Learning How to Autonomously Race a Car: A Predictive Control Approach

Ugo Rosolia and Francesco Borrelli

***Abstract*— We present a learning model predictive controller (LMPC) for autonomous racing. We model the autonomous racing problem as a minimum time iterative control task, where an iteration corresponds to a lap. The system trajectory and input sequence of each lap are stored and used to systematically update the controller for the next lap. In the proposed approach, the race time does not increase at each iteration. The first contribution is to propose a local LMPC which reduces the computational burden associated with existing LMPC strategies. In particular, we show how to construct a local safe set and approximation to the value function, using a subset of the stored data. The second contribution is to present a system identification strategy for the autonomous racing iterative control task. We use data from previous iterations and the vehicle's kinematic equations of motion to build an affine time-varying prediction model. The effectiveness of the proposed strategy is demonstrated by experimental results on the Berkeley Autonomous Race Car (BARC) platform.**

***Index Terms*— Autonomous racing, autonomous vehicles, iterative learning control, model predictive control (MPC), predictive control, real-time optimization, system identification.**

I. Introduction

AUTONOMOUS driving is an active research field. Over the past decades, several techniques have been proposed for different driving scenarios [1]–[9]. Depending on the control task (i.e., highway driving, urban driving, emergency maneuvers), the behavior of the vehicle can be modeled with linear or nonlinear equations of motions [10], [11]. When the nonlinearities of the vehicle are excited, the control task is inevitably more challenging. In this work, we are interested in designing a controller for autonomous racing which can operate the vehicle in the nonlinear regime, close to the limit of the vehicle's handling capability. We formulate the autonomous racing problem as an iterative control task, where at each iteration the controller drives the vehicle around the track trying to minimize the lap time.

Recently, several approaches have been proposed for autonomous racing. In [12], the authors reformulated the autonomous racing control task as a non-convex optimization problem and then proposed a linearization strategy to compute approximate solution. Verschueren *et al.* [13] proposed a nonlinear model predictive control (NMPC) strategy which exploits a Pacejka tire model identified from experimental data. The NMPC is implemented on an experimental setup using an exact Hessian sequential quadratic programm (SQP)-type optimization algorithm. NMPC strategies for autonomous racing are tested also in [14], where the authors compared two control methodologies based on different parametrizations of the vehicle's model. A model predictive contouring control (MPCC) was presented in [15]. In MPCC, the controller objective is a tradeoff between the progress along the track and the contouring error. First, a high-level MPC computes the optimal racing trajectory. Afterward, a low-level controller is used to track the optimal racing line. This strategy is extended in [16] to design a racing controller that guarantees recursive constraint satisfaction. Also in [17], the control problem is divided into two steps. First, a reference trajectory is computed using the method proposed in [18]. Afterward, an iterative learning control (ILC) approach is used for tracking. The authors showed the effectiveness of the proposed approach by experimental testing on a full-size vehicle. We proposed to reformulate the autonomous racing problem as an iterative control task. The controller is not based on a precomputed racing line and it learns from experiencing a trajectory which minimizes the lap time. In particular, the closed-loop trajectories at each lap are stored and used to systematically update the controller for the next lap. This brief builds on [19]–[21] and has two main contributions.

The first contribution is to propose a local learning model predictive controller (LMPC) strategy where the terminal cost and constraint are updated at each time step. In particular, at each time t, we exploit the planned trajectory at time $t-1$ to construct a local terminal cost and constraint. In contrast to our previous works [19]–[21], the terminal cost and constraint are computed using a subset of the stored data; therefore, the proposed local LMPC enables the reduction of the computational burden associated with existing LMPC strategies. The effectiveness of the proposed approach is demonstrated on the Berkeley Autonomous Race Car (BARC)[1] platform. We show that the proposed controller is able to improve the lap time, until it converges to a steady-state behavior. Finally, we analyze the lateral acceleration acting on the closed-loop system and we confirm that the controller learns to drive the vehicle at the limit of its handling capability.

The second contribution of this work is to propose a system identification strategy tailored to the autonomous racing application. We propose to exploit both the kinematic equations of motion and data from previous iterations to identify an affine

Manuscript received July 17, 2019; revised October 4, 2019; accepted October 11, 2019. Manuscript received in final form October 15, 2019. This work was supported in part by the Hyundai Center of Excellence at the University of California, Berkeley, and in part by the Office of Naval Research. Recommended by Associate Editor V. Zavala. *(Corresponding author: Ugo Rosolia.)*

The authors are with the Department of Mechanical Engineering, University of California at Berkeley, Berkeley, CA 94701 USA (e-mail: ugo.rosolia@berkeley.edu; fborrelli@berkeley.edu).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCST.2019.2948135

[1] A video of the experiment can be found at https://youtu.be/ZBFJWtIbtMo

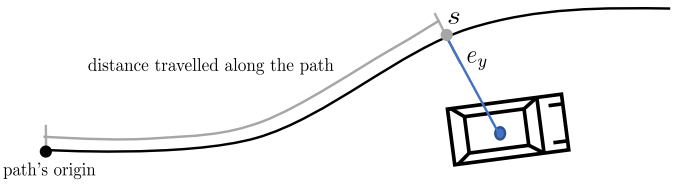

Fig. 1. Representation of the vehicle's position in the curvilinear reference frame.

time-varying (ATV) prediction model used for control. In particular, we use a local linear regressor to learn the relationships between the inputs and the vehicle's velocities. Furthermore, we linearize the kinematic equations of motion to approximate the evolution of the vehicle's position as a function of the velocities. In contrast to our previous works [19], [20], this strategy allows us to reformulate the LMPC as a quadratic program (QP) which can be solved efficiently.

This brief is organized as follows: in Section II, we introduce the problem formulation. Section III illustrates the LMPC design. In particular, it shows how to construct local safe sets and value function approximations using a subset of the collected data. Section IV illustrates the system identification strategy used in the experiments. Finally, in Section V we present the experimental results on the BARC platform. Section VIII provides final remarks.

II. Problem Formulation

Consider the following state and input vectors:

$$x = \left[v_x, v_y, w_z, e_\psi, s, e_y\right]^\top \text{ and } u = \left[\delta, a\ \right]^\top$$

where w_z, v_x, v_y, are the vehicle's yaw rate, longitudinal and lateral velocities. The position of the vehicle is represented in the curvilinear reference frame [22], where s is the distance traveled along the centerline of the track. The states e_ψ and e_y are the heading angle and lateral distance error between the vehicle and the centerline of the track, as shown in Fig. 1. Finally, δ and a are the steering and acceleration commands. The vehicle is described by the dynamic bicycle model

$$x_{t+1} = f(x_t, u_t) \tag{1}$$

where $f(\cdot,\cdot)$ is derived from kinematics and balancing the forces acting on the tires [10]. A detailed expression can be found in [10, Ch. 2]. Note that in the curvilinear reference frame state and input constraints are convex, that is

$$\begin{aligned} x_t &\in \mathcal{X} = \{x \in \mathbb{R}^n : F_x x \leq b_x\} \\ u_t &\in \mathcal{U} = \{u \in \mathbb{R}^d : F_u u \leq b_u\} \quad \forall t \geq 0. \end{aligned}$$

The goal of the controller is to drive the system from the starting point x_S to the terminal set $\mathcal{X}_F$. More formally, the controller aims to solve the following minimum time optimal control problem:

$$\begin{aligned} \min_{T,u_0,\ldots,u_{T-1}} \quad & \sum_{t=0}^{T-1} 1 \\ \text{s.t.} \quad & x_{t+1} = f(x_t, u_t) \qquad \forall t = [0,\ldots,T-1] \\ & x_t \in \mathcal{X},\ u_t \in \mathcal{U} \qquad \forall t = [0,\ldots,T] \\ & x_T = \mathcal{X}_F,\ x_0 = x_S \end{aligned} \tag{2}$$

where for a track of length L the terminal set

$$\mathcal{X}_F = \{x \in \mathbb{R}^n : [0\ 0\ 0\ 0\ 1\ 0]x = s \geq L\} \tag{3}$$

represents the states beyond the finish line.

III. Controller Design

In this section, we first show how to use historical data to construct a terminal constraint set and terminal cost function. Afterward, we exploit these quantities to design the controller.

A. Stored Data

As stated in Section I, we define one iteration as a successful lap around the race track and we store the closed-loop trajectories. In particular, at the jth iteration we define the vectors

$$\begin{aligned} \mathbf{u}^j &= \left[u_0^j, \ldots,\ u_{T^j}^j\right] \\ \mathbf{x}^j &= \left[x_0^j, \ldots,\ x_{T^j}^j\right] \end{aligned} \tag{4}$$

which collect the evolution of closed-loop system and associated input sequence. In the above definitions, T^j denotes the time at which the closed-loop system reached the terminal set, i.e., $x_{T^j} \in \mathcal{X}_F$.

B. Local Convex Safe Set

In this section, we define the local convex safe set. Different from our previous works [19]–[21], this quantity is constructed using a subset of the stored data points. In particular, the local convex safe set around x is defined as the convex hull of the K-nearest neighbors to x.

First, for the jth trajectory we define the set of time indices $[t_1^{j,*}, \ldots, t_K^{j,*}]$ associated with the K-nearest neighbors to the point x

$$\begin{aligned} \left[t_1^{j,*}, \ldots, t_K^{j,*}\right] = \underset{t_1,\ldots,t_K}{\text{argmin}} \quad & \sum_{i=1}^{K} \left|\left|x_{t_i}^j - x\right|\right|_D^2 \\ \text{s.t.} \quad & t_i \neq t_k \quad \forall i \neq k \\ & t_i \in \{0,\ldots,T^j\} \quad \forall i \in \{1,\ldots,K\}. \end{aligned} \tag{5}$$

In the above definition $||y||_D^2 = y^\top D^\top D y$ for the user-defined matrix D, which may be chosen to take into account the relative scaling or relevance of different variables. We chose $D = \text{diag}(0,0,0,0,1,0)$ to select the K-nearest neighbors with respect to the curvilinear abscissa s, which represents a proxy for the distance between two stored data points of the same lap. Furthermore, as the vehicle moves forward on the track, at each lap the stored data are ordered with respect to the traveled distance s and the computation of (5) is simplified. The K-nearest neighbors to x from the lth to the jth iteration are collected in the following matrix:

$$D_l^j(x) = \left[x_{t_1^{l,*}}^l, \ldots, x_{t_K^{l,*}}^l, \ldots, x_{t_1^{j,*}}^j, \ldots, x_{t_K^{j,*}}^j\right]$$

which is used to define the local convex safe set around x

$$\begin{aligned} \mathcal{CL}_l^j(x) = \big\{\bar{x} \in \mathbb{R}^n : \exists \boldsymbol{\lambda} \in \mathbb{R}^{K(j-l+1)}, \\ \boldsymbol{\lambda} \geq 0,\quad \mathbb{1}\boldsymbol{\lambda} = 1,\ D_l^j(x)\boldsymbol{\lambda} = \bar{x}\big\}. \end{aligned} \tag{6}$$

Note that the above local convex safe set $\mathcal{CL}_l^j(x)$ represents the convex hull of the K-nearest neighbors to x from the lth to jth iteration.

Finally, we define the matrix

$$S_l^j(x) = \left[x^l_{t_1^{l,*}+1}, \ldots, x^l_{t_K^{l,*}+1}, \ldots, x^j_{t_1^{j,*}+1}, \ldots, x^j_{t_K^{j,*}+1}\right]$$

which collects the evolution of the states stored in the columns of the matrix $D_l^j(x)$. The above matrix $S_l^j(x)$ will be used in Section III-D to construct the local convex safe set at each time step.

C. Local Convex Q-Function

In this section, we exploit the stored data to construct an approximation to the cost-to-go over the local convex safe set $\mathcal{CL}_l^j(x)$ around x. In particular, we define the local convex Q-function around x as the convex combination of the cost associated with the stored trajectories

$$\begin{aligned} Q_l^j(\bar{x}, x) = \min_{\boldsymbol{\lambda}} \quad & \mathbf{J}_l^j(x)\boldsymbol{\lambda} \\ \text{s.t} \quad & \boldsymbol{\lambda} \geq 0, \quad \mathbb{1}\boldsymbol{\lambda} = 1, D_l^j(x)\boldsymbol{\lambda} = \bar{x} \end{aligned} \tag{7}$$

where $\boldsymbol{\lambda} \in \mathbb{R}^{K(j-l)}$, $\mathbb{1}$ is a row vector of ones and the row vector

$$\begin{aligned} \mathbf{J}_l^j(x) = \big[& J^l_{t_1^{l,*} \to T^l}\left(x^l_{t_1^{l,*}}\right), \ldots, J^l_{t_M^{l,*} \to T^l}\left(x^l_{t_M^{l,*}}\right), \ldots, \\ & J^j_{t_1^{j,*} \to T^j}\left(x^j_{t_1^{j,*}}\right), \ldots, J^j_{t_M^{j,*} \to T^j}\left(x^j_{t_M^{j,*}}\right)\big] \end{aligned}$$

collects the cost-to-go associated with the K-nearest neighbors to x from the lth to the jth iteration. The cost-to-go $J^j_{t \to T^j}(x_t^j) = T^j - t$ represents the time to drive the vehicle from x_t^j to the finish line along the jth trajectory. We underline that the cost-to-go is computed after completion of the jth iteration.

D. Local LMPC Design

The local convex safe set and the local convex Q-function are used to design the controller. At each time t of the jth iteration, the controller solves the following finite-time optimal control problem:

$$\begin{aligned} & J^{\text{LMPC},j}_{t \to t+N}\left(x_t^j, z_t^j\right) \\ & = \min_{\mathbf{U}_t^j, \boldsymbol{\lambda}_t^j} \left[\sum_{k=t}^{t+N-1} h\left(x_{k|t}^j\right) + \mathbf{J}_l^{j-1}\left(z_t^j\right)\boldsymbol{\lambda}_t^j\right] && \text{(8a)} \\ & \text{s.t.} \quad x_{t|t}^j = x_t^j && \text{(8b)} \\ & \qquad \boldsymbol{\lambda}_t^j \geq 0, \quad \mathbb{1}\boldsymbol{\lambda}_t^j = 1, \quad D_l^{j-1}\left(z_t^j\right)\boldsymbol{\lambda}_t^j = x_{t+N|t}^j && \text{(8c)} \\ & \qquad x_{k+1|t}^j = A_{k|t}^j x_{k|t}^j + B_{k|t}^j u_{k|t}^j + C_{k|t}^j && \text{(8d)} \\ & \qquad x_{k|t}^j \in \mathcal{X}, \quad u_{k|t}^j \in \mathcal{U} && \text{(8e)} \\ & \qquad \forall k = t, \ldots, t+N-1 \end{aligned}$$

where $\mathbf{U}_t^j = [u_{t|t}^j, \ldots, u_{t+N-1|t}^j] \in \mathbb{R}^{d \times N}$, $\boldsymbol{\lambda}_t^j \in \mathbb{R}^{(j-l+1)K}$ and the stage cost in (8a)

$$h(x) = \begin{cases} 1, & \text{If } x \notin \mathcal{X}_F \\ 0, & \text{Else.} \end{cases}$$

In the above finite time optimal control problem (FTOCP) equations (8b), (8d), and (8e) represent the dynamic update, state, and input constraints. Finally, (8c) enforces $x_{t+N|t}^j$ into the local convex safe set defined in Section III-B. The optimal solution to (8) at time t of the jth iteration

$$\boldsymbol{\lambda}_t^{j,*}, \left[x_{t|t}^{j,*}, \ldots, x_{t+N|t}^{j,*}\right] \text{ and } \mathbf{U}_t^{j,*} = \left[u_{t|t}^{j,*}, \ldots, u_{t+N-1|t}^{j,*}\right] \tag{9}$$

is used to compute the following vector:

$$z_t^j = \begin{cases} x_N^{j-1}, & \text{If } t = 0 \\ S_l^j\left(z_{t-1}^j\right)\boldsymbol{\lambda}_{t-1}^{j,*}, & \text{Otherwise} \end{cases} \tag{10}$$

which at time t defines the local convex safe set $\mathcal{LS}_l^j(z_t^j)$ and local Q-function $Q_l^j(x, z_t^j)$ in (8). The above vector z_t^j represents a candidate terminal state for the planned trajectory of the LMPC at time t. First, we initialize the candidate terminal state z_0^j using the $(j-1)$th trajectory. Afterward, we update the vector z_t^j as the convex combination of the columns of the matrix $S_l^j(z_t^j)$ from Section III-B. Note that if the system is linear or if a linearized system approximates the nonlinear dynamics over the local convex safe set, then there exists a feasible input which drives the system from $x_{t+N|t}^{j,*} = D_t^{j-1}(z_t^j)\boldsymbol{\lambda}_t^{j,*}$ to $z_{t+1}^j = S_l^{j-1}(z_t^j)\boldsymbol{\lambda}_t^{j,*}$.

Finally, we apply to system (1) the first element of the optimizer vector

$$u_t^j = u_{t|t}^{j,*}. \tag{11}$$

The FTOCP (8) is repeated at time $t+1$, based on the new state $x_{t+1|t+1} = x_{t+1}^j$.

IV. System Identification Strategy

In this section, we illustrate the system identification strategy used to build an ATV model which approximates the vehicle dynamics. First, we introduce the kinematic equations of motion which describe the evolution of the vehicle's position as a function of the velocities. Afterward, we present the strategy used to approximate the dynamic equations of motion, which model the evolution of the vehicle's velocities as a function of the input commands. Finally, we describe the ATV model, which is computed online linearizing the kinematic equations of motion and evaluating the approximate dynamic equations of motion along the shifted optimal solution to the LMPC.

A. Kinematic Model

As mentioned in Section II, the position of the vehicle is expressed in the Frenet reference frame [22]. In particular, we describe the position of the vehicle in terms of lateral distance e_y from the centerline of the road and distance s traveled along a predefined path (see Fig. 1). The state e_ψ represents the difference between the vehicle's heading angle and the angle of the tangent vector to the path at the curvilinear abscissa s.

The rate of change of the vehicle's position in the curvilinear reference frame is described by the following kinematic

relationships:

$$\begin{aligned}\dot{e}_\psi &= w_z - \frac{v_x\cos(e_\psi) - v_y\sin(e_\psi)}{1-\kappa(s)e_y}\kappa(s)\\ \dot{s} &= \frac{v_x\cos(e_\psi) - v_y\sin(e_\psi)}{1-\kappa(s)e_y}\\ \dot{e}_y &= v_x\sin(e_\psi) + v_y\cos(e_\psi)\end{aligned}$$

where $\kappa(s)$ is the curvature of the centerline of the track at the curvilinear abscissa s [22]. The above equations can be Euler discretized to approximate the vehicle's motion as a function of the vehicle's velocities

$$\begin{aligned}e_{\psi_{k+1}} &= f_{e_\psi}(x_k) = e_{\psi_k}\\ &\quad + dt\left(w_{z_k} - \frac{v_{x_k}\cos(e_{\psi_k}) - v_{y_k}\sin(e_{\psi_k})}{1-\kappa(s_k)e_{y_k}}\kappa(s_k)\right)\\ s_{k+1} &= f_s(x_k) = s_k + dt\left(\frac{v_{x_k}\cos(e_{\psi_k}) - v_{y_k}\sin(e_{\psi_k})}{1-\kappa(s_k)e_{y_k}}\right)\\ \dot{e}_y &= f_{e_y}(x_k) = e_{y_k} + dt(v_{x_k}\sin(e_{\psi_k}) + v_{y_k}\cos(e_{\psi_k}))\end{aligned} \tag{12}$$

where dt is the discretization time. The above equations will be linearized to compute an ATV prediction model. It is interesting to note that (12) are independent of the vehicle's physical parameters, because these are derived from kinematic relationships between velocities and position.

B. Dynamic Model

The dynamic equations of motion, which describe the evolution of the vehicle's velocities, may be computed balancing the forces acting on the tires [10]. Therefore, the dynamic equations depend on physical parameters associated with the vehicle, tires and asphalt. These parameters may be estimated through a system identification campaign. However, the nonlinear dynamic equations of motion should be linearized in order to obtain an ATV model which allows us to reformulate the LMPC as a QP. Instead of identifying the parameters of a nonlinear model and then linearize it, we propose to directly learn a linear model around x using a local linear regressor. We introduce the Epanechnikov kernel function [23]

$$K(u) = \begin{cases}\frac{3}{4}(1-u^2), & \text{for } |u|<1\\ 0, & \text{else}\end{cases}$$

which is used to compute a local linear model around x for the longitudinal and lateral dynamics. In particular, for $l = \{v_x, v_y, w_z\}$ we compute the following regressor vector:

$$\Gamma^l(x) = \underset{\Gamma}{\operatorname{argmin}} \sum_{\{k,j\}\in I(x)} K\left(\frac{||x - x_k^j||_Q^2}{h}\right) y_k^{j,l}(\Gamma) \tag{13}$$

where the hyperparameter $h \in \mathbb{R}_+$ is the bandwidth, the row vector $\Gamma \in \mathbb{R}^5$,

$$\begin{aligned}y_k^{j,v_x}(\Gamma) &= ||v_{x_{k+1}}^j - \Gamma\left[v_{x_k}^j,\ v_{y_k}^j,\ w_{z_k}^j,\ a_k^j,\ 1\right]^T||\\ y_k^{j,v_y}(\Gamma) &= ||v_{y_{k+1}}^j - \Gamma\left[v_{x_k}^j,\ v_{y_k}^j,\ w_{z_k}^j,\ \delta_k^j,\ 1\right]^T||\\ y_k^{j,w_z}(\Gamma) &= ||w_{z_{k+1}}^j - \Gamma\left[v_{x_k}^j,\ v_{y_k}^j,\ w_{z_k}^j,\ \delta_k^j,\ 1\right]^T||\end{aligned}$$

and $I_l^j(x)$ is the set of indices

$$\begin{aligned}I_l^j(x) = \underset{\{k_1,j_1\},\ldots,\{k_P,j_P\}}{\operatorname{argmin}} \quad & \textstyle\sum_{i=1}^P ||x - x_{k_i}^{j_i}||_Q^2\\ \text{s.t.} \quad & k_i \neq k_n \quad \forall j_i = j_n\\ & k_i \in \{1,2,\ldots\} \quad \forall i \in \{1,\ldots,P\}\\ & j_i \in \{l,\ldots,j\} \quad \forall i \in \{1,\ldots,P\}\end{aligned}$$

where $||y||_Q = y^\top Q^\top Q y$ and the matrix Q is user defined. For the stored data from iteration l to iteration j, the set $I_l^j(x)$ collects the indices associated with the P-nearest neighbors to the state x. Finally, the user-defined matrix Q takes into account the relative scaling of different variables.

Note that the optimizer in (13) can be used to approximate the evolution of vehicle's velocities

$$\begin{aligned}\begin{bmatrix}v_{x_{k+1}}\\ v_{y_{k+1}}\\ w_{z_{k+1}}\end{bmatrix} &= \begin{bmatrix}\Gamma_{1:3}^{v_x}(x)\\ \Gamma_{1:3}^{v_y}(x)\\ \Gamma_{1:3}^{w_z}(x)\end{bmatrix}\begin{bmatrix}v_{x_k}\\ v_{y_k}\\ w_{z_k}\end{bmatrix}\\ &\quad + \begin{bmatrix}\Gamma_4^{v_x}(x) & 0\\ 0 & \Gamma_4^{v_y}(x)\\ 0 & \Gamma_4^{w_z}(x)\end{bmatrix}\begin{bmatrix}a_k\\ \delta_k\end{bmatrix} + \begin{bmatrix}\Gamma_5^{v_x}(x)\\ \Gamma_5^{v_y}(x)\\ \Gamma_5^{w_z}(x)\end{bmatrix}\end{aligned} \tag{14}$$

where for $l = \{v_x, v_y, w_z\}$ the scalar $\Gamma_i^l(x)$ denotes the ith element of the vector $\Gamma^l(x)$ and $\Gamma_{1:3}^l(x) \in \mathbb{R}^3$ is a row vector collecting the first three elements of $\Gamma^l(x)$ in (13).

C. ATV Model

In this section, we describe the strategy used to build an ATV model, which is then used for control. At time t of the jth iteration, we define the candidate solution $\bar{\mathbf{x}}_t^j = [\bar{x}_{t|t}^j, \ldots, \bar{x}_{t+N|t}^j]$ to Problem (8) using the optimal solution at time $t-1$ from (9)

$$\bar{x}_{k|t}^j = \begin{cases}x_{k|t-1}^{j,*}, & \text{If } k \in \{t,\ldots,t+N-1\}\\ z_t^j, & \text{If } k = t+N\end{cases}.$$

Finally at each time t of iteration j, the above candidate solution is used to build the following ATV model:

$$x_{k+1|t}^j = A_{k|t}^j x_{k|t}^j + B_{k|t}^j u_{k|t}^j + C_{k|t}^j \tag{15}$$

where $x_{k|t}^j = [v_{x_{k|t}}^j, v_{y_{k|t}}^j, w_{y_{k|t}}^j, e_{\psi_{k|t}}^j, s_{k|t}^j, e_{y_{k|t}}^j]$ and the matrices $A_{k|t}^j$, $B_{k|t}^j$, and $C_{k|t}^j$ are obtained linearizing (12) around $\bar{x}_{k|t}^j$ and evaluating (14) at $\bar{x}_{k|t}^j$

$$A_{k|t}^j = \begin{bmatrix}\Gamma_{1:3}^{v_x}(\bar{x}_{k|t}^j) & 0 & 0 & 0\\ \Gamma_{1:3}^{v_y}(\bar{x}_{k|t}^j) & 0 & 0 & 0\\ \Gamma_{1:3}^{w_z}(\bar{x}_{k|t}^j) & 0 & 0 & 0\\ \left(\nabla_x f_{e_\psi}(x)|_{\bar{x}_{k|t}^j}\right)^\top\\ \left(\nabla_x f_s(x)|_{\bar{x}_{k|t}^j}\right)^\top\\ \left(\nabla_x f_{e_y}(x)|_{\bar{x}_{k|t}^j}\right)^\top\end{bmatrix}, B_{k|t}^j = \begin{bmatrix}\Gamma_4^{v_x}(\bar{x}_{k|t}^j) & 0\\ 0 & \Gamma_4^{v_y}(\bar{x}_{k|t}^j)\\ 0 & \Gamma_4^{w_z}(\bar{x}_{k|t}^j)\\ 0 & 0\\ 0 & 0\\ 0 & 0\end{bmatrix} \tag{16}$$

and

$$C_k = \begin{bmatrix} \Gamma_5^{v_x}(\bar{x}^j_{k|t}) \\ \Gamma_5^{v_y}(\bar{x}^j_{k|t}) \\ \Gamma_5^{w_z}(\bar{x}^j_{k|t}) \\ f_{e_y}(\bar{x}^j_{k|t}) - \left(\nabla_x f_{e_y}(x)|_{\bar{x}^j_{k|t}}\right)^\top \bar{x}^j_{k|t} \\ f_s(\bar{x}^j_{k|t}) - \left(\nabla_x f_s(x)|_{\bar{x}^j_{k|t}}\right)^\top \bar{x}^j_{k|t} \\ f_{e_\psi}(\bar{x}^j_{k|t}) - \left(\nabla_x f_{e_\psi}(x)|_{\bar{x}^j_{k|t}}\right)^\top \bar{x}^j_{k|t} \end{bmatrix}. \quad (17)$$

V. Results

The proposed control strategy has been implemented on a 1/10-scale open-source vehicle platform called BARC [2]. The vehicle is equipped with a set of sensors, actuators, and two onboard CPUs to perform low-level control of the actuators as well as communication with a laptop, on which the high-level control is implemented. The CPUs are an Arduino Nano for low-level control of the actuators and an Odroid XU4 for WiFi communication with the i7 MSI GT72 laptop. The actuators are an electrical motor and a servo for the steering. The control architecture has been implemented in the Robot Operating System (ROS) framework, using Python and operator splitting solver for quadratic programs (OSQP) [24]. The code is available online[3].

We initialize the algorithm performing two laps of path following at constant speed. Each jth iteration collects the data of two consecutive laps. Therefore, the local safe set and local Q-function are defined also beyond the finish line. This strategy allows us to implement the LMPC for the repetitive autonomous racing control task, as shown in [19]. At each jth lap, we use the LMPC (8) and (11) to drive the vehicle from the starting line to the finish line and we use the closed-loop data to update the controller for the next lap. The parameters which define the controller are reported in Table I. We also added a small input rate cost in order to guarantee a unique solution to the QP associated with the LMPC.

We tested the controller on an oval-shaped and L-shaped tracks on which the vehicle runs in the counterclockwise direction. Fig. 2 shows that the lap time decreases until convergence is reached after 29 laps. Furthermore, Fig. 3 shows the raw acceleration measurements from the inertial measurement unit (IMU). We confirm that controller is able to operate the vehicle at the limit of its handling capability, reaching a maximum lateral acceleration close to 1 g[4]. Fig. 4 shows the evolution of the closed-loop trajectory on the xy plane and the velocity profile which is color coded. In the first row, we reported the path following trajectory used to initialize the LMPC and the closed-loop trajectories at laps 7 and 15. We note that the controller deviates from the initial feasible trajectory (reported in blue as the vehicle speed is 1.2 m/s) in order to explore the state space and to drive the vehicle at higher speeds until it converges to a steady-state behavior. The steady-state trajectories from laps 30 to 34 are reported in the bottom row of Fig. 4. Note that the color bar representing the velocity profile changed from the first to second row as the vehicle runs at a higher speed at the end of the learning process. We underline that the controller understands the benefit of breaking right before entering the curve and of accelerating when exiting. This behavior is optimal in racing as shown in [25].

[2]A video of the experiment can be found at https://youtu.be/ZBFJWtIbtMo

[3]The code is available on the BARC GitHub repository in the "devel-ugo" branch https://github.com/MPC-Berkeley/barc/tree/devel-ugo (github.com/MPC-Berkeley/barc)

[4]The maximum allowed lateral acceleration is computed assuming that the aerodynamic effects are negligible and that the lateral force acting on the vehicle is $F = \mu$mg for the friction coefficient $\mu = 1$.

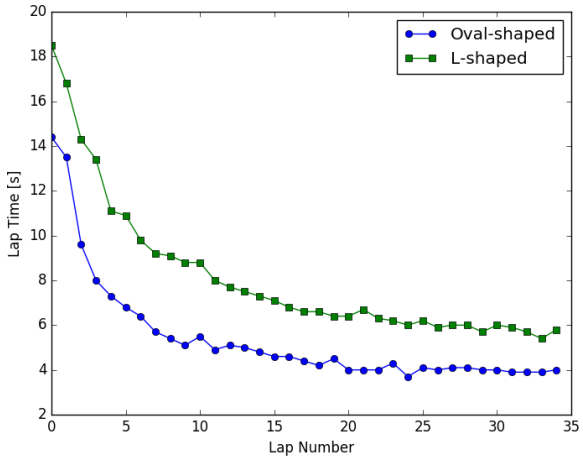

Fig. 2. Lap time of the LMPC on the oval-shaped and L-shaped tracks.

TABLE I
Parameters Used in the Controller Design

l	$j-2$
K	20
T	diag(0, 0, 0, 0, 1, 0)
Q	diag(0.1, 1, 1, 0, 0, 0)
P	80
h	10
N	12

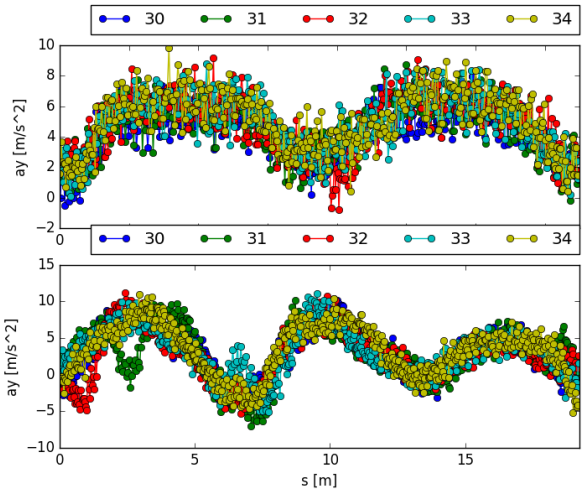

Fig. 3. Recorded lateral acceleration of the vehicle running on the oval-shaped track (top row) and L-shaped track (bottom row).

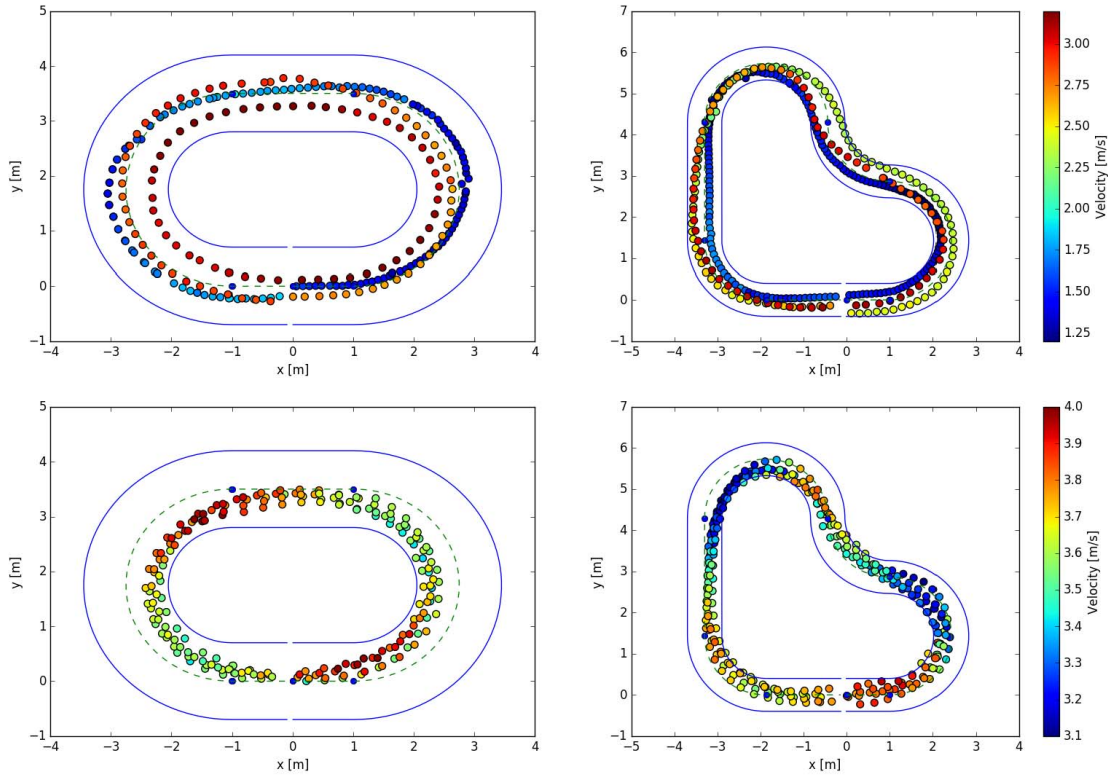

Fig. 4. First row in Fig. 3 shows the closed-loop trajectory that is used to initialize the LMPC and the closed-loop trajectories after few laps of learning. The second row shows the steady-state trajectories at which the LMPC has converged. Note that the scale of the color bar changes from the first to the second row, as the vehicle runs at higher speed after the learning process has converged.

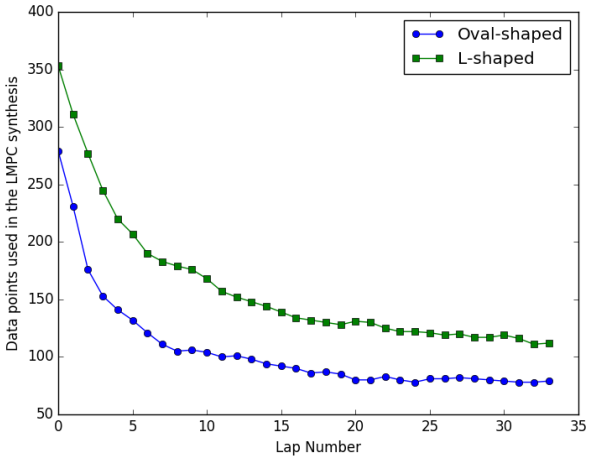

Fig. 5. Data points used in the LMPC design at each lap.

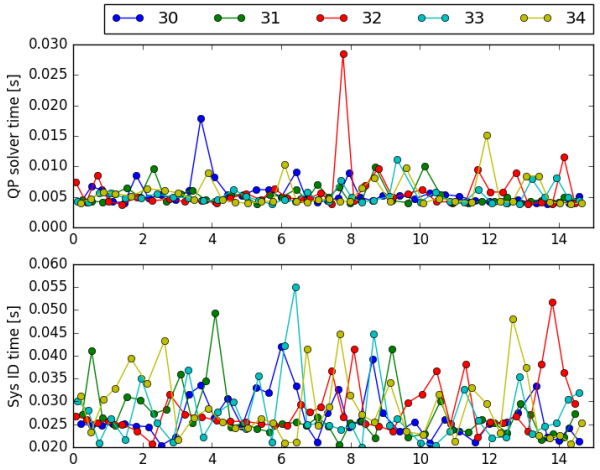

Fig. 6. The first rows shows the computational cost associated with the FTOCP. In the second row we reported the computational cost associated with the system identification strategy.

Furthermore, Fig. 5 shows the data points used to design the LMPC. Recall from Table I that at the jth lap the LMPC policy is synthesized using the trajectories from lap $l = j-2$ to lap $j-1$. Therefore, as the controller drives faster on the track, less data points are needed to design the LMPC. Moreover, in Fig. 6 we reported the computational time. It is interesting to note that on average the FTOCP (8) is solved in less than 10 ms, whereas it took 90 ms to solve the FTOCP associated with [19]. We underline that both strategies have been tested with a prediction horizon of $N = 12$ and a sampling time of 10 Hz. Therefore, this comparison shows the advantage of using the local convex safe set in (6), instead of the polynomial approximation to the safe set used in [20] and [21]. For more details on the polynomial approximation to the safe set, we refer to [20]. Finally, we note that it would be possible to parallelize the computation of the $N-1$ linear models which define the ATV model from (15). Indeed, at time t (16) and (17) may be evaluated independently and in parallel for each predicted time k.

VI. Conclusion

We presented an LMPC for autonomous racing. The proposed control framework uses historical data to construct safe sets and approximations to the value function. These quantities are systematically updated when a lap is completed, and as a result, the LMPC learns from experience to safely drive the vehicle at the limit of handling. We demonstrated the effectiveness of the proposed strategy on the BARC platform. Experimental results show that the controller learns to drive the vehicle aggressively, in order to minimize the lap time. In particular, the closed-loop system converged to a steady-state trajectory which cuts curves and reaches a lateral acceleration close to 1g.

References

[1] E. J. Rossetterand and J. C. Gerdes, "Lyapunov based performance guarantees for the potential field lane-keeping assistance system," *J. Dyn. Syst., Meas., Control*, vol. 128, no. 3, pp. 510–522, 2006.

[2] Y. Gao *et al.*, "Spatial predictive control for agile semi-autonomous ground vehicles," in *Proc. 11th Int. Symp. Adv. Vehicle Control*, 2012, pp. 1–6.

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

[4] J. V. Frasch *et al.*, "An auto-generated nonlinear MPC algorithm for real-time obstacle avoidance of ground vehicles," in *Proc. IEEE Eur. Control Conf. (ECC)*, Jul. 2013, pp. 4136–4141.

[5] M. Campbell, M. Egerstedt, J. P. How, and R. M. Murray, "Autonomous driving in urban environments: Approaches, lessons and challenges," *Philos. Trans. Roy. Soc. London A, Math. Eng. Phys. Sci.*, vol. 368, no. 1928, pp. 4649–4672, 2010.

[6] D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A review of motion planning techniques for automated vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 4, pp. 1135–1145, Nov. 2016.

[7] C. Katrakazas, M. Quddus, W.-H. Chen, and L. Deka, "Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions," *Transp. Res. C, Emerg. Technol.*, vol. 60, pp. 416–442, Nov. 2015.

[8] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Trans. Intell. Vehicles*, vol. 1, no. 1, pp. 33–55, Jun. 2016.

[9] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Aggressive driving with model predictive path integral control," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2016, pp. 1433–1440.

[10] R. Rajamani, *Vehicle Dynamics and Control*. New York, NY, USA: Springer, 2012.

[11] A. Alleyne, "A comparison of alternative intervention strategies for unintended roadway departure (URD) control," *Vehicle Syst. Dyn.*, vol. 27, no. 3, pp. 157–186, Mar. 1997.

[12] B. Alrifaee and J. Maczijewski, "Real-time trajectory optimization for autonomous vehicle racing using sequential linearization," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2018, pp. 476–483.

[13] R. Verschueren, S. De Bruyne, M. Zanon, J. V. Frasch, and M. Diehl, "Towards time-optimal race car driving using nonlinear MPC in real-time," in *Proc. 53rd IEEE Conf. Decis. Control*, Dec. 2014, pp. 2505–2510.

[14] R. Verschueren, M. Zanon, R. Quirynen, and M. Diehl, "Time-optimal race car driving using an online exact hessian based nonlinear MPC algorithm," in *Proc. Eur. Control Conf. (ECC)*, Jun. 2016, pp. 141–147.

[15] A. Liniger, A. Domahidi, and M. Morari, "Optimization-based autonomous racing of 1:43 scale RC cars," *Optim. Control Appl. Methods*, vol. 36, no. 5, pp. 628–647, 2015.

[16] A. Liniger and J. Lygeros, "Real-time control for autonomous racing based on viability theory," *IEEE Trans. Control Syst. Technol.*, vol. 27, no. 2, pp. 464–478, Mar. 2019.

[17] N. R. Kapania and J. C. Gerdes, "Path tracking of highly dynamic autonomous vehicle trajectories via iterative learning control," in *Proc. IEEE Amer. Control Conf. (ACC)*, Jul. 2015, pp. 2753–2758.

[18] P. A. Theodosis and J. C. Gerdes, "Generating a racing line for an autonomous racecar using professional driving techniques," in *Proc. ASME Dyn. Syst. Control Conf. Bath/ASME Symp. Fluid Power Motion Control*, 2011, pp. 853–860.

[19] M. Brunner, U. Rosolia, J. Gonzales, and F. Borrelli, "Repetitive learning model predictive control: An autonomous racing example," in *Proc. IEEE 56th Annu. Conf. Decis. Control*, Melbourne, VIC, Australia, Dec. 2017, pp. 2545–2550.

[20] U. Rosolia, A. Carvalho, and F. Borrelli, "Autonomous racing using learning model predictive control," in *Proc. Amer. Control Conf. (ACC)*, May 2017, pp. 5115–5120.

[21] U. Rosolia and F. Borrelli, "Learning model predictive control for iterative tasks: A computationally efficient approach for linear system," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 3142–3147, 2017.

[22] A. Micaelli and C. Samson, "Trajectory tracking for unicycle-type and two-steering-wheels mobile robots," in *Proc. INRIA*, 1993, pp. 1–42.

[23] V. A. Epanechnikov, "Non-parametric estimation of a multivariate probability density," *Theory Probab. Appl.*, vol. 14, no. 1, pp. 153–158, 1969.

[24] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: An operator splitting solver for quadratic programs," in *Proc. UKACC 12th Int. Conf. Control (CONTROL)*, Sep. 2018, p. 339.

[25] P. A. Theodosis and J. C. Gerdes, "Nonlinear optimization of a racing line for an autonomous racecar using professional driving techniques," in *Proc. ASME 5th Annu. Dyn. Syst. Control Conf.*, 2012, pp. 235–241.