Learning-based Model Predictive Control for Autonomous Racing

Author(s):
Kabzan, Juraj; Hewing, Lukas; Liniger, Alexander; Zeilinger, Melanie N.

Publication Date:
2019-06

Permanent Link:
https://doi.org/10.3929/ethz-b-000351561 →

Originally published in:
IEEE Robotics and Automation Letters , http://doi.org/10.1109/LRA.2019.2926677 →

Learning-based Model Predictive Control for Autonomous Racing

Juraj Kabzan[1], Lukas Hewing[1], Alexander Liniger[2], and Melanie N. Zeilinger[1]

***Abstract*—In this paper, we present a learning-based control approach for autonomous racing with an application to the AMZ Driverless race car *gotthard*. One major issue in autonomous racing is that accurate vehicle models that cover the entire performance envelope of a race car are highly nonlinear, complex and complicated to identify, rendering them impractical for control. To address this issue, we employ a relatively simple nominal vehicle model, which is improved based on measurement data and tools from machine learning. The resulting formulation is an online learning data-driven Model Predictive Controller, which uses Gaussian Processes regression to take residual model uncertainty into account and achieve safe driving behavior. To improve the vehicle model online, we select from a constant in-flow of data points according to a criterion reflecting the information gain, and maintain a small dictionary of 300 data points. The framework is tested on the full-size AMZ Driverless race car, where it is able to improve the vehicle model and reduce lap times by 10% while maintaining safety of the vehicle.**

***Index Terms*—Model Learning for Control, Learning and Adaptive Systems, Model Predictive Control, Autonomous Racing**

I. Introduction

In the past decade, autonomous driving has generated increasing interest in both academic and industrial research. In this paper, we focus on autonomous racing, a subfield of autonomous driving, where the goal is to drive a car around a track as quickly as possible. The field has received significant attention due to prominent races such as the DARPA Grand Challenge [1] or the recently initiated Formula Student Driverless (FSD)[3] competition [2].

Building on the increasing computational power, the use of machine learning and optimization-based techniques is now commonly investigated for these tasks [3]. In particular, many companies and researchers make use of Model Predictive Control (MPC) for path following and racing of autonomous vehicles [4], [5], [6]. MPC is an advanced control technique, which uses a model to optimize the predicted motion of a vehicle for a limited time horizon. This allows for enforcing constraints, such as collision or track constraints, and provides an intuitive way for trading off competing goals by shaping the cost function. MPC requires a system model that is able to adequately capture the vehicle dynamics while being simple enough to be used in an online optimization framework. Especially in autonomous racing, where the vehicle is operated at its performance limits, this is a challenging trade-off. Using a complex model may render MPC computationally intractable, whereas an overly simple model can result in reduced performance or even collisions. In addition, the dynamics can change between different racing instances or during a race, for instance, due to changing temperatures or tire wear, imposing the need to adapt the system model during operation.

Fig. 1: *gotthard* is a driverless electric 4WD race car.

This paper aims at meeting these challenges by considering a relatively simple *nominal* vehicle model, which is then improved online by learning the model error using Gaussian process regression. The proposed approach is based on a contouring MPC formulation for autonomous racing [4] and a learning-based MPC technique [7], [8]. Similar approaches for path tracking of mobile robots were previously presented e.g. in [9], [10] or [11], where the latter demonstrates rapid adaptation to changing model dynamics. A learning-based MPC approach for miniature race cars was presented in [12], which makes use of repetitive laps to learn an improved cost function of the MPC.

In contrast to these results, we develop and demonstrate our approach on a full-size autonomous race car *gotthard*, shown in Figure 1. Our main contribution lies in the implementation, and experimental validation of the learning-based control approach on this challenging race-car platform, which is operated at the performance limit achieving velocities of 15 $\frac{\mathrm{m}}{\mathrm{s}}$ and high lateral accelerations of up to 2 g, requiring fast sampling times and high-fidelity control. We extend the learning-based MPC approach in [8] with a data management system [13] in form of a dictionary of data points used by the GP model. This allows for continuously updating the learned vehicle model

Manuscript received: February 24, 2019; Revised May 24, 2019; Accepted Jun 21, 2019.

This paper was recommended for publication by Editor Paolo Rocco upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by the Swiss National Science Foundation under grant no. PP00P2 157601 / 1.

[1]Juraj Kabzan and Lukas Hewing and Melanie Zeilinger are with Institute for Dynamic Systems and Control, ETH Zurich, Switzerland kabzanj@gmail.com, lhewing|mzeilinger@ethz.ch

[2]Alexander Liniger is with Automatic Control Laboratory, ETH Zurich, Zurich, Switzerland liniger@control.ee.ethz.ch

Digital Object Identifier (DOI): see top of this page.

[3]http://www.formulastudent.de/

with recent state measurements during operation. The results include the first presentation of the *nominal* control framework of *gotthard*, which was used to successfully compete in a number of autonomous racing competitions. The experiments show that the *learning-based* approach is able to improve the system model, enabling lap time reductions of 10%.

II. Preliminaries

A. Notation

We use bold lowercase letters for vectors $\mathbf{x} \in \mathbb{R}^n$ and bold uppercase letters for matrices $\mathbf{X} \in \mathbb{R}^{n\times m}$, while scalars are non-bold. For vertical matrix concatenation $[\mathbf{X};\, \mathbf{U}] = [\mathbf{X}^T,\, \mathbf{U}^T]^T$ is used. We refer to the i-th element of vector $\mathbf{x}$ as $[\mathbf{x}]_i$ and similarly $[\mathbf{X}]_{\cdot,i}$ for the i-th column of matrix $\mathbf{X}$. A matrix without the i-th row is $\mathbf{X}_{\setminus i}$. We represent a diagonal matrix with elements $\mathbf{x}$ as $\mathrm{diag}(\mathbf{x})$. A normal distribution with mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\Sigma}$ is $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. The gradient of $\mathbf{f} : \mathbb{R}^{n_z} \to \mathbb{R}^{n_f}$ w.r.t. $\mathbf{x} \in \mathbb{R}^{n_x}$ is $\nabla_x \mathbf{f} : \mathbb{R}^{n_z} \to \mathbb{R}^{n_f \times n_x}$. We use $\mathbf{K} \leftarrow \tilde{\mathbf{K}}$ to express that matrix $\mathbf{K}$ is updated to $\tilde{\mathbf{K}}$. The squared weighted 2-norm $\mathbf{x}^T\mathbf{K}\mathbf{x}$ is $\|\mathbf{x}\|^2_{\mathbf{K}}$.

B. Gaussian Process Regression

In the following, we briefly introduce Gaussian Process (GP) regression. A detailed exposition can be found, e.g., in [14]. We identify an unknown function $\mathbf{d}_{\text{true}} : \mathbb{R}^{n_z} \to \mathbb{R}^{n_d}$ from a collection of inputs $\mathbf{z}_k \in \mathbb{R}^{n_z}$ and outputs $\mathbf{y}_k \in \mathbb{R}^{n_d}$

$$\mathbf{y}_k = \mathbf{d}_{\text{true}}(\mathbf{z}_k) + \mathbf{w}_k\,, \tag{1}$$

where $\mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}^{\mathbf{w}})$ is i.i.d. Gaussian noise with diagonal variance $\boldsymbol{\Sigma}^{\mathbf{w}} = \mathrm{diag}([\sigma_1^2, ..., \sigma_{n_d}^2])$. We consider a set of m input and output data pairs $(\mathbf{z}_i, \mathbf{y}_i)$ forming the dictionary

$$\begin{aligned}\mathcal{D} = \{&\mathbf{Y} = \left[\mathbf{y}_0^T; \ldots; \mathbf{y}_m^T\right] \in \mathbb{R}^{m\times n_d},\\ &\mathbf{Z} = \left[\mathbf{z}_0^T; \ldots; \mathbf{z}_m^T\right] \in \mathbb{R}^{m\times n_z}\}\,.\end{aligned} \tag{2}$$

Treating each output dimension $a \in \{1, ..., n_d\}$ independently, the posterior distribution in dimension a at a test point $\mathbf{z}$ is Gaussian with mean and variance

$$\mu^a(\mathbf{z}) = \mathbf{k}^a_{\mathbf{zZ}}(\mathbf{K}^a_{\mathbf{ZZ}} + \mathbf{I}\sigma_a^2)^{-1}[\mathbf{Y}]_{\cdot,a}\,, \tag{3a}$$

$$\Sigma^a(\mathbf{z}) = k^a_{\mathbf{zz}} - \mathbf{k}^a_{\mathbf{zZ}}(\mathbf{K}^a_{\mathbf{ZZ}} + \mathbf{I}\sigma_a^2)^{-1}\mathbf{k}^a_{\mathbf{Zz}}. \tag{3b}$$

Here, $\mathbf{K}^a_{\mathbf{ZZ}}$ is the Gram matrix, i.e. $[\mathbf{K}^a_{\mathbf{ZZ}}]_{ij} = k^a(\mathbf{z}_i, \mathbf{z}_j)$, $[\mathbf{k}^a_{\mathbf{Zz}}]_j = k^a(\mathbf{z}_j, \mathbf{z}) \in \mathbb{R}$, $\mathbf{k}^a_{\mathbf{Zz}} = (\mathbf{k}^a_{\mathbf{zZ}})^T \in \mathbb{R}^m$, and $k^a_{\mathbf{zz}} = k^a(\mathbf{z}, \mathbf{z}) \in \mathbb{R}$. We make use of the squared exponential kernel

$$k^a(\mathbf{z}, \bar{\mathbf{z}}) = \sigma_{f,a}^2 \exp\left(-(\mathbf{z} - \bar{\mathbf{z}})^T\mathbf{L}^a(\mathbf{z} - \bar{\mathbf{z}})\right), \tag{4}$$

where $\mathbf{L}^a \in \mathbb{R}^{n_z\times n_z}$ is the positive diagonal length scale matrix and $\sigma_{f,a}^2$ the squared signal variance.

The resulting multivariate GP approximation is given by

$$\mathbf{d}(\mathbf{z}) \sim \mathcal{N}(\boldsymbol{\mu}^d(\mathbf{z}), \boldsymbol{\Sigma}^d(\mathbf{z})) \tag{5}$$

with $\boldsymbol{\mu}^d(\mathbf{z}) = [\mu^1(\mathbf{z}); ...; \mu^{n_d}(\mathbf{z})] \in \mathbb{R}^{n_d}$ and $\boldsymbol{\Sigma}^d(\mathbf{z}) = \mathrm{diag}([\Sigma^1(\mathbf{z}); ...; \Sigma^{n_d}(\mathbf{z})]) \in \mathbb{R}^{n_d\times n_d}$. The computational complexity of GP regression strongly depends on the number of data points m, which motivates the use of sparse approximations using inducing points, outlined in the following.

C. Sparse Gaussian Process Regression

In order to reduce the computational cost, inducing inputs $\mathbf{Z}_{\text{ind}} = \left[\mathbf{z}_0^T; \ldots; \mathbf{z}_{\tilde{m}}^T\right]$, with $\tilde{m} \ll m$, can be used to approximate (3), see e.g. [15]. We use FITC [16] given by

$$\tilde{\mu}^a(\mathbf{z}) = \mathbf{Q}^a_{\mathbf{zZ}}(\mathbf{Q}^a_{\mathbf{ZZ}} + \boldsymbol{\Lambda})^{-1}[\mathbf{Y}]_{\cdot,a}\,, \tag{6a}$$

$$\tilde{\Sigma}^a(\mathbf{z}) = k^a_{\mathbf{zz}} - \mathbf{Q}^a_{\mathbf{zZ}}(\mathbf{Q}^a_{\mathbf{ZZ}} + \boldsymbol{\Lambda})^{-1}\mathbf{Q}^a_{\mathbf{Zz}}\,, \tag{6b}$$

with $\mathbf{Q}^a_{\boldsymbol{\zeta}\tilde{\boldsymbol{\zeta}}} = \mathbf{K}^a_{\boldsymbol{\zeta}\mathbf{Z}_{\text{ind}}}(\mathbf{K}^a_{\mathbf{Z}_{\text{ind}}\mathbf{Z}_{\text{ind}}})^{-1}\mathbf{K}^a_{\mathbf{Z}_{\text{ind}}\tilde{\boldsymbol{\zeta}}}$, $\boldsymbol{\Lambda} = \mathrm{diag}(\mathbf{K}^a_{\mathbf{ZZ}} - \mathbf{Q}^a_{\mathbf{ZZ}} + \mathbf{I}\sigma_a^2)$. Many quantities in (6) do not depend on $\mathbf{z}$ and can be precomputed, such that they only need to be updated when updating $\mathbf{Z}_{\text{ind}}$ or $\mathcal{D}$ itself. The resulting distribution is $\tilde{\mathbf{d}}(\mathbf{z}) \sim \mathcal{N}(\tilde{\boldsymbol{\mu}}^d(\mathbf{z}), \tilde{\boldsymbol{\Sigma}}^d(\mathbf{z}))$. By reducing the effective size of the kernel matrix via the reduced number of inducing points, the computational complexity can be controlled. In this work, we take advantage of the MPC formulation and select the inducing points along the prediction horizon, updating them in every time step. This enables the use of a small number of inducing points compared with the problem dimension and thereby very efficient evaluation of (6) during optimization, see Section IV-D and [7] for more details.

III. Race Car Model

In this section, we describe the employed vehicle model, which is commonly used for control and provides a good trade-off between simplicity for real-time implementation and accuracy for high-performance control of the autonomous race car. The model is based on a dynamic bicycle model illustrated in Figure 2, with states

$$\mathbf{x} = [X; Y; \varphi; v_x; v_y; r; \delta; T]\,,$$

namely the position $\mathbf{p} = [X; Y]$ and heading angle φ in the global coordinate frame, the velocities $\mathbf{v} = [v_x; v_y]$ in the vehicle's body frame and yaw rate r, as well as the steering angle δ and driver command T corresponding to a desired acceleration. The inputs to the system are the change in steering angle and applied driver command: $\mathbf{u} = [\Delta\delta; \Delta T]$.

The considered model used for control is of the form

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{B}_d(\mathbf{d}(\mathbf{z}_k) + \mathbf{w}_k)\,, \tag{7}$$

where $\mathbf{f}$ describes the nominal vehicle dynamics (Section III-A) and $\mathbf{d}$ the additional learned part of the dynamics, estimating the model error $\mathbf{d}_{\text{true}}$ of the nominal model (Section III-B). Together with the process noise $\mathbf{w}_k$, the learned part of the dynamics is assumed to only affect the subspace spanned by $\mathbf{B}_d$, corresponding to the velocity states of the vehicle, and to depend on a set of features $\mathbf{z}_k$ relevant for the regression, which are extracted from $\mathbf{x}_k$, $\mathbf{u}_k$.

A. Nominal Vehicle Model

As a nominal system model, we consider a dynamic bicycle model with nonlinear tire forces, an underlying torque vectoring controller and simple input dynamics.

The car is assumed to be one rigid body with mass m and a yaw moment of inertia I_z, while $l_{R/F}$ defines the distance between the center of gravity and the rear and front axle, respectively. The front/rear lateral tire forces are denoted by

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LRA.2019.2926677, IEEE Robotics and Automation Letters

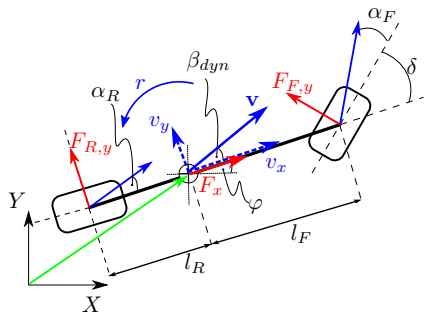


Fig. 2: Bicycle Model: The position vector is depicted in green, velocities in blue and forces in red.

$F_{F/R,y}$ and F_x is the longitudinal drive-train force. Altogether, the nominal model can be expressed as

$$\dot{\mathbf{x}} = \begin{bmatrix} v_x \cos\varphi - v_y \sin\varphi \\ v_x \sin\varphi + v_y \cos\varphi \\ r \\ \frac{1}{m}(F_x - F_{F,y}\sin\delta + m v_y r) \\ \frac{1}{m}(F_{R,y} + F_{F,y}\cos\delta - m v_x r) \\ \frac{1}{I_z}(F_{F,y} l_F \cos\delta - F_{R,y} l_R + \tau_{\text{TV}}) \\ \Delta\delta \\ \Delta T \end{bmatrix} . \tag{8}$$

Here, τ_{TV} is an additional yaw torque generated by an underlying torque vectoring controller

$$r_{\text{target}} = \delta \frac{v_x}{l_F + l_R} ,$$
$$\tau_{\text{TV}} = (r_{\text{target}} - r) P_{\text{TV}} ,$$

where $P_{\text{TV}} \in \mathbb{R}^+$ is the proportional controller gain. The front and rear slip angles $\alpha_{F/R}$ are used to compute the lateral force based on a simplified Pacejka tire model [17]

$$\alpha_R = \arctan\left(\frac{v_y - l_R r}{v_x}\right) ,$$
$$\alpha_F = \arctan\left(\frac{v_y + l_F r}{v_x}\right) - \delta ,$$
$$F_{R,y} = D_R \sin\Big(C_R \arctan\Big(B_R \alpha_R\Big)\Big) ,$$
$$F_{F,y} = D_F \sin\Big(C_F \arctan\Big(B_F \alpha_F\Big)\Big) ,$$

where $D_{F/R}, C_{F/R}, B_{F/R}$ are tire specific constants.

The applied longitudinal force depends on the desired driver command T in the range ± 1, where 1 corresponds to maximum acceleration and -1 to maximum braking. It is modeled as a single force applied at the center of gravity of the vehicle and is computed through the following equation

$$F_x = C_{m1} T - C_{r0} - C_{r2} v_x^2 ,$$

which consists of a simple drivetrain model $C_{m1}T$, rolling resistance C_{r0}, and drag $C_{r2}v_x^2$. The last two equations in (8) reflect a delay in the input commands.

For integration in a discrete-time MPC formulation, the system is discretized with a Runge-Kutta 4th-order integration using a sampling time of $T_s = 50\,\text{ms}$, resulting in $\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)$ in (7).

B. Model Learning

While the previously presented *nominal* system model is sufficient for operation of the race car, we aim at enhancing performance and enabling automatic model adaptation by inferring the model error $\mathbf{d}(\mathbf{z}_k)$ in (7) from the deviation of the nominal model from measurement data during operation. Based on physical considerations, we assume that the model error only affects the dynamic part of the system equations, i.e. the velocity states, by selecting $\mathbf{B}_d = [\mathbf{0}_{3\times3}; \mathbf{I}_{3\times3}; \mathbf{0}_{2\times3}]$. To reduce dimensionality of the learning problem, we furthermore assume model errors to be independent of the vehicle's position by selecting as regression features

$$\mathbf{z} = \left[v_x; v_y; r; \delta + \frac{1}{2}\Delta\delta; T + \frac{1}{2}\Delta T \right] , \tag{9}$$

where we heuristically correct the steering angle and driver command to account for the input dynamics, i.e. $\delta + \frac{1}{2}\Delta\delta$ approximates the physical steering angle between time step k and $k+1$. The training data $\mathbf{y}_k$ is generated from the difference between measurements $\mathbf{x}_{k+1}$ and the nominal model predictions

$$\mathbf{y}_k = \mathbf{B}_d^\dagger (\mathbf{x}_{k+1} - \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)) = \mathbf{d}_{\text{true}}(\mathbf{z}_k) + \mathbf{w}_k ,$$

where $\mathbf{B}_d^\dagger$ is the Moore-Penrose pseudo-inverse, and $\mathbf{d}_{\text{true}}(\mathbf{z}_k)$ the *true* model error function we want to approximate. Note that this is the standard form for a regression task as in (1), and applying GP regression results in a stochastic estimate $\mathbf{d}(\mathbf{z}_k) \sim \mathcal{N}(\boldsymbol{\mu}^d(\mathbf{z}_k), \boldsymbol{\Sigma}^d(\mathbf{z}_k))$ given the collected data $\mathcal{D}$.

An important step of GP regression is the choice of hyperparameters in the kernel function (4). Since we assume the general character of the model error to remain constant during operation of the vehicle, we determine the hyperparameters and noise level $\mathbf{w}_k$ before operation using maximum likelihood optimization based on historical data and fix them during the race.

IV. Learning-based Controller Formulation

In this section, we recount the used contouring-based MPC formulation for autonomous racing [4] and learning-based extension [7], [8], including some modifications for the considered platform. The MPC formulation without learned GP model corresponds to the *nominal* control formulation previously used in autonomous racing competitions.

A. State & Uncertainty Prediction

Model Predictive Control requires the prediction of the system states over a finite horizon to optimize the control sequence. The use of an uncertain stochastic system resulting from the learned dynamics renders the predicted state sequence a random variable. We derive a tractable approximation of the uncertainty at each state in the prediction horizon, which is then used to formulate *chance constraints*, resulting in dynamic safety margins to the track boundaries (see Section IV-C). Considering uncertainty explicitly in the control formulation can improve safety and performance in a race, as e.g. also demonstrated in [18].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LRA.2019.2926677, IEEE Robotics and Automation Letters

Evaluation of the GP model $\mathbf{d}$ at a given input results in a mean $\boldsymbol{\mu}^d$ and variance $\boldsymbol{\Sigma}^d$, which directly represents the residual uncertainty of the prediction model. In order to evaluate the uncertainty in the states over the MPC prediction horizon, we propagate the mean and variance using successive linearizations, similar to extended Kalman filtering [8]. This results in mean $\boldsymbol{\mu}^{\mathbf{x}}$ and variance $\boldsymbol{\Sigma}^{\mathbf{x}}$ propagation given by

$$\boldsymbol{\mu}^{\mathbf{x}}_{k+1} = \mathbf{f}(\boldsymbol{\mu}^{\mathbf{x}}_k, \mathbf{u}_k) + \mathbf{B}_d \boldsymbol{\mu}^d(\boldsymbol{\mu}^{\mathbf{z}}_k)\,, \tag{10a}$$

$$\begin{aligned}\boldsymbol{\Sigma}^{\mathbf{x}}_{k+1} = &\begin{bmatrix}\nabla_x \mathbf{f}(\boldsymbol{\mu}^{\mathbf{x}}_k, \mathbf{u}_k) & \mathbf{B}_d\end{bmatrix} \\ &\begin{bmatrix}\boldsymbol{\Sigma}^{\mathbf{x}}_k & \star \\ \nabla_x \boldsymbol{\mu}^d(\boldsymbol{\mu}^{\mathbf{z}}_k)\boldsymbol{\Sigma}^{\mathbf{x}}_k & \boldsymbol{\Sigma}^d(\boldsymbol{\mu}^{\mathbf{z}}_k) + \boldsymbol{\Sigma}^{\mathbf{w}}\end{bmatrix} \\ &\begin{bmatrix}\nabla_x \mathbf{f}(\boldsymbol{\mu}^{\mathbf{x}}_k, \mathbf{u}_k) & \mathbf{B}_d\end{bmatrix}^T\,,\end{aligned} \tag{10b}$$

where the star denotes the corresponding elements of the symmetric matrix. Here $\boldsymbol{\mu}^{\mathbf{z}}_k$ are the regression features (9) evaluated at the mean state prediction.

B. Contouring Control & Resulting Cost Function

We consider the task of racing along a race track of varying width, where the centerline is given by a piecewise cubic spline polynomial. The track is parameterized by $\theta \in [0, \theta_{\text{max}}]$, i.e. given a θ, the corresponding centerline position $[X_c(\theta); Y_c(\theta)]$, orientation $\phi_c(\theta)$, as well as the track radius $R_c(\theta)$ can be evaluated.

The control formulation aims at maximizing the progress along the given track by adding a state θ_k representing the approximate position of the vehicle along this center line. Progress along the track is encouraged by introducing the incremental progress $\theta_{k+1} = \theta_k + v_k$, where v_k is a decision variable that is maximized by introducing a linear negative cost $-\kappa v_k$ in the optimization.

The progress variable θ_k is linked to the physical location of the car by penalizing the so-called lag error e_l and contouring error e_c, defined as

$$\begin{aligned}e_l(\boldsymbol{\mu}^{\mathbf{x}}_k, \theta_k) = &-\cos\left(\Phi(\theta_k)\right)(\mu^X_k - X_c(\theta_k)) \\ &-\sin\left(\Phi(\theta_k)\right)(\mu^Y_k - Y_c(\theta_k))\,, \\ e_c(\boldsymbol{\mu}^{\mathbf{x}}_k, \theta_k) = &\quad\sin\left(\Phi(\theta_k)\right)(\mu^X_k - X_c(\theta_k)) \\ &-\cos\left(\Phi(\theta_k)\right)(\mu^Y_k - Y_c(\theta_k))\,,\end{aligned}$$

where $[\mu^X_k; \mu^Y_k]$ is the current mean position of the vehicle. For sufficiently small errors, $[X_c(\theta_k); Y_c(\theta_k)]$ therefore approximates the vehicle position projected onto the centerline. The resulting contouring cost is given by

$$J(\boldsymbol{\mu}^{\mathbf{x}}_k, \theta_k, v_k) = q_c e_c(\boldsymbol{\mu}^{\mathbf{x}}_k, \theta_k)^2 + q_l e_l(\boldsymbol{\mu}^{\mathbf{x}}_k, \theta_k)^2 - \kappa v_k\,,$$

where $q_c, q_l, \kappa \in \mathbb{R}^+$ are weights.

In addition to the contouring cost, two regularization terms are used. The first is a quadratic cost on the steering angle and applied driver command, as well as their corresponding changes, i.e.

$$U(\boldsymbol{\mu}^{\mathbf{x}}_k, \mathbf{u}_k) = \left\|[\mu^\delta_k;\, \mu^T_k]\right\|^2_{\mathbf{R_x}} + \left\|[\Delta\delta_k;\, \Delta T_k]\right\|^2_{\mathbf{R_u}}\,.$$

The second regularizer is given as

$$L(\boldsymbol{\mu}^{\mathbf{x}}_k) = q_\beta(\beta^{\text{kin}}_k - \beta^{\text{dyn}}_k)^2\,,$$

and influences the driving aggressiveness by keeping the dynamic slip angle $\beta^{\text{dyn}}_k = \arctan\left(\mu^{v_y}_k / \mu^{v_x}_k\right)$ close to the kinematic slip angle $\beta^{\text{kin}}_k = \arctan\left(\tan(\mu^\delta_k) l_R / (l_R + l_F)\right)$.

C. Track, Tire & Input Constraints

While maximizing progress along the track, it is critical to keep the vehicle within the track boundaries, i.e. within the track radius. We express this as a constraint on the vehicle position, taking the model uncertainty into account, which is given by its variance $\boldsymbol{\Sigma}^{XY}_k$, i.e. the relevant submatrix of $\boldsymbol{\Sigma}^{\mathbf{x}}_k$. We use the uncertainty to effectively tighten the track radius imposed as a constraint on the mean prediction by

$$R_{\text{GP}}(\boldsymbol{\Sigma}^{XY}_k) = \sqrt{\chi^2_2(p)\lambda_{\max}(\boldsymbol{\Sigma}^{XY}_k)}\,,$$

in which $\lambda_{\max}(\boldsymbol{\Sigma}^{XY}_k)$ is the maximum eigenvalue of the variance matrix, and $\chi^2_2(p)$ the quantile function of the chi-squared distribution, corresponding to a maximum violation probability of p. For details on the formulation, see [7]. The resulting constraint can then be expressed as

$$\left\|\begin{bmatrix}\mu^X_k \\ \mu^Y_k\end{bmatrix} - \begin{bmatrix}X_c(\theta_k) \\ Y_c(\theta_k)\end{bmatrix}\right\|^2 \leq \left\|R(\theta_k) - R_{\text{GP}}\left(\boldsymbol{\Sigma}^{XY}_k\right)\right\|^2\,, \tag{11}$$

with $R(\theta_k)$ being the track radius at centerline position θ_k. Since the MPC prediction is open loop, the variance given by (10b) can grow rapidly. This is typically conservative since MPC provides feedback at every time step, which reduces the error. One could address this by implementing feedback over the planning horizon [8], but it is typically challenging to design a simple ancillary controller for highly nonlinear problems. For computational and simplicity reasons we therefore heuristically address this issue by limiting the tightening to a shorter horizon $N_{\text{shrink}} < N$, which was shown to perform well in practice [18].

In addition to the track constraints, tire forces are limited to a tire-specific frictional ellipse

$$(p_{\text{long}} F_x)^2 + F^2_{F/R,y} \leq (p_{\text{ellipse}} D_{F/R})^2\,, \tag{12}$$

with $p_{\text{long/ellipse}} \in \mathbb{R}$ influencing the shape of the ellipse. Finally, we restrict the steering angle δ, and the driver command T as well as their corresponding rates to

$$\begin{bmatrix}\delta_{\min} \\ T_{\min} \\ \Delta\delta_{\min} \\ \Delta T_{\min}\end{bmatrix} \leq \begin{bmatrix}\mu^\delta_k \\ \mu^T_k \\ \Delta\delta_k \\ \Delta T_k\end{bmatrix} \leq \begin{bmatrix}\delta_{\max} \\ T_{\max} \\ \Delta\delta_{\max} \\ \Delta T_{\max}\end{bmatrix}\,. \tag{13}$$

D. Computational Simplifications and Resulting Formulation

The MPC problem for autonomous racing, derived in the previous sections, has to be solved in real-time at fast sampling rates below $50\,\text{ms}$. Aside from the basic MPC problem itself, this is in particular challenged by the increased dimensionality of the problem including the variance dynamics and the required GP evaluations. To derive a real-time capable formulation, we approximate the problem by leveraging the receding horizon character and the available solution trajectory from the previous time step, based on the idea that the trajectories show small changes between the fast sampling times.

The variance dynamics (10b) and subsequently the track constraints are therefore evaluated for the previous trajectory and precomputed, such that they remain fixed during optimization. We furthermore make use of this trajectory for a dynamic sparse approximation of the GP, by placing inducing points equally spaced along this trajectory for a local approximation of the GP. This allows for precomputing the relevant quantities in (6a) for each instance of the MPC optimization, significantly reducing the evaluation complexity of the GP during optimization, see Section II-C and [7], [8]. The resulting MPC problem is

$$\begin{aligned} \min_{\{\mathbf{u}_k,\, v_k\}} \quad & \sum_{k=1}^{N} J(\boldsymbol{\mu}_k^{\mathbf{x}}, \theta_k, v_k) + U(\boldsymbol{\mu}_k^{\mathbf{x}}, \mathbf{u}_k) + L(\boldsymbol{\mu}_k^{\mathbf{x}}) \\ \text{s.t.} \quad & \boldsymbol{\mu}_0^{\mathbf{x}} = \mathbf{x}(t), \\ & \boldsymbol{\mu}_{k+1}^{\mathbf{x}} = \mathbf{f}(\boldsymbol{\mu}_k^{\mathbf{x}}, \mathbf{u}_k) + \mathbf{B}_d \tilde{\boldsymbol{\mu}}^d(\boldsymbol{\mu}_k^{\mathbf{z}}), \\ & \theta_{k+1} = \theta_k + v_k, \\ & (11) \text{ with } \overline{\boldsymbol{\Sigma}}_k^{XY},\, \overline{\theta}_k, \\ & (12), (13)\,, \end{aligned} \tag{14}$$

where $\overline{\theta}_k$, $\overline{\boldsymbol{\Sigma}}_k^{XY}$ indicate that the quantities have been precomputed based on the previous solution trajectory, and $\mathbf{x}(t)$ is the currently measured state of the system for which the problem is solved. To ensure feasibility of the optimization problem, all state constraints are implemented as soft constraints.

Problem (14) can be solved using a pre-determined data set for the GP, e.g. from a previous experimental run, as presented in [8]. In order to successively improve controller performance and account for online variation, the following section introduces a procedure for continuously updating the GP model, i.e. the data set $\mathcal{D}$, during operation.

V. Online Learning

Gaussian Process regression is a nonparametric technique, where the function estimate is directly based on the selected set of data points. In a naive implementation of an online learning scenario, one would keep track of *all* data from a continuous data stream, inevitably leading to computational infeasibility over time. In order to keep the computation time for our application limited, we restrict the number of actively used data points to M, which are stored in a dictionary $\mathcal{D}$. To update the dictionary online, we employ a selection technique similar to [13], outlined in the following.

A. Dictionary Selection

The goal of the data selection is to cover the feature-space as well as possible using a limited amount of data points from the constant data in-flow. The following describes the implemented procedure for deciding when a new data sample $\mathbf{d}_{m+1} = (\mathbf{z}_{m+1}, \mathbf{y}_{m+1})$ is added to the dictionary and, once the dictionary reaches maximum size M, which data point is replaced.

The data selection is based on a *distance measure* γ_i for each dictionary point $\mathbf{d}_i$, expressing its similarity to the other data points $\mathbf{d}_j \in \mathcal{D}$. It is defined as the posterior variance at the data point location $\mathbf{z}_i$, given all other data points currently in the dictionary $\mathbf{Z}_{\setminus i}$, i.e.

$$\gamma_i = k^a_{\mathbf{z}_i \mathbf{z}_i} - \mathbf{k}^a_{\mathbf{z}_i \mathbf{Z}_{\setminus i}} (\mathbf{K}^a_{\mathbf{Z}_{\setminus i} \mathbf{Z}_{\setminus i}} + \mathbf{I}\lambda)^{-1} \mathbf{k}^a_{\mathbf{Z}_{\setminus i} \mathbf{z}_i}\,, \tag{15}$$

where we introduce a noise level λ as a tuning parameter for regularization of the Gram matrix. A high value of γ_i means that the input location $\mathbf{z}_i$ is not well covered by other data points, which is related to the information gain of adding data point i to the dictionary [13].

1) Adding a data point to the dictionary: We include a new data point $\mathbf{d}_{m+1}$, if its distance value γ_{m+1} is greater than a threshold η, which is chosen to control the update frequency. In addition, the data point is also added if γ_{m+1} is greater than the median of all other distance measures in the dictionary.

Since the distance measure for each data point γ_i requires $\{k^a_{\mathbf{z}_i, \mathbf{z}_i}, \mathbf{k}^a_{\mathbf{Z}_{\setminus i}, \mathbf{z}_i}, \mathbf{K}^a_{\mathbf{Z}_{\setminus i}, \mathbf{Z}_{\setminus i}}, \gamma_i\}$, which are dependent on all other dictionary points, all distance measures must be updated when including the new sample $\mathbf{d}_{m+1}$ in the dictionary. Inserting a new point extends the input data set $\mathbf{Z} \leftarrow [\mathbf{Z}; \mathbf{z}_{m+1}^T] \in \mathbb{R}^{(m+1)\times n_z}$ and results in updates for each point i

$$\mathbf{K}^a_{\mathbf{Z}_{\setminus i}, \mathbf{Z}_{\setminus i}} \leftarrow \begin{bmatrix} \mathbf{K}^a_{\mathbf{Z}_{\setminus i}, \mathbf{Z}_{\setminus i}} & \mathbf{k}^a_{\mathbf{Z}_{\setminus i}, \mathbf{z}_{m+1}} \\ \star & k^a_{\mathbf{z}_{m+1}, \mathbf{z}_{m+1}} \end{bmatrix} \in \mathbb{R}^{m\times m},$$

$$\mathbf{k}^a_{\mathbf{Z}_{\setminus i}, \mathbf{z}_i} \leftarrow \begin{bmatrix} \mathbf{k}^a_{\mathbf{Z}_{\setminus i}, \mathbf{z}_i} \\ k^a_{\mathbf{z}_{m+1}, \mathbf{z}_i} \end{bmatrix} \in \mathbb{R}^{m}\,.$$

Since the matrices $(\mathbf{K}^a_{\mathbf{Z}_{\setminus i}, \mathbf{Z}_{\setminus i}} + \lambda \mathbf{I})$ are positive definite, a robust Cholesky decomposition with pivoting is used in order to compute (15) and update γ_i.

2) Replacing a data point in the dictionary: If the number of data points exceeds the maximum size of the dictionary M, we drop the point with the lowest distance measure. In order to encourage older data points to be removed first, we modify the distance measure via an exponential forgetting factor

$$\tilde{\gamma}_i = \exp\Big(-\frac{(t-t_i)^2}{2h}\Big)\gamma_i\,,$$

where t is the current time, while t_i is the time-stamp of each data point and h is a tuning parameter. This modification ensures that over time the model can adjust to changes in the vehicle dynamics by fading out older data points. Replacing a data point $\mathbf{d}_j$ with a new point $\mathbf{d}_{m+1}$ again requires that the corresponding matrices and vectors $\mathbf{K}^a_{\mathbf{Z}_{\setminus i}, \mathbf{Z}_{\setminus i}}$, $\mathbf{k}^a_{\mathbf{Z}_{\setminus i}, \mathbf{z}_i}$ of each remaining dictionary point are updated to compute the new distance measures γ_i. This is done by replacing each instance of $\mathbf{z}_j$ with $\mathbf{z}_{m+1}$ in the computation of (15), which is then similarly solved using a full Cholesky decomposition.

B. Outlier Rejection

GP regression is generally susceptible to outliers, which can dramatically deteriorate the performance of the learned error correction. In addition, large and sudden changes in the GP can lead to erratic and undesirable driving behavior. To alleviate these problems and enforce a gradual change in the GP predictions over time, we make use of two types of filters for outlier rejection.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LRA.2019.2926677, IEEE Robotics and Automation Letters

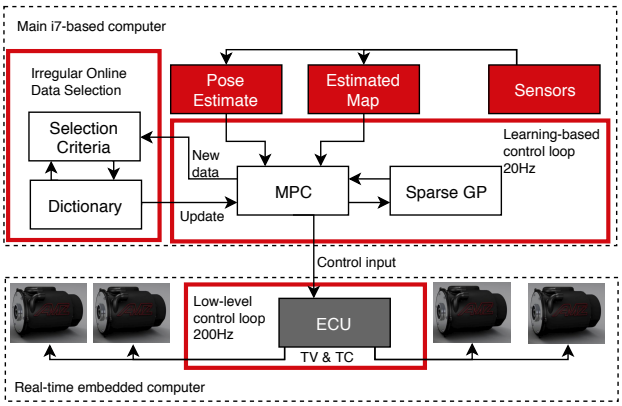


Fig. 3: Architecture of the control framework of *gotthard*. Low-level and high-level (MPC) controllers run in real-time at 200 Hz and 20 Hz, respectively, while data point selection and GP-update is irregular and non real-time.

First, we only consider state measurements that lie within predefined bounds $\pm\mathbf{y}_{\text{lim}}$, e.g. from physical considerations or previous experiment runs. This filter will reject large errors, which could for instance result from issues in the state estimation and localization. The second filter makes use of the current prediction of the GP by requiring a new data point $\mathbf{d}_{m+1} = (\mathbf{z}_{m+1}, \mathbf{y}_{m+1})$ to satisfy the condition

$$\boldsymbol{\mu}(z_{m+1}) - s\boldsymbol{\sigma}(z_{m+1}) \leq \mathbf{y}_{m+1} \leq \boldsymbol{\mu}(z_{m+1}) + s\boldsymbol{\sigma}(z_{m+1}),$$

where s specifies a confidence level. This means that only data points are considered which do not deviate too strongly from the currently employed system model. The purpose of this second filter is to prevent sudden changes in the GP and enforce a gradual adaptation. In order to ensure rapid initial learning, we enable this second filter only after the dictionary is filled to 4/5 of its maximal size.

VI. Implementation & Experimental Results

We test the proposed learning-based control scheme on an autonomous race car, the details of which are outlined in the following sections before we present the experimental results.

A. Experimental Platform

The algorithm is implemented on the AMZ Driverless[1] vehicle *gotthard*, used in the 2018 Formula Student Driverless competition. *gotthard* is an electric 4WD race car with a full aerodynamic package, lightweight design and high efficiency, built by AMZ in 2016. The car enables superior lateral (aerodynamic grip), longitudinal (4WD traction, no gear shifts) and yaw (torque vectoring) acceleration and has successfully competed in autonomous racing competitions.

The vehicle is equipped with sensors for localization and state estimation, such as LiDAR, cameras, optical absolute speed sensor and INS, among others. Sensor data is processed on an onboard Intel i7-3612QE 2.1GHz main computer, which in addition runs the proposed control framework, as well as the *autonomous system* consisting of mapping, localization and state estimation, the details of which can be found in [19], [20] and [21], where the latter also provides additional information on the *nominal* control system. An Electronic Control Unit (ECU) is used as a real-time capable computer, which handles the low-level vehicle controllers. The low-level control loops consist of traction control (TC), torque vectoring (TV) and the four-wheel torque distribution based on the normal load of the wheels. The low-level controllers play a significant role in shaping the overall vehicle dynamics and ensure that a simple model as in (8) is adequate. The overall architecture is depicted in Figure 3. Note that since the data-point selection (Section V) is not time-critical, it runs in parallel with irregular timing.

[1] www.amzracing.ch

B. Controller Implementation

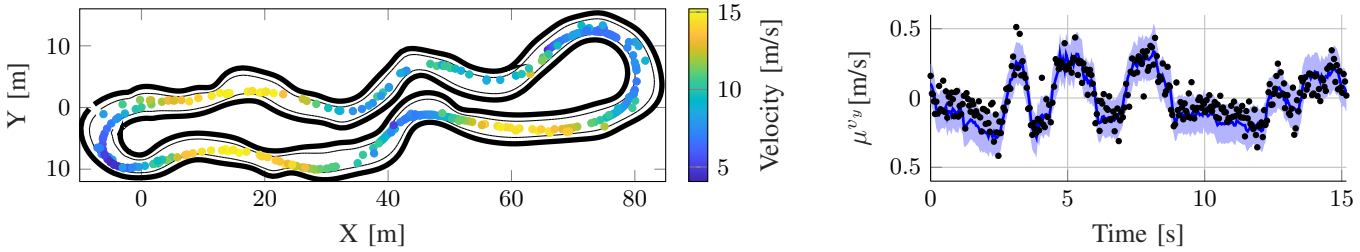
The GP-MPC problem in (14) is implemented with a prediction horizon of $N = 40$ at a sampling time of $T_s = 50\,\text{ms}$, resulting in a $2\,\text{s}$ look-ahead. We use dynamic constraint tightening for the first $N_{\text{shrink}} = 30$ time steps and consider 10 inducing inputs to reduce the complexity of GP evaluations. The FORCES Pro solver [22], [23] was used to solve the underlying optimization problem, in which the maximum number of iterations was limited to 60 to ensure consistent maximum solve times. A delay compensation of one time step is used to compensate for the solver computation time. In addition to the numeric solution of the MPC, a number of precomputations are executed, such as sparsification of the GP (6a) using dynamic inducing inputs, carried out at each time step. Thanks to a custom C++ sparse GP implementation, the precomputation of (6a) with the 10 inducing inputs placed equally along the previous solution trajectory is consistently carried out in under $3\,\text{ms}$, such that most of the time can be allocated to the numerical solution of the MPC optimization problem (14).

C. Experimental Setup

The experimental validation of the proposed control scheme is carried out by racing the car around a track of approximately $200\,\text{m}$ length, consisting of sharp hairpins, straights, long corners as well as chicanes, which is generated by placing traffic cones on an airfield and using the mapping procedure described in [20]. The resulting track is shown in Figure 4a. For safety reasons, we limit the top speed to $15\,\frac{\text{m}}{\text{s}}$ and the driver command T to ± 0.3 during the experiment.

The car starts racing with the *nominal* controller, meaning that all GP-dependent variables are set to zero in (14). All parameters of the *nominal* system model (8) were selected during extensive testing and fitted to experimental data. The nominal model and the corresponding cost function parameters are tuned conservatively to ensure safe driving behavior, since we cannot allow crashes or track constraint violations without aborting the race. Throughout the experiment, we set $\mathbf{y}_{\text{lim}}$ in the first filter to reject outliers greater than 3-σ of historic error data and $s = 1$ in the second filter. The dynamic constraint tightening in (11) is done with $\chi^2_2(p) = 1$. The data collected under the nominal controller is used to fill the dictionary with an initial set of 250 data points, corresponding to slightly less than two laps, after which the GP-based controller (14)

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LRA.2019.2926677, IEEE Robotics and Automation Letters



(a) Race track and corresponding location of dictionary points after 9 laps. The dictionary has high density around the sharp and challenging corners. The color gradient represents the velocity magnitude at which the point was recorded.

(b) Recorded model error in v_y (black dots) and GP error prediction. The blue line is the mean predicted error with 2-σ confidence intervals in shaded blue.

Fig. 4: Experimental results of learning-based control framework.

TABLE I: Experimental Results

Lap	Time [s]	$\overline{\|\mathbf{e}_{\text{nom}}\|}$	$\overline{\|\mathbf{e}_{\text{GP}}\|}$	$\|a\|_{\max}$ [g]	Dict. updates	1-σ [%]
1	20.19	0.16	-	1.52	178	-
2	20.29	0.18	-	1.49	65	-
3	19.16	0.19	0.15	2.05	31	65.42
4	18.80	0.23	0.15	2.01	16	68.66
5	18.47	0.23	0.16	2.02	15	68.31
6	18.44	0.24	0.15	2.00	11	69.07
7	17.98	0.23	0.15	1.82	8	68.15
8	18.47	0.24	0.17	2.15	9	68.55
9	18.25	0.24	0.16	2.10	11	68.81

TABLE II: Comparison of Data Selection Mechanisms

Scheme	$\overline{\|\mathbf{e}\|}$	$\|\mathbf{e}\|_{\max}$	$\overline{\|\sqrt{\text{diag}(\mathbf{\Sigma}^d)}\|}$	$\|\sqrt{\text{diag}(\mathbf{\Sigma}^d)}\|_{\max}$
Proposed	0.15	0.64	0.11	0.25
Random	0.16	0.70	0.23	0.36
Static	0.18	0.67	0.15	0.22

is activated. We found this initial data collection necessary to enable reliable behavior, which we could not ensure when using the GP correction based on very few data points. After the maximal dictionary size of 300 data points is reached, data-points are replaced as explained in Section V-A2 in order to offer the best model fit given new measurement data. The experiment was carried out for a total of 9 laps.

D. Results

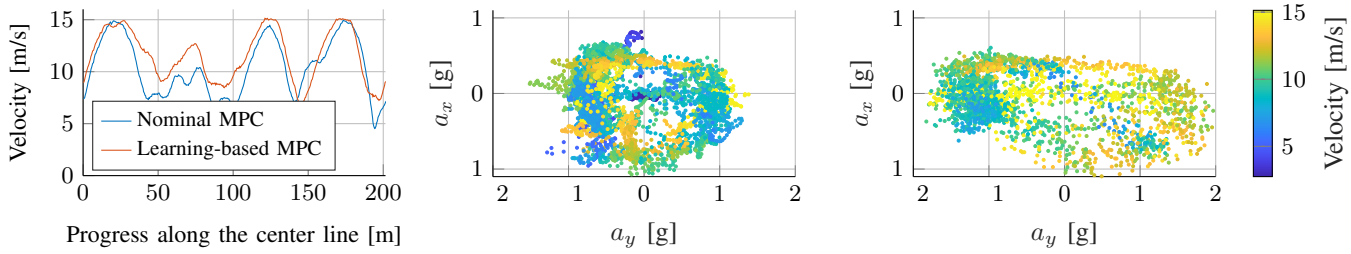
To quantify the performance of the proposed control scheme and the improvement due to the learning, we compare the lap times of the successive laps, summarized in Table I, where the dashed line highlights the activation of the learning-based model correction. It can be clearly observed that starting from lap times around $20.2\,\text{s}$ with the nominal controller, there is an immediate improvement when using the model correction in lap 3, which is further improved until settling around lap 5 at an average lap time of approximately $18.3\,\text{s}$, constituting an improvement of almost 10%.

The improvement is also shown by Figure 5a, where the first and last lap of the race are compared. The results show that the last lap is faster in almost every section of the track, indicating that the learning-based controller allows more aggressive driving. The increased aggressiveness is also visible in the observed accelerations, as shown in Table I for each lap and the GG diagrams in Figures 5b and 5c, showing the lateral and longitudinal acceleration during the race. The maximum lateral acceleration increases from $1.3\,\text{g}$ to $2.0\,\text{g}$ when using learning, which turns out to be higher than the lateral acceleration rating of the tires at $1.6\,\text{g}$, indicating that the learning-based controller is able to make use of the increased grip due to the aerodynamics package. In Figures 5b and 5c, the tight limits on the driver command are also clearly visible by the relatively low longitudinal accelerations.

In addition, we investigate the model learning performance, which is illustrated in Figure 4b showing the predicted model error in v_y and the actual encountered errors during the race. The results show that both the mean and uncertainty estimate of the GP provide a good fit of the true model error, enabling a safe performance increase. We quantify the learning performance by comparing the average 2-norm of the model error in each lap w.r.t. the nominal model, i.e. $\|\mathbf{e}_{\text{nom}}\| = \|\mathbf{x}_{k+1} - \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)\|$ and w.r.t. the corrected system dynamics, i.e. $\|\mathbf{e}_{\text{GP}}\| = \|\mathbf{x}_{k+1} - (\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{B}_d\tilde{\boldsymbol{\mu}}^d(\mathbf{z}_k))\|$, shown in Table I. It is evident, that the model learning is able to keep the average resulting model error $\overline{\|\mathbf{e}_{\text{GP}}\|}$ virtually constant under increasingly aggressive driving, whereas the nominal model error $\overline{\|\mathbf{e}_{\text{nom}}\|}$ increases by almost 40%. The uncertainty estimate of the GP fits the measurement data well, with about 65 to 69% of the measured deviation within the $1-\sigma$ confidence interval. Table I also provides information about the dictionary updates, showing that after a quick initial learning phase, the update rate converges to about 10 updates per lap.

Finally, we investigate the online learning mechanism by comparing the prediction error on the driven trajectory to the predictions with a *static* GP from a previous experiment, as well as to a random selection of data points in each lap from the thus far collected data, similar to [10]. The results are provided in Table II, showing the average prediction error over all laps $\overline{\|\mathbf{e}\|}$ and the average predicted standard deviation of the GP $\overline{\|\sqrt{\text{diag}(\mathbf{\Sigma}^d)}\|}$ along (6b), where the square root is taken element-wise. Additionally, we provide the maximum values over all laps. The results show that a *static* GP from a previous experiment results in increased errors of about 20%, demonstrating the need for a learning procedure during operation. We evaluate the *random* selection by averaging the results from 100 random seeds, resulting in

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LRA.2019.2926677, IEEE Robotics and Automation Letters



(a) Velocity profiles (b) GG-diagram: Nominal controller (c) GG-diagram: Learning-based controller

Fig. 5: Comparison of nominal (first lap) and learning-based controller (final lap).

a mean prediction error similar to our proposed approach. It is noticeable, however, that the random selection results in significantly higher uncertainty estimates, which are expected to result in performance deterioration due to the cautious nature of the controller. The employed data selection scheme results in an even distribution of data points along the race line, which are shown in Figure 4a for the last lap, providing an accurate and low-uncertainty estimate of the dynamics.

Overall, the results demonstrate that the proposed learning-based control scheme is able to improve from an initial nominal controller to achieve high-performance control while maintaining safe operation at all times.

VII. Conclusion

This paper has presented an MPC control approach for autonomous racing, which uses Gaussian Process regression to enhance a simple nominal model and improve racing performance. Based on the residual uncertainty of the Gaussian Process, constraints are dynamically tightened in order to achieve safe driving behavior. The data points used for GP predictions are selected online from the continuous stream of measurements based on an information gain criterion, enabling continuous learning during operation. The framework was tested on a full-size AMZ Driverless car, demonstrating significant performance improvements under the proposed learning-based controller with reductions in lap-time of 10%, while maintaining safety at all times.

Acknowledgment

We would like to thank the entire AMZ Driverless team for their outstanding work, and in particular Manuel Dangel who supported the first deployment of the nominal control framework on an older AMZ Driverless car.

References

[1] S. Thrun *et al.*, "Stanley: The robot that won the DARPA Grand Challenge," *Journal of Field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.
[2] A. Hofacker, "Let's go driverless: Challenges of the first season," *ATZextra worldwide*, vol. 22, no. 2, pp. 22–27, 2017.
[3] 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," *Trans. Intelligent Vehicles*, vol. 1, no. 1, pp. 33–55, 2016.
[4] A. Liniger, A. Domahidi, and M. Morari, "Optimization-based autonomous racing of 1:43 scale RC cars," *Optimal Control Applications and Methods*, vol. 36, no. 5, pp. 628–647, 2015.
[5] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Aggressive driving with model predictive path integral control," *Int. Conf. Robotics and Automation*, pp. 1433–1440, 2016.
[6] U. Rosolia, X. Zhang, and F. Borrelli, "Data-driven predictive control for autonomous systems," *Annu. Review Control, Robotics, and Autonomous Systems*, vol. 1, pp. 259–286, 2018.
[7] L. Hewing, A. Liniger, and M. N. Zeilinger, "Cautious NMPC with Gaussian process dynamics for autonomous miniature race cars," *European Control Conf.*, pp. 1341–1348, 2018.
[8] L. Hewing and M. N. Zeilinger, "Cautious model predictive control using Gaussian process regression," *arXiv:1705.10702*, 2017.
[9] C. J. Ostafew, A. P. Schoellig, T. D. Barfoot, and J. Collier, "Learning-based nonlinear model predictive control to improve vision-based mobile robot path tracking," *J. Field Robotics*, vol. 33, no. 1, pp. 133–152, 2016.
[10] C. D. McKinnon and A. P. Schoellig, "Experience-based model selection to enable long-term, safe control for repetitive tasks under changing conditions," *Int. Conf. Intelligent Robots and Systems*, pp. 2977–2984, 2018.
[11] C. McKinnon and A. P. Schoellig, "Learn fast, forget slow: Safe predictive learning control for systems with unknown and changing dynamics performing repetitive tasks," *Robotics and Automation Letters*, vol. 4, no. 2, pp. 2180–2187, 2019.
[12] M. Brunner, U. Rosolia, J. Gonzales, and F. Borrelli, "Repetitive learning model predictive control: An autonomous racing example," *Conf. Decision and Control*, pp. 2545–2550, 2017.
[13] D. Nguyen-Tuong and J. Peters, "Incremental online sparsification for model learning in real-time robot control," *Neurocomputing*, vol. 74, no. 11, pp. 1859–1867, 2011.
[14] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, 2006.
[15] J. Quiñonero-Candela, C. E. Rasmussen, and C. K. I. Williams, "Approximation methods for Gaussian process regression," in *Large-Scale Kernel Machines*. MIT Press, 2007, pp. 203–223.
[16] E. Snelson and Z. Ghahramani, "Sparse Gaussian processes using pseudo-inputs," *Adv. Neural Information Processing Systems*, pp. 1257–1264, 2006.
[17] H. B. Pacejka and E. Bakker, "The magic formula tyre model," *Vehicle system dynamics*, vol. 21, no. S1, pp. 1–18, 1992.
[18] J. V. Carrau, A. Liniger, X. Zhang, and J. Lygeros, "Efficient implementation of randomized MPC for miniature race cars," *European Control Conf.*, pp. 957–962, 2016.
[19] M. I. Valls *et al.*, "Design of an autonomous racecar: Perception, state estimation and system integration," *Int. Conf. Robotics and Automation*, pp. 2048–2055, 2018.
[20] N. B. Gosala *et al.*, "Redundant perception and state estimation for reliable autonomous racing," *arXiv:1809.10099*, 2018.
[21] J. Kabzan *et al.*, "Amz driverless: The full autonomous racing system," *arXiv:1905.05150*, 2019.
[22] A. Domahidi and J. Jerez, "FORCES Professional," embotech GmbH (http://embotech.com/FORCES-Pro), 2014.
[23] A. Zanelli, A. Domahidi, J. Jerez, and M. Morari, "FORCES NLP: an efficient implementation of interior-point methods for multistage nonlinear nonconvex programs," *Int. J. Control*, 2017.