha(s)/\bar{\alpha}} V_g \geq v_g \\ v_g, & \text{otherwise,} \end{cases}$$

where $\alpha(s)$ is the total change in curvature throughout a fixed and arbitrary length of road ahead of the current position of the vehicle, $\bar{\alpha}$ is an arbitrary amount of allowable curvature change within this length of road, and $v_g$ is an arbitrarily defined minimum required velocity.

The RHNOCP is similar to the approach described in [10] for reference generation, but with some key differences. The approach in [10] implements an explicit method to solve the reference generation problem, placing the emphasis on the inclusion of collision avoidance (non-convex) constraints. Nevertheless, the explicit solution is inherently suboptimal, and the subsequent reference tracking is implemented via proportional controllers over a bicycle vehicle model. We, on the other hand, do not consider collision avoidance and employ the move-blocking MPC paradigm, which allows longer prediction horizons and the use of general purpose solvers. The latter, in turn, permits the following:

- To introduce input feedback in the reference generation. Indeed, although linear velocity and yaw rate are considered inputs of the kinematic model, they are indeed the dynamic response of the vehicle, hence their value at the current sampling time is not manipulable. We acknowledge and handle this via the inclusion of initial condition constraints in (4) for $V(t)$ and $\omega(t)$, thereby enhancing the kinematic model with some degree of inertia.
- To initialize the vehicle at arbitrary conditions (position and velocity), and with any target velocity, unlike [10] whose bespoke optimisation algorithm requires initialization on the path, at rest, and with a null target velocity.

### B. Solution of the RHNOCP

To solve the RHNOCP we employ the nonlinear programming (NLP) solver known as IPOPT [36], via one of its MATLAB interfaces [37]. IPOPT is a general purpose solver that implements an interior point line search method to find a local solution of the NLP. In the context of the RHNOCP (4), i.e. without constraints, the implementation of IPOPT requires the user to provide the value of the cost (5), its gradient and Hessian, at each iteration of the solver. We estimate the gradient

of (5) numerically, while the Hessian is estimated via a Quasi-Newton method native to IPOPT. IPOPT employs different termination criteria to decide whether the current iteration corresponds to a local minima, including iteration number and tolerances on various indicators such as cost function change. The specific values used are presented in Section V.

**Remark 2.** The RHNOCP has no hard constraints, nor terminal ingredients, which are common in MPC. Thus there is no guarantees on the behaviour of the optimal cost function, i.e. whether the optimised trajectory $(\psi(t), y_d(t), s(t))$ remains close to the lane's centre. Formal guarantees are currently under investigation, however we do observe an acceptable tracking behaviour in simulation.

### C. Reference tracking via TMPC

In the second step of our controller we track the references defined by the RHNOCP. To do so we implement the structurally robust MPC controller known as TMPC, which is able to take into account external perturbations that may affect the car and also the modelling error inherent to the data-driven model used to make the predictions. For completeness, we now recall some standard definitions and present a brief description of the TMPC optimal control problem applied in our framework, but the reader is referred to [38, Chapter 3] for a detailed description of the technique. In what follows "$\oplus$" denotes the Minkowski sum of sets and "$\ominus$" the Pontryagin set subtraction [39].

Note that we describe the TMPC technique in a state-feedback setting, although the prediction models introduced in Section II-B are in output-feedback form. We do this to simplify the exposition by reducing notation, nevertheless the extension to output-feedback is easily achieved by including a suitable state estimation technique (such as the Kalman Filter), and following the methodology described in [40]. Furthermore, when processing the data through our modelling approach we find a good trade-off between simplicity and match to dynamics in a low order model, for which the outputs are the states (see Section IV). By choosing such model we remove the need for state estimation and the additional machinery in TMPC to deal with estimation error.

**Definition 1** (Positive invariant (PI) set)**.** A set $\mathbb{T} \subset \mathbb{R}^n$ is a PI set for the dynamics $x(t+1) = \bar{A}_K x(t)$ if $\bar{A}_K \mathbb{T} \subseteq \mathbb{T}$.

**Definition 2** (Robust PI (RPI) set)**.** A set $\mathbb{S} \subset \mathbb{R}^n$ is an RPI set for the dynamics $x(t+1) = \bar{A}_K x(t) + w(t)$ with $w(t) \in \mathbb{W}$ if $\bar{A}_K \mathbb{S} \oplus \mathbb{W} \subseteq \mathbb{S}$.

Consider a discrete time version of (2)

$$x(t+1) = A_d x(t) + B_d u(t) + w(t) \tag{6}$$

where $A_d$ and $B_d$ depend on $A$, $B$ and the discretisation time $T_s$, and $w(t)$ is composed of the modelling uncertainties and measurement noise. Furthermore, suppose that states and inputs are subject to constraints in the form of compact and convex sets $\mathbb{X} \subset \mathbb{R}^n$ and $\mathbb{U} \subset \mathbb{R}^m$ respectively, and the disturbances are bounded inside a compact and convex set $\mathbb{W} \subset \mathbb{R}^n$. Consider now an undisturbed representation of (6)

$$z(t+1) = A_d z(t) + B_d v(t),$$

and a certain steady state pair $(z_{ss}, v_{ss})$ that is to be tracked. The resulting error system is

$$\hat{z}(t+1) = A_d \hat{z}(t) + B_d \hat{v}(t), \tag{7}$$

where $\hat{z}(t) = z(t) - z_{ss}$ and $\hat{v}(t) = v(t) - v_{ss}$. Note that the model matrices $(A_d, B_d)$ are uncertain, thus the computation of $v_{ss}$—for given a target $z_{ss}$—also carries uncertainty. The control law employed to regulate the disturbed plant is

$$u(t) = \hat{\kappa}(x(t), \hat{z}(t)) = \kappa\left(\hat{z}(t)\right) + v_{ss} + K_T\left(x(t) - z(t)\right), \tag{8}$$

where $K_T$ is stabilizing for $(A_d, B_d)$ and $\kappa\left(z(t)\right)$ is the receding horizon control law that stems from a nominal MPC controller designed to regulate the error model in (7), subject to tightened and translated versions of the constraints represented by the sets $\mathbb{Z} \subset \mathbb{R}^n$ and $\mathbb{V} \subset \mathbb{R}^m$.

The optimal control problem (OCP) to be solved at each sampling time is defined as:

$$\mathbb{P}_N(\hat{z}(t)) : \min_{\hat{\boldsymbol{v}}} J_N\left(\hat{z}, \hat{\boldsymbol{v}}\right) \tag{9}$$

subject to (for $k = t, \ldots, t+N-1$):

$$\hat{z}(k+1|t) = A_d \hat{z}(k|t) + B_d \hat{v}(k|t) \tag{10a}$$

$$(\hat{z}(k|t), \hat{v}(k|t)) \in (\mathbb{Z} \ominus \{z_{ss}\} \times \mathbb{V} \ominus \{v_{ss}\}) \tag{10b}$$

$$\hat{z}(N|t) \in \mathbb{Z}_f\left(z_{ss}, v_{ss}\right). \tag{10c}$$

where the cost function $J_N\left(\hat{z}, \hat{\boldsymbol{v}}\right)$ is defined as the usual quadratic cost with terminal penalty

$$J_N\left(\hat{z}, \hat{\boldsymbol{v}}\right) = l_f\left(\hat{z}(t+N|t)\right) + \sum_{k=t}^{t+N-1} l\left(\hat{z}(k|t), \hat{v}(k|t)\right)$$

with

$$l\left(\hat{z}(k|t), \hat{v}(k|t)\right) = \hat{z}(k|t)^T Q \hat{z}(k|t) + \hat{v}(k|t)^T R \hat{v}(k|t) \tag{11a}$$

$$l_f\left(\hat{z}(t+N|t)\right) = \hat{z}(t+N|t)^T P \hat{z}(t+N|t), \tag{11b}$$

In (11) $Q$, $R$ and $P$ are respectively the state, input and terminal penalties. The controller's prediction horizon is $N$ and the optimization variable $\hat{\boldsymbol{v}}$ represents the sequence of control actions throughout the prediction horizon, i.e. $\hat{\boldsymbol{v}} = \{\hat{v}(t|t), \hat{v}(t+1|t), \ldots, \hat{v}(t+N-1|t)\}$. The solution to (9)–(10) is a sequence of optimal inputs $\boldsymbol{v}^*(\hat{z}(t))$, the set $\mathcal{Z}_N\left(z_{ss}, v_{ss}\right)$ is the set of all states $\hat{z}$ such that such a solution exists for a given pair $(z_{ss}, v_{ss})$, and the implicit nominal control law is defined as the first control action of the optimal sequence $\kappa\left(\hat{z}(t)\right) = \hat{v}^*(t|t)$.

### D. Fast tube model predictive control

The optimisation problem of standard MPC implementations is typically formulated as a quadratic program (QP), which can be slow to solve using general purpose methods. A collection of methods for fast MPC is demonstrated in [31], including (i) exploiting the structure of the MPC problem to reduce the computational complexity from cubic, $O(N^3)$, to linear, $O(N)$, in the planning horizon $N$, (ii) warm-starting the optimisation, where the control solution from the previous time-step is used to initialise the current time-step, and (iii) early termination of the optimisation.

One of the advantages of linear TMPC is that its associated optimisation problem (9)–(10) is comparable to that of standard MPC, and hence the methods in [31] can be exploited in our setup. The approach is based on an infeasible start primal barrier method, where the OCP defined in (9)–(10) is formulated as a QP with a barrier to replace the inequality constraints, leading to the approximate problem

$$\begin{aligned} \min_{\boldsymbol{z}} \quad & \boldsymbol{z}^\top H \boldsymbol{z} + \lambda \phi(\boldsymbol{z}) \\ \text{subject to} \quad & \Gamma \boldsymbol{z} = b \end{aligned}$$

where

$$\boldsymbol{z} = (\hat{v}(k), \hat{z}(k+1), \dots, \hat{v}(k+N-1), \hat{z}(k+N))$$

$$H = \text{blkdiag}\{R, Q, R, Q, \cdots, R, P\}$$

$$\Gamma = \begin{pmatrix} -B & I & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & -A & -B & I & \dots & 0 & 0 & 0 \\ 0 & 0 & 0 & -A & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & I & 0 & 0 \\ 0 & 0 & 0 & 0 & \dots & -A & -B & I \end{pmatrix}$$

$$b = (A\hat{z}(k), 0, \dots, 0)^\top,$$

and where $\lambda$ is a barrier parameter and $\phi(\boldsymbol{z})$ is a log barrier function. This barrier method approximates the original optimisation problem, and converges to the optimal solution as the parameter $\lambda$ approaches zero.

Wang and Boyd [31] propose a solution to the primal barrier method defined above using an infeasible start Newton method, and demonstrate how to exploit the structure in $H$ and $\Gamma$ in a fast Newton update that has linear computational complexity in the horizon $N$ (precise details are omitted here for brevity). A further step enables an even greater computational saving: a typical primal barrier method is solved for a decreasing sequence of $\lambda$ values, however they show it is possible to use just a single value of $\lambda$ without loss of performance (provided some offline tuning of $\lambda$). This turns each MPC step into a Newton process, for which it is possible to leverage warm-starts, where the control solution from the previous time-step is used to initialise the current one. The use of warm-starts can reduce the number of Newton iterations per MPC step by an order of magnitude.

The final part of the process is to use early termination via a fixed iteration limit for the Newton method. Although the optimisation is then not necessarily solved to optimality, this does not appear to degrade the control solution. The explanation for this is likely due to the fact that MPC is largely a planning exercise, where future control inputs are discarded. Hence, solving the optimisation to full accuracy places an emphasis on optimizing future control inputs that are never used.

When all these steps for fast TMPC are combined, the solution of the TMPC problem can be obtained orders of magnitude faster than a naive implementation (with the speed-up largely depending on the size of the horizon $N$). These steps are critical to obtaining a computationally efficient controller, enabling the practical implementation of TMPC in driverless cars where the control loops operate at relatively high sample rates. Note also that the use of LTI data-driven models of vehicle dynamics, as proposed here, enables us to leverage the efficient solution of the approximate primal barrier method.

### *E. Stability of the fast TMPC*

The following result summarizes the main properties of the TMPC controller described above.

**Assumption 1.** The steady state pair $(z_{ss}, v_{ss})$ is such that $\mathbb{Z} \ominus \{z_{ss}\}$, $\mathbb{V} \ominus \{v_{ss}\}$ and $\mathbb{Z}_f(z_{ss}, v_{ss})$ have non-empty interiors, and $\mathcal{Z}_n(z_{ss}, v_{ss}) \neq \emptyset$. The set $\mathbb{S} \subset \mathbb{R}^n$ is an RPI set for the closed-loop $A_d + B_d K_T$ in presence of disturbances $\mathbb{W}$. The tightened constraint sets are such that $\mathbb{Z} \subseteq \mathbb{X} \ominus \mathbb{S}$ and $\mathbb{V} \subseteq \mathbb{U} \ominus K_T \mathbb{S}$. The set $\mathbb{Z}_f$ is a PI set for $A_{d,K} = A_d + B_d K_f$ such that $\mathbb{Z}_f \subseteq \mathbb{Z} \ominus \{z_{ss}\}$ and $K_f \mathbb{Z}_f \subseteq \mathbb{V} \ominus \{v_{ss}\}$. The LQR cost weights fulfil $Q \geq 0$, $R > 0$ and $A_{d,K}^\top P A_{d,K} + \left(Q + K_f^\top R K_f\right) \leq P$

**Theorem 1** (Stability[1])**.** Suppose Assumption 1 holds. If (a) the steady state pair $(z_{ss}, v_{ss})$ is constant, and (b) the nominal system is initialized such that $x(0) \in \{z(0)\} \oplus \mathbb{S} \subset \mathcal{Z}_N \oplus \mathbb{S}$, then the OCP (9)–(10) is feasible at all times, the true state and input constraints are met at all times, and the set $\mathbb{S} \times \{z_{ss}\}$ is exponentially stable for (6) when in closed-loop with (8).

In our architecture, the steady state to be tracked is set to the velocity and yaw rate references generated by the RHNOCP, i.e. $z_{ss} = [V^\star(t+1|T_s), \omega^\star(t+1|T_s)]^\top$, and the corresponding input $v_{ss}$ is computed via the prediction model. The RHNOCP updates the velocities to be tracked at each time instant to account for current vehicle attitude, which is necessary due to modelling error and the different modelling paradigms employed in each step of the controller. This results in the need to continuously update the terminal constraint set, which could be dealt with as shown in [42], [43], however it also breaks a key assumption necessary for Theorem 1 to hold, namely the invariance of the steady state pair to be tracked.

There exist some MPC approaches that enjoy stability guarantees in the context of driverless cars, like [17], where LTV models for obstacle avoidance are studied. However, it is not clear if this result remains valid in view of a changing reference, and feasibility of the optimisation is only assumed. Another interesting approach is [13], where an input-to-state stability resultn is obtained for a unicycle robot, yet in the context of kinematic models only. MPC for tracking changing references has also been studied without a specific application attached. In [44], for example, the tracking of periodically changing references is studied and a result of asymptotic stability is provided. A more general issue is studied in [20], [45], where references that change via the system dynamics (or more specifically, the prediction model), are analysed. The proposed controllers enjoy stability results that depend on the length of the prediction horizon, removing the need for the online (or offline) computation of terminal constraint sets.

The RHNOCP could indeed be modified to guarantee that its resulting references are reachable via the prediction model, however the techniques in [20], [45] also require that the reference is entirely determined at initialization (at least for

[1]We omit the proof for brevity, but refer the reader to [8], [38] for the proof and to [41] for a discussion about model uncertainty.

a duration equal to the prediction horizon). This assumption is unachievable in our application, since the references are updated at each time instant by the RHNOCP to take into account the current attitude of the vehicle, which does not match predictions due to inherent modelling error.

The problem of tracking a reference that is randomly changing within a compact set is studied in [46]. The proposed controller is shown to drive the state towards a neighbourhood of a set of possible references in finite time. This is akin to our own setup, since from the perspective of TMPC the reference is indeed randomly changing. Nevertheless, a fixed (and possibly tight) bound on the reference would constrain the pool of applicability to manoeuvres with low levels of curvature.

An analogous issue is studied in [47], where the stabilizing ingredients of a standard MPC controller are modified to account for a disturbance whose prediction is available, but may be different at each time instant. The modifications proposed in [47] result in finite time convergence of the closed-loop to a sublevel set of the MPC's value function, whose size depends on a measure of the allowable change in the disturbance prediction. In our problem, the error system could be recast as

$$\hat{z}(t+1) = A_d\hat{z}(t) + B_d\hat{v}(t) + \hat{w}(t),$$

where $\hat{w}(t)$ represents the change in references from one time instant to the next, that is $\hat{w}(t) = A_d(z_{ss}(t-1) - z_{ss}(t)) + B_d(v_{ss}(t-1) - v_{ss}(t))$, allowing for such an implementation to be employed. Nevertheless, the authors acknowledge the limited practical use of the approach due to the numerous parameters that need to be computationally estimated.

Furthermore, we attempt to solve the optimisation associated to the TMPC via the fast MPC algorithm described in Section III-D, which has early termination as a key feature. In practice, this results in suboptimality of the MPC solution and raises the well-known issue of whether the controller provides stability in view of this, since conventional stability analysis in MPC assumes that the global minimum is achieved each time the problem is solved. Added to this is the fact that the true dynamics are unknown and the identified dynamics are merely an approximation, with inevitable modelling error.

While several results are available establishing robust stability of the closed loop in response to both issues (e.g. [48]–[50], we do not attempt to adapt them to our setting because of their reliance on assumptions on the true dynamics and design conditions on the costs and constraints. Instead, we note that the two-step architecture of our controller is such that the reference employed in the TMPC is computed via its own MPC-like problem, hence it enjoys some preview of the road ahead which promotes the smoothness of the reference profile. In other words, no large changes in the reference values are observed over one time step, which is in line with the results in [47]. Furthermore, the controller shows a good tracking performance in practice, particularly for long horizons, which is in line with the results shown in [20], [45]. Therefore, given that the objective of this paper is not to develop a new tracking MPC technique alongside a manifold of theoretical guarantees, we forgo from further modifications of the TMPC problem.

### F. Unconstrained reference generation

A vehicle's attitude is usually bound to lie within a certain driving envelope defined by regulations and safety requirements (particularly during normal driving conditions). This envelope is represented by the velocity and yaw rate constraint sets we enforce in the reference tracking step, however the RHNOCP is an unconstrained optimisation problem, hence the generated references could lie outside of the driving envelope.

IPOPT allows for the inclusion of constraints in the optimisation, however we have decided to keep the RHONCP constraints-free to avoid creating an undesirable interdependency between the reference generator and the vehicle's dynamic model. Indeed, a constrained RHNOCP would guarantee references within the constraint sets of the TMPC, however the latter do not depend solely on the driving envelope, but also on the TMPC controller parameters (such as the tube gain used for tightening) and the identified dynamic model. By not including constraints in the RHNOCP we favour a modular design, and allow for further development of the reference tracking step without requiring modifications upstream. Moreover, we identify the RHNOCP as the most critical point in computation (c.f. Section V-C), therefore omitting constraints also alleviates the processing load associated to the RHNOCP.

In order to guarantee that the references to be tracked are within the constraint sets of the TMPC, we implement a projection step between Steps 1 and 2. Our simulations show the scheme works well with an unconstrained RHNOCP, with the velocity references violating the TMPC bounds by a maximum of 2%.

## IV. Data-driven modelling results

We first present the data-driven modelling results, since the characteristics of the prediction model are key in designing and tuning MPC controllers.

### A. Experimental data

The experimental data was gathered through $108\,\mathrm{km}$ of driving on public roads in the Piemonte region in Italy, on a Lancia Delta car. The route was designed to include a typical selection of extra-urban and urban roads, motorways, roundabouts and intersections, but the driving itself was unscripted since it had to accommodate for real-time traffic conditions. The driving route is depicted in Figure 3(a) and it was circumnavigated twice. The entire set of data was divided into training and validation portions as shown by the grey line in Figure 3(a), with $2501\,\mathrm{s}$ of data for training and $2383\,\mathrm{s}$ for validation. The data was sampled at $100\,\mathrm{Hz}$, however the power of the relevant signals was found to be concentrated at very low frequencies. In view of this we resampled it at $20\,\mathrm{Hz}$, to decrease the computational load of the estimation and control algorithms. Note also that we use $T_s = 0.05\mathrm{s}$ for discretisation purposes of all our models. In what follows, we refer to post-gearbox torque, brake master cylinder pressure, steering angle and road inclination by $T$, $P$, $\delta$ and $\phi$. Figure 3(b), (c) show the measurements employed to drive the identification procedure

Table I
PERFORMANCE METRIC SUMMARY FOR THE LANCIA DELTA MODELLING.

| Configuration | | | $V$ | | $\omega$ | |
|---|---|---|---|---|---|---|
| | $n$ | $r$ | $\mathcal{F}\%$ | $\mathcal{V}\%$ | $\mathcal{F}\%$ | $\mathcal{V}\%$ |
| $(V,\omega) \leftarrow (T,P,\phi,\delta)$ | 2 | – | 80.19 | 96.06 | 63.50 | 86.82 |
| – | 3 | – | 82.49 | 97.71 | 64.32 | 86.79 |
| – | 4 | – | 84.98 | 97.91 | 64.94 | 88.02 |
| $V \leftarrow (rT-P,\phi)$ | 1 | 16 | 80.76 | 97.00 | – | – |
| $\omega \leftarrow \delta$ | 1 | – | – | – | 64.66 | 87.52 |
| NLPM | | | 68.90 | 93.70 | 43.00 | 66.70 |

Table II
PERFORMANCE METRIC SUMMARY FOR THE MEGANE MODELLING.

| Configuration | | | $V$ | | $\omega$ | |
|---|---|---|---|---|---|---|
| | $n$ | $r$ | $\mathcal{F}\%$ | $\mathcal{V}\%$ | $\mathcal{F}\%$ | $\mathcal{V}\%$ |
| $(V,\omega) \leftarrow (rT-P,\delta)$ | 2 | 15.8 | 84.78 | 98.01 | 63.01 | 86.33 |
| $V \leftarrow (rT-P)$ | 1 | 16 | 84.31 | 97.96 | – | – |
| $\omega \leftarrow \delta$ | 1 | – | – | – | 63.12 | 86.46 |

Table III
ONE-STEP AHEAD ERROR BOUNDS COMPUTED VIA SIMULATION FOR THE MODELS OF THE LANCIA DELTA AND MEGANE.

| Vehicle | $e_{V,LB}$ m/s | $e_{V,UB}$ m/s | $e_{\omega,LB}$ rad/s | $e_{\omega,UB}$ rad/s |
|---|---|---|---|---|
| Lancia Delta | $-0.044$ | $0.044$ | $-0.012$ | $0.012$ |
| NLPM | $-0.012$ | $0.012$ | $-0.024$ | $0.024$ |
| Megane | $-0.010$ | $0.010$ | $-0.041$ | $0.041$ |

for a part of the training portion, while Figure 3(d)–(g) show the corresponding inputs[2].

Table I shows the performance metrics computed over the validation portion of the data for several modelling configurations, including the NLPM. We first obtained models with a $(V,\omega) \leftarrow (T,P,\delta,\phi)$ input-output configuration and state orders ranging from 2 to 4. The performance of the estimated state-space models (see Table I) does not change significantly throughout the different state orders, albeit the number of parameters grows from 16 to 40. In view of the above, and to avoid overfitting, we decide to set the state order to $n = 2$. This also has the advantage of avoiding state estimation, allowing us to employ the state-feedback TMPC described in Section III directly. The performance of the NLPM is also shown in Table I.

In normal driving, however, it is unexpected to experience acceleration and braking simultaneously, so we proceed to merge these two into a single input. We define the new merged input as $TP = rT - P$, where $r$ is scaling factor that was tuned with respect to model fit metrics. Furthermore, from the cross terms in the state and input matrices of $(V,\omega) \leftarrow (T,P,\phi,\delta)$ it appears that the coupling between longitudinal and lateral dynamics is weak, at least within the available data. In view of the above is that we estimate uncoupled models, whose performance metrics differ from those of the fully coupled model by less than 2% (cf. Table I).

Finally, note that the NLPM has poorer performance metrics than any of the linear data-driven models. A detailed discussion pertaining these results can be found in [29], however the main reason for this is the numerical complexity involved in an accurate estimation of the NLPM parameters.

### B. CarMaker data

The IPGCarMaker software provides high fidelity simulations of vehicle dynamics through a 3D physics environment, and allows flexible coupling with software such as Simulink for measurement acquisition and controller testing. IPGCarMaker allows for a meticulous vehicle design process that would allow us to replicate the Lancia Delta car in the software. However, the main reason for using a data-driven model of the vehicle dynamics is that we do not know (or have no means to obtain) several of the many physical parameters that define the vehicle.

In order to test our algorithm with a high fidelity simulator then, we employ a vehicle already loaded in the IPGCarMaker database. We chose a Renault Megane for its similarities to the Lancia Delta. In order to replicate the conditions under which we obtained the Lancia results (see Table I) we drive the Megane with the same set of inputs the Lancia was driven. We feed gas pedal $G$, brake pedal $B$ and steering angle to the IPGCarMaker simulation environment, but we allow the internal driver to choose the appropriate gear. Furthermore, since the vehicles are not identical, their response will necessarily be different. In view of this we do not load any particular route, but let the vehicle to be driven over a flat square of pavement, $60\,\mathrm{km}$ in length.

The results of this modelling experiment are shown in Table II. Note that, although we fed gas and brake pedal signals, we still employ post-gearbox torque and master cylinder pressure as inputs to our model. We do so to avoid the nonlinearities that most likely exist in the engine dynamics. The performance metrics are similar to those from the Lancia Delta car both in magnitude and shape, hence we also move forward with the uncoupled models for the design of the TMPC (c.f. Appendix for the model parameter values).

### C. Modelling error quantification

It follows from the fit metrics that the data-driven models, for both the Lancia and the Megane, present some modelling error. Indeed, 60% fit in yaw rate may seem unacceptably low to guarantee safety. Nevertheless, the TMPC step of our controller is able to deal with some modelling error as long as it can be bounded inside the perturbation set $\mathbb{W}$. In order to quantify the modelling error as an uncertainty like the one described in (6) we compute the one-step ahead prediction error of our model for each data point in the validation data set, and fit a normal distribution to the obtained values. We then define the bounds on the modelling error as two standard deviations of the fitted normal distribution, which account for a 95% of the error values. With this approach we obtain the error bounds depicted in Table III, where LB and UB stand for lower and upper bound respectively.

[2]For commercial confidentiality, we report gas pedal and not post-gearbox torque. We employ road inclination to improve model performance but is later dropped in the controller analysis due to it being an uncontrolled input.

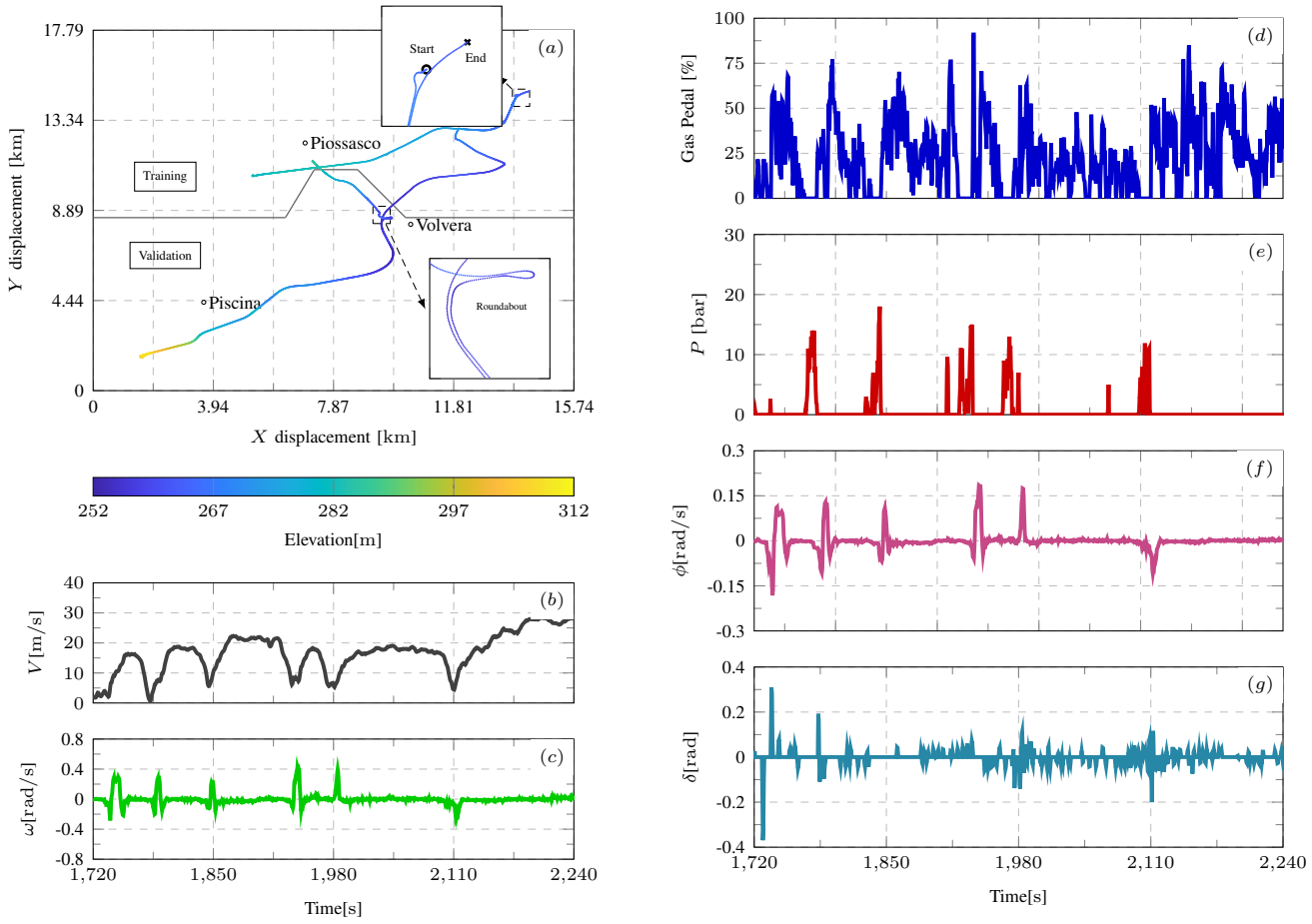


Figure 3. Experimental data for the Lancia Delta: (a) Driving route, the gray line indicates the split of the route between training data (above) and validation data (below), (b) Longitudinal velocity $V$, (c) Yaw rate $\omega$, (d) Gas pedal position, (e) Brake pressure $P$, (f) Road slope angle $\phi$, (g) Steering angle $\delta$.

## V. Path-following algorithm performance

### A. Nonlinear baseline

To acknowledge the prevalence of nonlinear solutions in autonomous driving, we begin with a performance comparison between linear data-driven and nonlinear physics-based MPC. The latter is a nonlinear MPC (NMPC) fitted with the NLPM as a prediction model. Note however that the NLPM has a poorer fit to data than the linear data-driven models, thus to avoid reaching biased conclusions, we will temporarily consider the NLPM as the true plant (i.e. the NLPM will be used to simulate the feedback, resulting in perfect prediction of the NMPC).

Note however, that we do not attempt to produce a linear controller that outperforms carefully designed nonlinear controllers, nor do we make such claims; rather we propose a novel linear data-driven solution in the context of MPC for autonomous driving. Thereby we employ a general NMPC solution provided by the *nlmpc* object in MATLAB. The comparison is based on the driving performance of our two-step controller over certain cornering manoeuvres, with the reference tracking step being solved by NMPC and linear data-driven MPC (a non-structurally robust implementation, which does not tighten constraints nor allocates part of the control authority for a corrective action).

At each time instant the NMPC solves the OCP $\mathbb{P}_N^n(x(t))$ : $\min_{\boldsymbol{u}} J_N^n(\boldsymbol{u})$ subject to (for $k = t, \dots, t+N-1$):

$$x(k+1|t) = f_d\left(x(k|t), u(k|t)\right)$$
$$\left(x(k|t), u(k|t)\right) \in (\mathbb{X} \times \mathbb{U}),$$

where $f_d$ represents a discretization of the NLPM, and

$$J_N^n(\boldsymbol{u}) = \sum_{k=t}^{t+N-1} \hat{x}(k|t)^T Q \hat{x}(k|t) + \Delta u(k|t)^T R \Delta u(k|t)$$
$$\hat{x}(k|t) = x(k|t) - x_{ss}(t),\ \Delta u(k|t) = u(k|t) - u(k-1|t).$$

Note that input tracking is replaced by input rate weighting. This is due to the complexity of the nonlinear model, which hinders the online computation of steady state inputs.

The Lancia Delta model was obtained with data gathered during normal driving on public roads, hence we assess our algorithm's capabilities against cornering manoeuvres found in the validation portion of our data set. Panels (a) and (f) of Figure 4 present the test cornering manoeuvres and their optimised Bezier curves, running through the centre line of the driving lane. These will be referred to as Manoeuvres I and II henceforth. These were driven with a road target velocity of $40\,\mathrm{km/h}$ and $60\,\mathrm{km/h}$ respectively, employing $\bar{\alpha} = 2\pi$ and $v_g = 10\,\mathrm{km/h}$ as modulating parameters. These targets

Table IV
STEP 1 PARAMETERS (RHNOCP) FOR THE LANCIA DELTA AND MEGANE.

| Parameter | Value | Parameter | Value |
|---|---|---|---|
| Cost function | | | |
| $N_g/N_c$ | $18/3$ | $Q_y$ | 10 |
| $Q_{\dot{y}}$ | 0.1 | $Q_{\dot{s}}$ | 10 |
| IPOPT main parameters | | | |
| Tolerance | 0.1 | Linear solver | *mumps* |
| Maximum iterations | 5 | Memory update | *bfgs*-6 |

Table V
STEP 2 PARAMETERS (MPC/NMPC) FOR THE LANCIA DELTA.

| Parameter | Value | Parameter | Value |
|---|---|---|---|
| Constraints | | | |
| $V_{LB}$ | $-2\,\mathrm{m/s}$ | $V_{UB}$ | $27.77\,\mathrm{m/s}$ |
| $\omega_{LB}$ | $-\pi\mathrm{rad/s}$ | $\omega_{UB}$ | $\pi\mathrm{rad/s}$ |
| $TP_{LB}$ | $-40$ | $TP_{UB}$ | 40 |
| $\delta_{LB}$ | $-3\pi\mathrm{rad}$ | $\delta_{UB}$ | $3\pi\mathrm{rad}$ |
| Gains | | | |
| $K_f$ | $\begin{bmatrix}-0.26 & 0\\ 0 & 6.82\end{bmatrix}$ | – | – |
| Cost function | | | |
| $Q$ | $\begin{bmatrix}1 & 0\\ 0 & 500\end{bmatrix}$ | $R$ | $\begin{bmatrix}0.01 & 0\\ 0 & 0.1\end{bmatrix}$ |
| $P$ | $\begin{bmatrix}250.42 & 0\\ 0 & 50592.56\end{bmatrix}$ | $N$ | 10 |
| Fast MPC parameters [31] | | | |
| $n_{iter}$ | 5 | $\lambda$ | 0.1 |

and modulation parameters are chosen given the understeering characteristics of the Lancia Delta.

The parameters associated to step 1 (reference generation via the RHNOCP) can be found in Table IV. For Step 2 we consider box constraint for states and inputs with $V_{LB} \leq V(t) \leq V_{UB}$ and analogous bounds for the remaining variables. Table V lists the parameters chosen for both controllers. Note also that we consider a negative lower bound on velocity, however this is only to accommodate for theoretical requirements of MPC that require the origin to be in the interior of the constraint sets. We do not expect the vehicle to go in reverse.

The control objectives are to stay in the middle of the lane and travel at a certain speed, hence performance is assessed with respect to maximum lateral deviation from the lane's center line, defined by $\bar{y}$, and the tracking cost defined as

$$\mathcal{C} = \sum_{t=0}^{T_f} \hat{x}(t)^T Q \hat{x}(t).$$

Table VI summarizes the results. The difference in $C$ across the manoeuvres is at most 4%, indicating that the linear data-driven MPC performs comparably to the NMPC at tracking the velocity, despite the model mismatch. Furthermore, the MPC is able to keep the car 40% closer to the lane's center line when compared to the NMPC (c.f. $\bar{y}$). We conclude from these simulations that the performance of the linear data-driven MPC is within the neighbourhood of the NMPC performance, and thus provides an alternative to nonlinear autonomous driving solutions.

Table VI
PERFORMANCE COMPARISON (MPC/NMPC) FOR THE LANCIA DELTA.

| | Manoeuvre I | | | Manoeuvre II | | |
|---|---|---|---|---|---|---|
| | $\mathcal{C}$ | $\bar{y}\,[m]$ | $T_f\,[s]$ | $\mathcal{C}$ | $\bar{y}\,[m]$ | $T_f\,[s]$ |
| MPC | 576.5 | 0.07 | 219.45 | 3817.8 | 0.29 | 142.05 |
| NMPC | 558.4 | 0.15 | 216.65 | 3802.3 | 0.48 | 140.55 |

Table VII
STEP 2 PARAMETERS (TMPC) FOR THE LANCIA DELTA.

| Parameter | Value | Parameter | Value |
|---|---|---|---|
| Disturbances $\mathbb{W}$ | | | |
| $w_{V,LB}$ | $-0.20\,\mathrm{m/s}$ | $w_{V,UB}$ | $0.20\,\mathrm{m/s}$ |
| $w_{\omega,LB}$ | $-0.15\mathrm{rad/s}$ | $w_{\omega,UB}$ | $0.15\mathrm{rad/s}$ |
| Gains and cost function | | | |
| $K_f$ | $\begin{bmatrix}-0.26 & 0\\ 0 & 6.82\end{bmatrix}$ | $Q$ | $\begin{bmatrix}0.1 & 0\\ 0 & 500\end{bmatrix}$ |
| $P$ | $\begin{bmatrix}25.20 & 0\\ 0 & 50592.56\end{bmatrix}$ | $N$ | 40 |

### B. Disturbance rejection (robustness)

We now demonstrate the capabilities of our proposed path-following algorithm with respect to disturbance rejection. To do so we perform simulations in which the feedback is generated by the linear data-driven model solved over the sampling period. We do this only on the Lancia Delta model for brevity.

As evidenced by the constraint tightening in Assumption 1, the RPI set $\mathbb{S}$ takes away some autonomy from the MPC and allocates it to the linear feedback gain $K_T$. It is usual then to try to find the smallest possible RPI set given a disturbance set $\mathbb{W}$ [30]. Finding the minimal RPI set is usually computationally expensive and seldom provides a sensible result, however given the simplicity of our model (two single input single output models stacked) we can easily find said RPI set for any pair $K_T$ and $\mathbb{W}$. Note that it depends on $K_T$ whether $\mathbb{Z}$ and $\mathbb{V}$ are nonempty. Table VII lists the new/updated parameters chosen for the TMPC (the rest can be found in Table V). The bounds on the acceptable disturbance are larger than the quantified model error in order to also allow for some external perturbations in the built-in robustness of TMPC.

We first discuss the advantages of built-in robustness as opposed to relying on inherent robustness. Figure 4(a)–(e) presents a comparison in performance of the proposed controller for when Step 2 is executed by vanilla MPC (inherent robustness) and TMPC (built-in robustness). Note that by vanilla MPC we refer to a standard non-structurally robust MPC implementation, which does not tighten constraints nor allocates part of the control authority for a corrective action. The parameters for this standard MPC controller are, where applicable, as depicted in Table VII. For this simulation Manoeuvre I was driven with a road target velocity of $100\,\mathrm{km/h}$ without modulation, which amounts to drive at the velocity constraint (see Table IV).

Panels (b) and (c) show the lateral deviation from the centre line obtained with MPC and TMPC (respectively) for 100 independent and random realisations of a disturbance sequence (the same 100 realisations are fed to the algorithm with MPC and TMPC for fairness of comparison). The width of the vehicle is $1.8\,\mathrm{m}$ and the estimated lane width is $3.5\,\mathrm{m}$; it follows

then that both controllers are able to keep the vehicle within acceptable distance from the centreline with deviations of less than $0.4\,\mathrm{m}$ in any direction. When compared to TMPC, MPC is able to complete the manoeuvre in shortest time (about $14\,\mathrm{s}$ or $24\%$ faster) and also reduce the lateral deviation by about $15\,\mathrm{cm}$. This is because MPC does not need to allocate control authority to a linear gain, which ultimately modulates the control action for disturbance rejection purposes, but also because TMPC needs to restrict the reachable states (constraint tightening) thereby restricting the maximum achievable velocity to be $98.6\,\mathrm{km/h}$ in this case. However, since MPC does not explicitly account for the disturbance, and the target is to drive exactly at the velocity constraint, the vehicle's velocity does violate the imposed constraint in several occasions when controlled by vanilla MPC, as depicted in the histogram in panel (d). Nevertheless, given the nature of the disturbance and the move-blocking MPC technique which provides smoother references, the constraint is violated in only $0.7\%$ of the sampling times, and by a maximum of $0.2\,\mathrm{m/s}$ or $0.7\%$. Finally, panel (e) shows the first 10 TMPC trajectories plotted against the map, which confirms the capability of TMPC to follow the reference generated in step 1 and keep the vehicle close to the lane's centre line.

Given the results in panels (b)–(d), built-in robustness might seem an unnecessary trade-off in the form of removing control authority from the MPC controller, but the next set of results, depicted in Figure 4(f)–(i), show that this is not the case. Manoeuvre II, shown in Panel (f), represents a sharper corner when compared to Manoeuvre I, and thus it requires a much lower velocity to be driven safely. Manoeuvre II was driven with a target velocity of $25\,\mathrm{km/h}$ without modulation, and a single realisation of the disturbance which was set to $w(t) = [0.2\;0]^\top$, that is a fixed maximum allowable disturbance in velocity and zero disturbance in yaw rate. This type of disturbance could represent an unexpected road slope, or an unaccounted actuator response (model uncertainty). Panel (g) shows the target velocity and the velocity profiles obtained by MPC and TMPC. Both controllers converge to a value with a steady state error, however MPC relies solely on inherent robustness and hence it converges to $14.7\,\mathrm{m/s}$ or $52.9\,\mathrm{km/h}$, more than three times the true target velocity. TMPC, on the other hand, uses the linear gain $K_T$ to actively correct for the disturbance and hence converges with a steady state error of $5.71\%$. This, in turn, means that MPC is not able to complete the cornering manoeuvre safely, as evidenced by the lateral deviation of almost $4\,\mathrm{m}$ depicted in panel (h). On the other hand, TMPC completes the manoeuvre with a maximum lateral deviation of $10.29\,\mathrm{cm}$.

The results depicted in Figure 4 show that in many situations the inherent robustness of MPC might be enough to provide satisfactory constraint handling and better performance than TMPC due to the retention of full control authority. However, there exist scenarios in which the lack of built-in robustness could result in lane-departure and possible collision. In view of this, we proceed with TMPC for the remaining analysis. Note that similar results are obtained for when feedback is realized with the NLPM (allowing for reduced target velocities and modulation), however are omitted here for brevity.

**Remark 3.** The source of the high frequency lateral vibration shown in panels (b) and (c) of Figure 4 is mainly due to the persistent perturbation we include in this simulation. The latter is not necessarily representative of real driving conditions, thus we defer passenger comfort analysis for Section VI, where a high fidelity suite is employed for simulating the vehicle dynamics (providing a more realistic model uncertainty setup).

### C. Computational performance

Control systems designed for autonomous driving must be able to produce control actions at sufficient sampling rates using the available hardware. Our controller, as depicted in Section III, generates a new control action at each time instant via the solution of two optimisation problems, the RHNOCP solved via IPOPT, and the TMPC solved by means of the fast MPC algorithm, henceforth referred to as FMPC. IPOPT and FMPC have been chosen due to the array of algorithmic features they enjoy, which make them computationally more efficient when compared to other solvers [31], [36], nevertheless the real-time feasibility of any algorithm is contingent to the hardware on which it will be implemented. To give an approximate estimate of computational performance on a realistic embedded system, such as would be used in a driverless car, we evaluated a performance benchmark scaled to the NVIDIA Jetson TX2 [51] low power embedded processing board, which uses the same Tegra X2 chipset as the NVIDIA Drive PX2 [52] made specifically for the driverless car market.

We do not attempt to claim, at this stage, that our control architecture is real-time capable, since it is out of the scope of this paper to deploy the algorithm in target hardware akin to that employed in autonomous driving applications. Instead, we perform a computational performance analysis in line with the original message in [31], [36]; that is, to study the effect of the algorithmic improvements that IPOPT and FMPC have over alternative, possibly naive, solutions.

We study the processing time of our algorithm under the following conditions:

i. Step 1 solved via:
   - The MATLAB function *fminsearch*, which implements a direct search method.
   - The IPOPT algorithm as obtained from [37].

ii. Step 2 solved via:
   - The MATLAB function *quadprog* which is optimised to solve optimisation problems such as (9)–(10).
   - The FMPC algorithm as obtained from [53].

The *quadprog* function is the go-to option for solving quadratic programming problems in MATLAB, although it is not necessarily optimised for real-time capabilities. The FMPC algorithm [53] has several limitations when compared to *quadprog*, being a key one that an arbitrary terminal set is not allowed, that is $\mathbb{Z}_f\left(z_{ss}, v_{ss}\right) = \mathbb{Z} \ominus \{z_{ss}\}$. Nevertheless, it has been shown [20] that terminal constraints can be replaced by a long enough prediction horizon, which we assume to be our case with $N = 40$. Finally, we fed the solution at the previous sampling time as warm start for all solvers.

All the processing time results were obtained by running our proposed control algorithm in MATLAB R2017b, on an

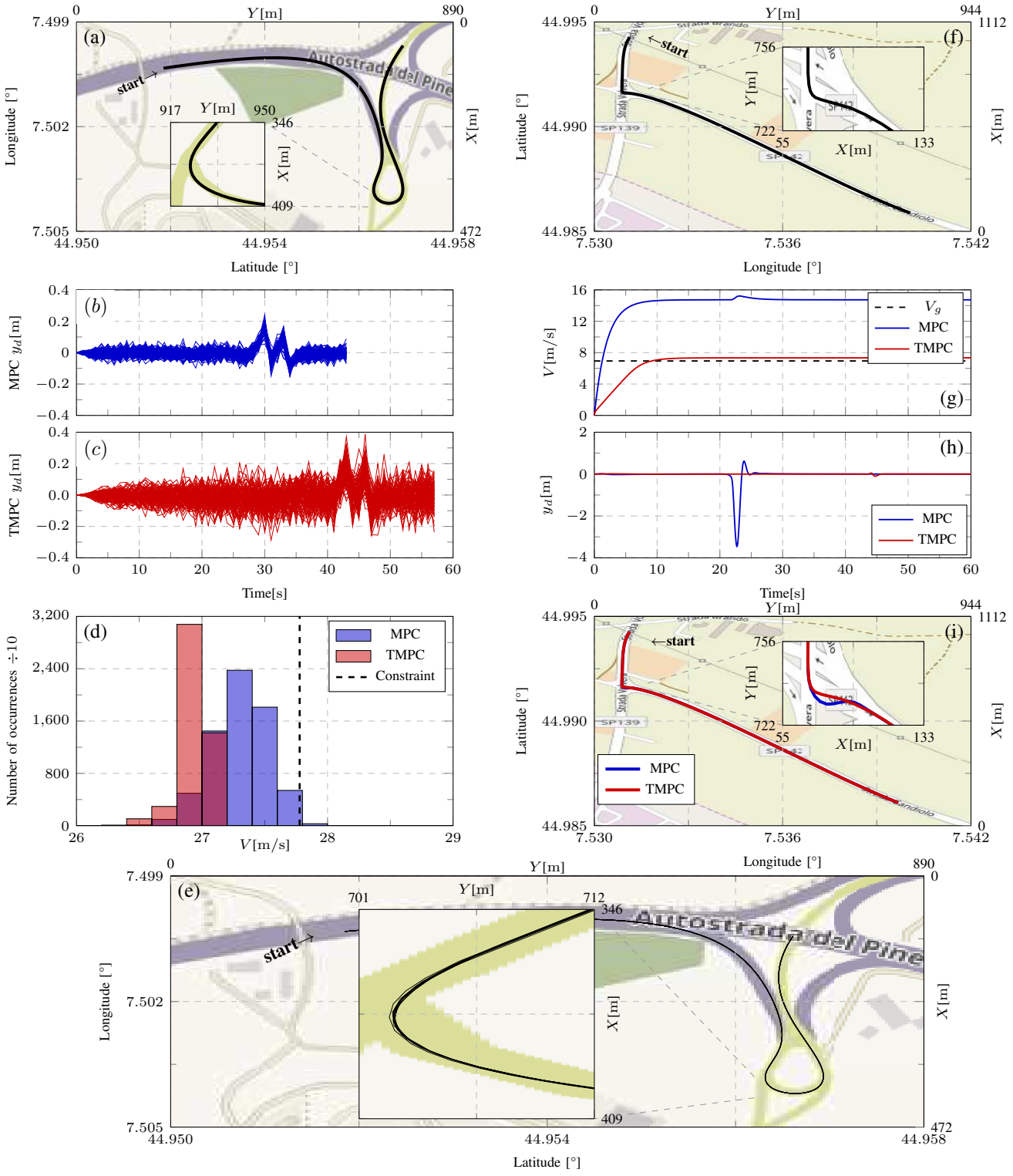


Figure 4. (a)–(e)→Cornering Manoeuvre I (Lancia): (a) Map and Bezier curve optimised to follow the centre of the rightmost lane, (b) Lateral deviation from the centre line of the vehicle with MPC in Step 2, (c) Lateral deviation from the centre line of the vehicle with TMPC in Step 2, (d) Histogram of the velocity $V$ measurement, (e) Map and trajectories of several TMPC simulations. (f)–(i)→Cornering Manoeuvre II: (f) Map and Bezier curve optimised to follow the centre of the rightmost lane, (g) Unmodulated target velocity and true vehicle velocity for MPC and TMPC, (h) Lateral deviation from the centre line of the vehicle with MPC and TMPC, (i) Map and trajectories of MPC and TMPC simulations

Intel(R) Core(TM) i7-4700MQ CPU @ $2.40\,\mathrm{GHz}$ processor, fitted with L1, L2 and L3 cache memories of 0.256, 1 and 6 MB respectively. An approximate scaling is then performed to estimate processing times on the Jetson TX2 based on the results of running the Netlib Whetstone Benchmark [54] on the desktop computer, and Jetson TX2.

We assessed the performance of our controller with a simulation of Manoeuvre I under a single realisation of the disturbance. Different prediction horizons were employed for the RHNOCP, however *fminsearch* was able to find a stabilizing solution only when a single block was employed, hence an additional simulation is performed for the pair IPOPT–FMPC with multiple blocks. Table VIII summarise the computational performance of our algorithm in terms of mean processing time, its standard deviation, and maximum processing time (all as a percentage of the sampling time). As expected, increasing the prediction horizon of the RHNOCP (from 10 to 18) increases the time needed to obtain a reference, independent of the strategy used to solve the optimisation. Moreover, although small, there exist variations in the time taken to solve Step 2 for different horizons of the RHNOCP (either via *quadprog* or FMPC). The reason for these is that different horizons in the RHNOCP will unavoidably result in different references, hence modifying the TMPC optimisation problem.

The average and maximum time used to solve Step 1 increases by a factor of 1.3 when IPOPT is implemented, however using a larger number of blocks does not necessarily result in further increments. Indeed, solving the RHNOCP via IPOPT with $(N_g, N_c) = (18, 1)$ takes a maximum of 40% of the sampling time (and an average of 21%), however when 3 blocks are used the RHNOCP is solved in a maximum of 34% of the sampling time (and an average of 28%). Oppositely, solving Step 2 with FMPC results in a reduction of an order of magnitude in the average and maximum processing times, when compared to the usage of *quadprog*.

The bottleneck seems to be solving the RHNOCP (Step 1), which can take between 31% and 40% of the sampling time depending on the horizon length and number of blocks. However, the implementation of IPOPT allows for the usage of more blocks in the solution of the RHNOCP, resulting in a references with better characteristics (as observed in the comfort analysis). Furthermore, the reduction in processing time obtained via the implementation of FMPC is able to heavily counter the increased load of IPOPT, resulting in overall faster computations for the IPOPT-FMPC pair, even when the horizon of the RHNOCP is split into more blocks.

In summary, the algorithmic improvements provided by the IPOPT–FMPC pair result in maximum processing times of 41% of the sampling time, a decrease of 17% when compared to *fminsearch–quadprog*. These results are well below the 100% sampling time threshold imposed by some researchers [34], and in line with similar approaches such as [19] which reported maximum processing times of 42% in experiments, and [15], [55] which in simulation achieved average processing times of 50% and 28% (respectively).

Furthermore, note that to achieve these processing capabilities [19], [55] resort to LTV models and LTV-MPC algorithms with relatively short prediction horizons (3 and 10 respectively). Nonlinear MPC (based on a nonlinear physical model) is solved in [15] with an off-the-shelf tool, however reporting relatively long processing times (50% of the sampling time in average) and in a race-car driving setup in which deviations of over $5\,\mathrm{m}$ from the centre line seem to be acceptable. Our approach, on the other hand, is able to leverage on the mathematical simplicity of the uncoupled data driven model to achieve processing times under 30% of the sampling time (in average) and to provide safe driving in public roads.

## VI. CarMaker simulation

The results in Section V show the ability of our proposed controller to safely steer a vehicle throughout cornering manoeuvres for a perfect prediction model. In order to analyse the performance of our control architecture in presence of modelling uncertainty we employ the high fidelity simulator IPGCarMaker. Note, however, that the IPGCarMaker software does not allow to define post-gearbox torque externally (i.e. as a control input), since it would require bypassing several engine modules for axle torque computation. In view of this we map our input $TP$ to gas and brake pedals as follows

$$(G(t), B(t)) = \begin{cases} (TP(t)/TP_{UB}, 0) & , \quad TP(t) \geq 0 \\ (0, TP(t)/TP_{LB}) & , \quad TP(t) < 0. \end{cases}$$

This mapping neglects engine and brake system dynamics and hence introduces uncertainty. A possible solution to this issue is to use Hammerstein-Wiener models to find a (static) nonlinear map between the pedals and the chosen inputs, however this remains as future work. For the following results we allow for increased uncertainty bounds to account for this.

Table IX shows all the controller parameters that changed from the Lancia simulation (see Tables IV and VII). Figure 5 shows the results of driving the Megane through Manoeuvre I with a road target velocity $V_g$ of $80\,\mathrm{km/h}$ and modulating velocities $v_g$ of $30\,\mathrm{km/h}$ and $50\,\mathrm{km/h}$. Panels (a) and (b) show how the modulated target velocity $V_f$ (green-dotted line) changes depending on the modulating velocity $v_g$ and the road curvature it can be seen that the reference velocity is tracked accurately while it experiences slow changes (for example in the interval from initialisation until $30\,\mathrm{s}$), despite the modelling error described in II.

Panel (c) shows the lateral deviation from the centreline for both modulating velocities. A larger modulating velocity results in larger target velocities throughout the manoeuvre, thus in the vehicle being driven faster. This, in turn, results in a larger lateral deviation (as depicted in panel (c)), particularly when the vehicle is in the middle of the roundabout. Nevertheless the maximum deviation remains under acceptable values given an estimated lane width of $3.5\,\mathrm{m}$ and a vehicle width of $1.8\,\mathrm{m}$. Finally, panel (d) shows both trajectories plotted against the map. The maximum difference in lateral deviation with respect to the $s$ coordinate (i.e. when passing through the same patch of track) is under $30\,\mathrm{cm}$, which explain the apparent superimposition of the trajectories in view of the map resolution.

Table VIII
SCALED PROCESSING TIME (MANOEUVRE I - LANCIA DELTA - PERTURBED). COMPARISON BETWEEN *fminsearch-quadprog* AND IPOPT-FMPC.

| Solver | *fminsearch-quadprog* | | IPOPT-FMPC | | IPOPT-FMPC | |
|---|---|---|---|---|---|---|
| Horizons $N_g - N_c$ | $18-1$ | $10-1$ | $18-1$ | $10-1$ | $18-3$ | $18-2$ |
| | Maximum(mean)[standard deviation $\sigma$] | | | | | |
| Step 0 | 4( 2)[1] | 4( 2)[1] | 3( 2)[0] | 4( 2)[0] | 4( 2)[0] | 4( 2)[0] |
| Step 1 | 32(13)[3] | 25(10)[2] | 40(17)[3] | 31(12)[2] | 34(24)[3] | 34(16)[4] |
| Step 2 | 24(15)[4] | 30(16)[4] | 4( 2)[1] | 5( 2)[1] | 4( 2)[1] | 4( 2)[1] |
| Total | 55(31)[6] | 52(28)[6] | 44(21)[3] | 35(16)[2] | 37(28)[3] | 37(20)[4] |

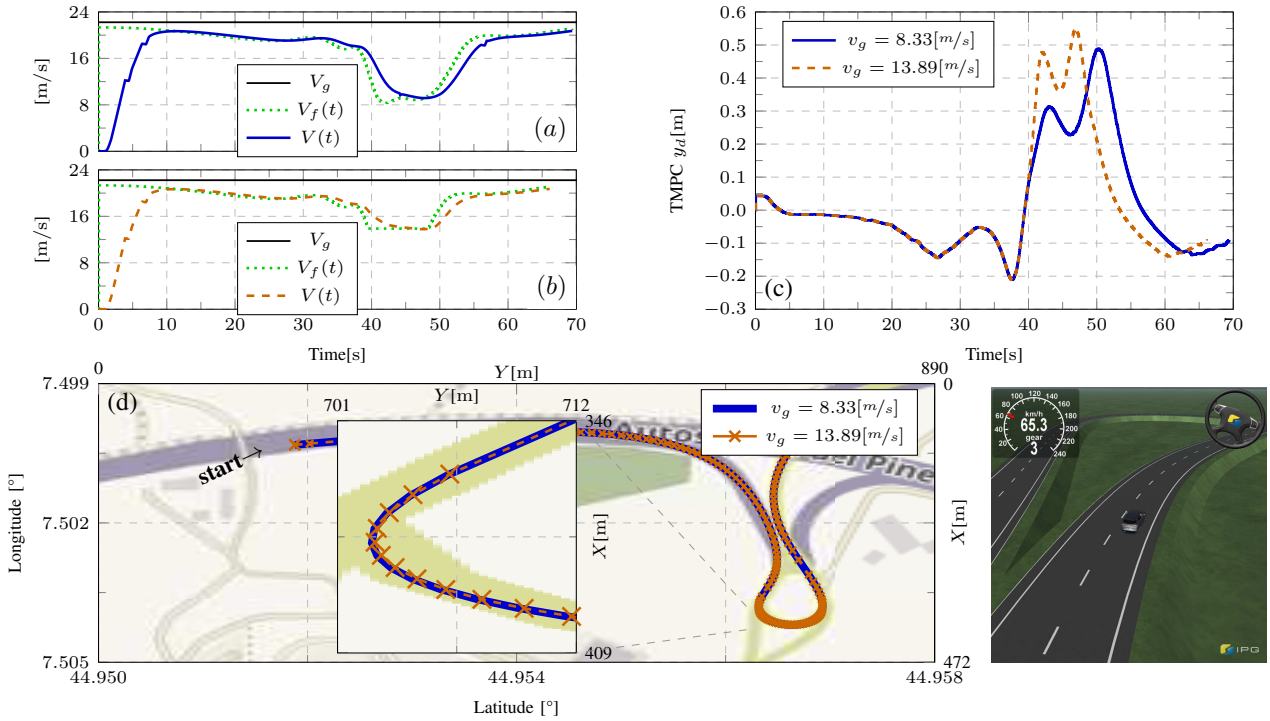


Figure 5. CarMaker simulation of cornering Manoeuvre I (Megane): (a) Velocity profile for $v_g = 8.33\,\mathrm{m/s}$, (b) Velocity profile for $v_g = 13.89\,\mathrm{m/s}$, (c) Lateral deviation from the lane centre line, (d) Map and trajectories of the IPGCarMaker simulation.

Table IX
CONTROLLER PARAMETERS FOR THE MEGANE.

| Parameter | Value | Parameter | Value |
|---|---|---|---|
| | Constraints | | |
| $TP_{LB}$ | $-80$ | $TP_{UB}$ | 80 |
| $w_{V,LB}$ | $-0.23\,\mathrm{m/s}$ | $w_{V,UB}$ | $0.23\,\mathrm{m/s}$ |
| $w_{\omega,LB}$ | $-0.45\mathrm{rad/s}$ | $w_{\omega,UB}$ | $0.45\mathrm{rad/s}$ |
| | Gains | | |
| $K_T$ | $\begin{bmatrix} -96.80 & 0 \\ 0 & -0.20 \end{bmatrix}$ | $K_f$ | $\begin{bmatrix} -0.26 & 0 \\ 0 & 6.50 \end{bmatrix}$ |
| | Cost function | | |
| $P$ | $\begin{bmatrix} 25.20 & 0 \\ 0 & 50549.12 \end{bmatrix}$ | $N$ | 40 |

### A. Passenger comfort analysis

To assess the performance of our controller in terms of passenger comfort we use measures of local and average comfort. In particular, we observe maximum lateral acceleration $\bar{a}_y$, maximum lateral jerk $\bar{J}_y$ and the frequency content of the lateral acceleration within certain ranges, measured via the power spectrum $PSD_{a_y}$, which measure local and average comfort [56]. Table X shows the acceptable limits of these indicators, alongside their values for Manoeuvre I simulated with $v_g = 8.33\,\mathrm{m/s}$. It is clear from Table X that, although no specific comfort requirement has been included in the design of our two-step controller, the instantaneous measure of comfort related to the lateral acceleration is met, however, the maximum lateral jerk is an order of magnitude over the comfort limit.

The source of this excessive jerk is twofold, the first being the corrective action of the TMPC. The second contributor is the constant update of the reference yaw rate by the RHNOCP, which is necessary to account for the modelling dichotomy that exists between the two steps of our controller. In order to improve the comfort performance of our controller we include a filter to attenuate the high frequency content present in the reference generated by the RHNOCP. We implement a first order low pass filter with cutoff frequency of $1.5\,\mathrm{Hz}$. Table X shows that this does result in a decrease of the maximum lateral jerk, however at the expense of worsened average comfort (as measured by the acceleration frequency content).

Given that the main source of the large jerk is the RHNOCP optimisation, we also perform a simulation with a reduced

Table X
PASSENGER COMFORT COMPARISON BETWEEN FILTERED AND UNFILTERED REFERENCE FOR MANOEUVRE I WITH $v_g = 8.33\,\mathrm{m/s}$ (MEGANE)

| Indicator | Limit | Unfiltered | Filtered | |
|---|---|---|---|---|
| | | | $N_c = 3$ | $N_c = 2$ |
| $\bar{a}_y$ $\mathrm{m/s^2}$ | $\leq 2$ | 0.77 | 0.93 | 0.71 |
| $\bar{J}_y$ $\mathrm{m/s^3}$ | $\leq 0.9$ | 13.34 | 12.46 | 4.42 |
| $\int PSD_{a_y}$ 2–3Hz($\times 10^2$) | – | 2.97 | 3.49 | 2.79 |
| $\int PSD_{a_y}$ 4–5Hz($\times 10^2$) | – | 1.29 | 1.02 | 0.99 |

amount of blocks in the move-blocking MPC strategy, in order to reduce the frequency content of the optimal reference passed on to the TMPC. This results, as portrayed in Table X, in a considerable reduction of the maximum jerk, and a slight improvement in all the other comfort measures. Although the maximum jerk is still over the recommended limit, this threshold is surpassed only during 14% of the total manoeuvre duration (around $9\,\mathrm{s}$ in aggregate), and the mean jerk is $0.57\,\mathrm{m/s^3}$, a reduction of 2.3 times when compared to the unfiltered manoeuvre.

## VII. CONCLUSIONS

In this paper we have proposed a two-step robust MPC controller for the purpose of path-following. The key advantage of our controller is the use of uncoupled LTI data-driven models for the prediction of lateral and longitudinal dynamics, which show good fit to data and small one-step ahead prediction error. These characteristics allow for a good control performance with linear MPC with built-in robustness, tested against a high-fidelity simulator in IPGCarMaker. Furthermore, the implementation of linear MPC allows for computationally efficient algorithms for solving the online optimisation problem, achieving processing times which show their advantage over naive solvers, and could provide an advantageous baseline for future implementation in target hardware.

## ACKNOWLEDGMENT

The authors would like to thank the EU for funding support through grant number 731593 (Dreams4Cars), as well as A. Saroldi and Centro Ricerche Fiat for their contribution in collecting the experimental data.

## APPENDIX

### A. Uncoupled linear models employed for predictions.

| | | $V \leftarrow (rT - P, \phi)$ | $\omega \leftarrow \delta$ |
|---|---|---|---|
| Lancia | $A_d$ | 0.9996 | 0.7116 |
| | $B_d$ | 0.0061 | 0.0415 |
| Megane | $A_d$ | 0.9994 | 0.5703 |
| | $B_d$ | 0.0052 | 0.0653 |

## REFERENCES

[1] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Z. Kolter, D. Langer, O. Pink, V. Pratt, M. Sokolsky, G. Stanek, D. Stavens, A. Teichman, M. Werling, and S. Thrun, "Towards Fully Autonomous Driving: Systems and Algorithms," in *Proceedings of the 2011 IEEE Intelligent Vehicles Symposium*. Baden, Germany: IEEE, 2011, pp. 163–168.

[2] M. Doumiati, A. Victorino, A. Charara, and D. Lechner, "Estimation of vehicle lateral tire-road forces : a comparison between extended and unscented Kalman filtering," in *Proceedings of the 2009 European Control Conference*, 2009, pp. 4804–4809.

[3] M. M. Shirazi and A. B. Rad, "L_1 Adaptive Control of Vehicle Lateral Dynamics," *IEEE Transactions on Intelligent Vehicles*, vol. 3, no. 1, pp. 92–101, 2018.

[4] E. H. M. Lim and J. K. Hedrick, "for Automated Vehicle Operation," in *Proceedings of the 1999 IEEE American Control Conference*, 1999, pp. 3676–3680.

[5] M. H. Lee, K. S. Lee, H. G. Park, Y. C. Cha, D. J. Kim, B. Kim, S. Hong, and H. H. Chun, "Lateral controller design for an unmanned vehicle via kalman filtering," *International Journal of Automotive Technology*, vol. 13, no. 5, pp. 801–807, 2012.

[6] G. Tagne, R. Talj, and A. Charara, "Design and Comparison of Robust Nonlinear Controllers for the Lateral Dynamics of Intelligent Vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 3, pp. 796–809, 2016.

[7] C. Acosta Lúa and S. Di Gennaro, "Nonlinear adaptive tracking for ground vehicles in the presence of lateral wind disturbance and parameter variations," *Journal of the Franklin Institute*, vol. 354, no. 7, pp. 2742–2768, 2017.

[8] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, pp. 789–814, 2000.

[9] P. F. Lima, G. Collares Pereira, J. Mårtensson, and B. Wahlberg, "Experimental validation of model predictive control stability for autonomous driving," *Control Engineering Practice*, vol. 81, pp. 244–255, 2018.

[10] U. Rosolia, S. De Bruyne, and A. G. Alleyne, "Autonomous Vehicle Control: A Nonconvex Approach for Obstacle Avoidance," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 2, pp. 469–484, 2017.

[11] Y. Gao, A. Gray, J. V. Frasch, T. Lin, E. Tseng, J. K. Hedrick, and F. Borrelli, "Spatial Predictive Control for Agile Semi-Autonomous Ground Vehicles," in *Proceedings of the 11th International Symposium on Advanced Vehicle Control*, 2012, pp. 1–6.

[12] C. Liu, S. Lee, S. Varnhagen, and H. E. Tseng, "Path planning for autonomous vehicles using model predictive control," in *Proceedings of the 2017 IEEE Intelligent Vehicles Symposium*. IEEE, 2017, pp. 174–179.

[13] Z. Sun, L. Dai, K. Liu, Y. Xia, and K. H. Johansson, "Robust MPC for tracking constrained unicycle robots with additive disturbances," *Automatica*, vol. 90, pp. 172–184, 2018.

[14] P. Falcone, F. Borrelli, J. Asgari, H. E. Tseng, and D. Hrovat, "Predictive Active Steering Control for Autonomous Vehicle Systems," *IEEE Transactions on Control Systems Technology*, vol. 15, no. 3, pp. 566–580, 2007.

[15] A. Buyval, A. Gabdulin, R. Mustafin, and I. Shimchik, "Deriving overtaking strategy from nonlinear model predictive control for a race car," in *Proceedings of the 2017 IEEE International Conference on Intelligent Robots and Systems*, 2017, pp. 2623–2628.

[16] S. Arrigoni, F. Braghin, and F. Cheli, "Decentralized fast-MPC path planner for automated vehicles," in *Proceedings of the 2017 IEEE International Conference of Electrical and Electronic Technologies for Automotive*, 2017, pp. 1–7.

[17] P. Falcone, F. Borrelli, H. E. Tseng, J. Asgari, and D. Hrovat, "Linear time-varying model predictive control and its application to active steering systems: Stability analysis and experimental validation," *International Journal of Robust and Nonlinear Control*, vol. 18, no. 8, pp. 862–875, 2008.

[18] C. E. Beal and J. C. Gerdes, "Model predictive control for vehicle stabilization at the limits of handling," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 4, pp. 1258–1269, 2013.

[19] M. Jalali, S. Khosravani, A. Khajepour, S.-k. Chen, and B. Litkouhi, "Model predictive control of vehicle stability using coordinated active steering and differential brakes," *Mechatronics*, vol. 48, pp. 30–41, 2017.

[20] L. Grüne and J. Pannek, *Nonlinear Model Predictive Control: Theory and Algorithms*, 1st ed. Springer-Verlag London, 2011.

[21] J. Liu, P. Jayakumar, J. L. Stein, and T. Ersal, "A study on model fidelity for model predictive control-based obstacle avoidance in high-speed autonomous ground vehicles," *Vehicle System Dynamics*, vol. 54, no. 11, pp. 1629–1650, 2016.

[22] Y. Gao, A. Gray, H. E. Tseng, and F. Borrelli, "A tube-based robust nonlinear predictive control approach to semiautonomous ground vehicles," *Vehicle System Dynamics*, vol. 52, no. 6, pp. 802–823, 2014.

[23] E. Alcala, V. Puig, J. Quevedo, and O. Sename, "Fast Zonotope-Tube-based LPV-MPC for Autonomous Vehicles," *arXiv e-prints*, p. arXiv:2009.02248, 2020.

[24] S. A. Eker and M. Nikolaou, "Linear control of nonlinear systems: Interplay between nonlinearity and feedback," *AIChE Journal*, vol. 48, no. 9, pp. 1957–1980, 2002.

[25] S. M. Yoon, K. S. Lee, S. Y. Kim, J. H. Kang, and M. H. Lee, "Lateral Control of an UCT (Unmanned Container Transporter) Using Ultrasonic Satellite System and System Identification," in *Proceedings of the 2008 International Conference on Control, Automation and Systems*. Seoul: IEEE, 2008, pp. 296–300.

[26] N. Liu and A. G. Alleyne, "Iterative learning identification applied to automated off-highway vehicle," *IEEE Transactions on Control Systems Technology*, vol. 22, no. 1, pp. 331–337, 2014.

[27] M. Canale, L. Fagiano, and M. C. Signorile, "Nonlinear model predictive control from data: a set membership approach," *International Journal of Robust and Nonlinear Control*, vol. 24, pp. 123–139, 2014.

[28] M. Gevers, "Identification for control: From the early achievements to the revival of experiment design*," *European Journal of Control*, vol. 11, no. 4, pp. 335–352, 2005.

[29] B. A. Hernandez Vicente, S. James, and S. R. Anderson, "Linear System Identification Versus Physical Modeling of Lateral-Longitudinal Vehicle Dynamics," *IEEE Transactions on Control Systems Technology*, 2020.

[30] D. Q. Mayne, M. M. Seron, and S. V. Raković, "Robust model predictive control of constrained linear systems with bounded disturbances," *Automatica*, vol. 41, no. 2, pp. 219–224, 2005.

[31] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," *IEEE Transactions on Control Systems Technology*, vol. 18, no. 2, pp. 267–278, 2010.

[32] IPG Automotive, "CarMaker: Virtual testing of automobiles and light-duty vehicles." [Online]. Available: https://ipg-automotive.com/products-services/simulation-software/carmaker/

[33] L. Ljung, *System Identification Theory for the user*, 2nd ed. Upper Saddle River, NJ: Prentice Hall, 1999.

[34] K. Berntorp and F. Magnusson, "Hierarchical predictive control for ground-vehicle maneuvering," in *Proceedings of the 2015 American Control Conference*. IEEE, 2015, pp. 2771–2776.

[35] R. C. Shekhar and J. M. MacIejowski, "Robust variable horizon MPC with move blocking," *Systems and Control Letters*, vol. 61, no. 4, pp. 587–594, 2012.

[36] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, 2006.

[37] E. Bertolazzi, "mexipopt v1.1.2 [source code]," https://github.com/ebertolazzi/mexIPOPT, 2015–2021.

[38] J. B. Rawlings and D. Q. Mayne, *Model Predictive Control: Theory and Design*, electronic ed. Madison, Wisconsin: Nob Hill, 2014.

[39] I. Kolmanovsky and E. G. Gilbert, "Theory and Computation of Disturbance Invariant Sets for Discrete-Time Linear Systems," *Mathematical Problems in Engineering*, vol. 4, pp. 317 – 367, 1998.

[40] D. Mayne, S. Raković, R. Findeisen, and F. Allgöwer, "Robust output feedback model predictive control of constrained linear systems," *Automatica*, vol. 42, no. 7, pp. 1217 – 1222, 2006.

[41] B. A. Hernandez Vicente and P. A. Trodden, "Stabilizing predictive control with persistence of excitation for constrained linear systems," *Systems and Control Letters*, vol. 126, pp. 58–66, 2019.

[42] P. A. Trodden and J. Maestre, "Distributed predictive control with minimization of mutual disturbances," *Automatica*, vol. 77, pp. 31 – 43, 2017.

[43] D. Simon, J. Lofberg, and T. Glad, "Reference tracking MPC using dynamic terminal set transformation," *IEEE Transactions on Automatic Control*, vol. 59, no. 10, pp. 2790–2795, 2014.

[44] D. Limon, M. Pereira, D. Munoz De La Pena, T. Alamo, C. N. Jones, and M. N. Zeilinger, "MPC for Tracking Periodic References," *IEEE Transactions on Automatic Control*, vol. 61, no. 4, pp. 1123–1128, 2016.

[45] J. Kohler, M. A. Muller, and F. Allgower, "Nonlinear reference tracking: An economic model predictive control perspective," *IEEE Transactions on Automatic Control*, vol. 64, no. 1, pp. 254–269, 2019.

[46] P. Falugi, "Model predictive control for tracking randomly varying references," *International Journal of Control*, vol. 88, no. 4, pp. 745–753, 2015.

[47] P. R. Baldivieso Monasterios and P. A. Trodden, "Model Predictive Control of Linear Systems With Preview Information: Feasibility, Stability, and Inherent Robustness," *IEEE Transactions on Automatic Control*, vol. 64, no. 9, pp. 3831–3838, 2019.

[48] P. O. M. Scokaert, D. Q. Mayne, and J. B. Rawlings, "Suboptimal Model Predictive Control (Feasibility Implies Stability)," *IEEE Transactions on Automatic Control*, vol. 44, no. 3, pp. 648–654, 1999.

[49] G. Pannocchia, J. B. Rawlings, and S. J. Wright, "Is suboptimal nonlinear MPC inherently robust?" in *Proceedings of the 18th IFAC World Congress*. IFAC, 2011, pp. 7981–7986.

[50] D. A. Allan, C. N. Bates, M. J. Risbeck, and J. B. Rawlings, "On the inherent robustness of optimal and suboptimal nonlinear MPC," *Systems and Control Letters*, vol. 106, pp. 68–78, 2017.

[51] NVIDIA, "Jetson TX2 Module," 2022, [Online] Available at: https://developer.nvidia.com/embedded/jetson-tx2 (Accessed 17 June 2022).

[52] ——, "5.x_Linux_DPX_SDK," 2018, [Online] Available at: https://docs.nvidia.com/drive/active/5.0.10.3L/nvvib_docs/ (Accessed 17 June 2022).

[53] Y. Wang and S. Boyd, "Fast model predictive control using online optimization v alpha [source code]," https://web.stanford.edu/~boyd/papers/fast_mpc.html, 2008.

[54] R. Painter, "C Converted Whetstone Double Precision Benchmark." [Online]. Available: http://www.netlib.org/benchmark/whetstone.c

[55] K. Liu, J. Gong, A. Kurt, H. Chen, and U. Ozguner, "Dynamic Modeling and Control of High-Speed Automated Vehicles for Lane Change Maneuver," *IEEE Transactions on Intelligent Vehicles*, vol. 3, no. 3, pp. 329–339, 2018.

[56] J. Eriksson and L. Svensson, "Tuning for Ride Quality in Autonomous Vehicle Application to Linear Quadratic Path Planning Algorithm," Ph.D. dissertation, 2015.


**Bernardo A. Hernandez Vicente** received the M.Eng. degree from the University of Concepcion, Chile, in 2013 and the Ph.D. degree in Model Predictive Control theory from the Department of Automatic Control and Systems Engineering, University of Sheffield, UK in 2018. Since 2021, he is an assistant professor at the Department of Mechanical Engineering, University of Concepcion, Chile.

**Paul A. Trodden** received the MEng degree in Engineering Science from the University of Oxford, UK, in 2003, and the PhD degree in Aerospace Engineering from the University of Bristol, UK, in 2009. Since 2012, he is a lecturer in the Department of Automatic Control and Systems Engineering, University of Sheffield, UK. His research interests include robust and distributed model predictive and optimization-based control, and control and optimization for power and energy systems.

**Sean R. Anderson** received the M.Eng. degree in control systems engineering in 2001 and in 2005 the Ph.D. in nonlinear system identification and predictive control from the Department of Automatic Control and Systems Engineering, University of Sheffield, UK. From 2005 to 2010, he was a Research Associate in the Neural Algorithms Research Group, University of Sheffield. From 2010 to 2011 he was a Research Associate in the Department of Automatic Control and Systems Engineering, University of Sheffield, then became lecturer there in 2012, and has been a senior lecturer since 2015.