Robotica (2023), **41**, pp. 3040–3058
doi:10.1017/S0263574723000863

CAMBRIDGE UNIVERSITY PRESS

RESEARCH ARTICLE

Exploration-exploitation-based trajectory tracking of mobile robots using Gaussian processes and model predictive control

Hannes Eschmann, Henrik Ebel and Peter Eberhard

Institute of Engineering and Computational Mechanics, University of Stuttgart, Stuttgart, Germany
Corresponding author: Peter Eberhard; Email: peter.eberhard@itm.uni-stuttgart.de

Received: 16 March 2023; **Revised:** 10 May 2023; **Accepted:** 28 May 2023; **First published online:** 30 June 2023

Keywords: mobile robots; control of robotic systems; uncertainty; Gaussian process; model predictive control

Abstract

Mobile robots are a key component for the automation of many tasks that either require high precision or are deemed too hazardous for human personnel. One of the typical duties for mobile robots in the industrial sector is to perform trajectory tracking, which involves pursuing a specific path through both space and time. In this paper, an iterative learning-based procedure for highly accurate tracking is proposed. This contribution shows how data-based techniques, namely Gaussian process regression, can be used to tailor a motion model to a specific reoccurring reference. The procedure is capable of explorative behavior meaning that the robot automatically explores states around the prescribed trajectory, enriching the data set for learning and increasing the robustness and practical training accuracy. The trade-off between highly accurate tracking and exploration is done automatically by an optimization-based reference generator using a suitable cost function minimizing the posterior variance of the underlying Gaussian process model. While this study focuses on omnidirectional mobile robots, the scheme can be applied to a wide range of mobile robots. The effectiveness of this approach is validated in meaningful real-world experiments on a custom-built omnidirectional mobile robot where it is shown that explorative behavior can outperform purely exploitative approaches.

1. Introduction

Mobile robots are widely used in various industries due to their automation capabilities. In many applications, such as parcel centers, these robots follow the same path repeatedly. Additionally, mobile robots can be utilized as measurement platforms, for example, for automating measurement campaigns of wireless signals during product evaluation and development. Extremely high accuracy and reproducibility are critical in these cases. Data obtained from driving these periodic paths can be utilized to enhance the tracking capabilities of mobile robots through an approach that combines optimization and data analysis. As more input–output data become available, the reference is adapted in an iterative fashion lowering the tracking error (exploitation) while at the same time exploring previously unseen reference solutions (exploration).

Exploration-exploitation itself is associated with various fields ranging from psychology and economics to computer science, most notably reinforcement learning. The core idea is that an agent needs to *exploit* its knowledge to maximize a certain reward but at the same time needs to *explore* to gather knowledge to make better decisions in the long run. The dilemma known as the exploration-exploitation trade-off arises; in a complex environment, following only one of these two goals will in general not maximize the reward [1].

In the proposed procedure, this exploration-exploitation trade-off is done automatically in a controlled and safe manner using a novel offline optimization-based reference generator and a special learning cost function [2]. The main advantage of explorative over purely exploitative approaches is that they are less

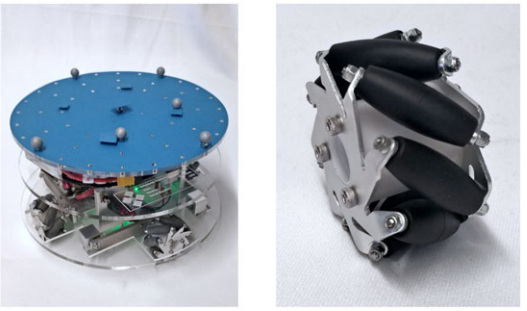


Figure 1. *The custom-built omnidirectional mobile robot platform (left) and its special Mecanum wheels (right).*

prone to getting stuck in local optima, which in the case of mobile robots can reliably improve the tracking performance. A practical tracking accuracy in the range of a few millimeters becomes attainable even when using comparatively cheap robotic hardware, which can further boost the employment of mobile robots for the automation of various tasks in industrial practice. While the approach is conceptually similar to the work of ref. [3], trajectory tracking instead of path following is considered and the model in this work is tailored to a specific reference trajectory, reducing the tracking error further without significantly increasing the computational burden. Contrary to comparable iterative approaches [4], the scheme is model- and optimization-based and respects the physical constraints of the robot using a model predictive controller (MPC).

This study is focused on omnidirectional mobile robots. The proposed procedure, however, is applicable to general robots in a wide range of applications. In Section 2, the robotic hardware is presented. Section 3 deals with the necessary Gaussian process fundamentals. Subsequently, in Section 4, the model setup is explained in detail, whereas in Section 5, the proposed reference generator and the corresponding model-based controller are introduced. Finally, in Section 6, the effectiveness of the approach is shown in real-world trajectory tracking experiments of an omnidirectional mobile robot. To the best of the authors' knowledge, the approach taken is conceptually novel and unique in the very high fidelity it achieves. This paper is an invited extension of ref. [5].

2. The omnidirectional mobile robot

There are various methods to achieve omnidirectional motion, but this study employs a specialized type of wheel called the Mecanum wheel. This type of wheel has several rollers attached to its circumference at an angle. When using Mecanum wheels for omnidirectional mobile robots, there are different wheel arrangements available, with rectangular and radial arrangements being the most popular. The custom-built robot used in the hardware experiments of Section 6 adopts the radial arrangement, which utilizes four equally spaced wheels as shown in Fig. 1. The robot receives inputs in the form of $\boldsymbol{u} := [v^{\mathrm{x}}\ v^{\mathrm{y}}\ \omega]^{\mathsf{T}}$, which refers to the desired translational velocities in the robot's frame of reference, denoted as v^{x} and v^{y}, and the angular velocity around the vertical axis, represented as ω. To compute the four resulting desired wheel speeds Ω_i for wheel i, the relationship

$$\boldsymbol{\Omega} := \begin{bmatrix} \Omega_1 \\ \Omega_2 \\ \Omega_3 \\ \Omega_4 \end{bmatrix} = \frac{1}{R} \begin{bmatrix} 1 & 1 & -\ell \\ 1 & -1 & -\ell \\ -1 & 1 & -\ell \\ -1 & -1 & -\ell \end{bmatrix} \begin{bmatrix} v^{\mathrm{x}} \\ v^{\mathrm{y}} \\ \omega \end{bmatrix} =: \boldsymbol{D}\boldsymbol{u} \tag{1}$$

is used, where $R = 3$ cm is the radius of a wheel and $\ell = 11.5$ cm is the distance between the robot's and each wheel's center in the motion plane. Over time, relationship (1) can be utilized to compute the corresponding set points for each wheel's individual proportional-integral-derivative controller, which runs at a frequency of 100 Hz. As the robot should rotate independently from its translational velocity

https://doi.org/10.1017/S0263574723000863 Published online by Cambridge University Press

$[v^x\ v^y]^\mathsf{T}$, the input used to compute (1) is rotated by an angle of $-\omega/100\,\text{Hz}$ about the vertical axis after each cycle unless a new input is received. Without this counter-rotation, the robot would drive arcs, similar to a differential-drive [6] or car-like mobile robot, which would not fully utilize the robot's omnidirectional capabilities [7]. The custom control software runs on an onboard BeagleBone Blue Linux-based computer with integrated motor drivers. The input $\boldsymbol{u}$ is updated through a wireless network at a frequency of 5 Hz. The robot's pose, which consists of its position within the movement plane in the inertial frame of reference and its orientation, is determined with a frequency of 100 Hz by an external motion tracking system consisting of five Optitrack Prime 13W cameras. The pose measurements and the inputs to the mobile robot are communicated between the different machines via a local network using the Lightweight Communications and Marshalling library [8].

3. Gaussian process regression

Throughout this paper, a suitable regression method is needed to deal with collected input–output data. We opt for nonparametric Gaussian processes (GPs) [9] as the regression technique, as opposed to parametric regression with a fixed number of basis functions [10], because the structure of the mismatches is unknown a priori and the right choice of basis functions in high dimensional input spaces can be challenging. Neural Networks, on the other hand, while being less restrictive, require large amounts of training data to fit the high number of unknown parameters [11].

A typical scenario for a regression problem involves an unknown scalar function f and a dataset consisting of N_f noisy observations

$$y_i = f\left(\boldsymbol{x}_i^\text{f}\right) + \varepsilon, \quad \varepsilon \sim \mathcal{N}\left(0, \sigma_\text{n}^2\right), \quad i \in \left\{1, \ldots, N_f\right\}, \tag{2}$$

at the corresponding input $\boldsymbol{x}_i^\text{f} \in \mathbb{R}^n$. Here, the additive noise ε follows an independent, identically distributed Gaussian distribution with zero mean and noise variance σ_n^2. The provided input–output tuples are collected in the data set $\mathcal{D}_f = \left\{\left(\boldsymbol{x}_i^\text{f}, y_i\right), i \in \left\{1, \ldots, N_f\right\}\right\}$.

A Gaussian process can be seen as a generalization of Gaussian distributions to the function space. Any GP is uniquely defined by a mean function $m : \mathbb{R}^n \to \mathbb{R}$ and a covariance function $k : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$. Usually, for notational simplicity the mean function is taken to be zero $m(\boldsymbol{x}) \equiv 0$. The squared exponential (SE) kernel with inputs $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^n$ is a popular choice for the covariance function k

$$k_\text{SE}(\boldsymbol{x}, \boldsymbol{x}') = \sigma_\text{SE}^2 \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}')^\mathsf{T} \boldsymbol{A} (\boldsymbol{x} - \boldsymbol{x}')\right). \tag{3}$$

Here, σ_SE^2 is the signal variance and typically $\boldsymbol{A}$ is chosen as $\boldsymbol{A} = \text{diag}\left(\ell_1^{-2}, \ldots, \ell_\text{n}^{-2}\right)$. The parameters ℓ_i are called the length scales and control how much the kernel function will be affected by changes in each input dimension. Therefore, the length scales play a crucial role in determining the smoothness of the regression function and are inversely related to the importance of each input component. In case the function (2) has a scalar input $n = 1$ and is periodic with (known) fixed period P_PER, a periodic kernel can be used. A suitable periodic kernel can be defined as ref. [12]

$$k_\text{PER}(x, x') = \sigma_\text{PER}^2 \exp\left(-\frac{2\sin^2\left(\frac{\pi}{P_\text{PER}}(x - x')\right)}{\ell_\text{PER}^2}\right). \tag{4}$$

Again, σ_PER^2 is the signal variance and ℓ_PER is the length scale. The periodic kernel can be combined with other kernels, for example, by multiplication with the SE kernel in case the function f has a multidimensional input and is only periodic in one of its arguments. The posterior distribution at a given input $\boldsymbol{x}^*$ is Gaussian $\mathcal{N}(\mu(\boldsymbol{x}^*), \sigma^2(\boldsymbol{x}^*))$ with mean μ and variance σ^2. In the following, the ith component of a vector-valued variable is indicated via $[\bullet]_i$ while $[\bullet]_{ij}$ is the entry of a matrix in row i and column j.

https://doi.org/10.1017/S0263574723000863 Published online by Cambridge University Press

By using the notation $[\boldsymbol{K}_{\mathrm{ab}}]_{ij} = \boldsymbol{k}\left(\boldsymbol{x}_i^{\mathrm{a}}, \boldsymbol{x}_j^{\mathrm{b}}\right), \left[\boldsymbol{k}_{*\,\mathrm{b}}\right]_i = \boldsymbol{k}\left(\boldsymbol{x}^*, \boldsymbol{x}_i^{\mathrm{b}}\right)$ and $[\boldsymbol{y}]_i = y_i$, the posterior mean and variance are given by

$$\mu(\boldsymbol{x}^*) = \boldsymbol{k}_{*\mathrm{f}}\left(\boldsymbol{K}_{\mathrm{ff}} + \sigma_{\mathrm{n}}^2 \boldsymbol{I}\right)^{-1} \boldsymbol{y}, \tag{5a}$$

$$\sigma^2(\boldsymbol{x}^*) = \boldsymbol{k}_{**} - \boldsymbol{k}_{*\mathrm{f}}\left(\boldsymbol{K}_{\mathrm{ff}} + \sigma_{\mathrm{n}}^2 \boldsymbol{I}\right)^{-1} \boldsymbol{k}_{*\mathrm{f}}^{\mathsf{T}}. \tag{5b}$$

Evaluating the mean and variance at a given input requires the inversion of a matrix of size $N_f \times N_f$, which is computationally expensive for large N_f and of order $\mathcal{O}\left(N_f^3\right)$. However, with some pre-computations, the order of evaluating the mean and variance can be reduced to $\mathcal{O}\left(N_f\right)$ and $\mathcal{O}\left(N_f^2\right)$, respectively [13].

For the SE kernel with automatic relevance detection (ARD), the kernel hyperparameters, namely the signal σ_{SE}^2 and noise σ_{n}^2 variance and length scales ℓ_i, are optimized by solving a maximum log-likelihood problem, see ref. [9] for details. However, the hyperparameters can also be chosen with the help of expert knowledge.

Sparse Gaussian processes

The computational effort of evaluating the posterior mean and variance scales with the number of data points N_f. This can limit the use of GPs for larger data sets $\mathcal{D}_f$ or real-time applications and prohibit an effective usage for this paper's purposes. Therefore, several approaches have been proposed to overcome this issue [14–16], with one of them being sparse approximations (sGP) of GPs; see ref. [13] for an overview. In this paper, the variational free energy (VFE) [17] approach is used, which provides a computationally efficient method for approximating the posterior distribution of the GP, with the potential to reproduce the full GP exactly. This approximation technique relies on so-called pseudo-inputs. The number of pseudo-inputs M is usually chosen to be significantly lower than the number of training points $M \ll N_f$. Subsequently, the notation $\boldsymbol{Q}_{\mathrm{ab}} = \boldsymbol{K}_{\mathrm{au}} \boldsymbol{K}_{\mathrm{uu}}^{-1} \boldsymbol{K}_{\mathrm{ub}}$ is used, where index u refers to the superscript of the pseudo-inputs $\boldsymbol{x}_i^{\mathrm{u}}$, $i \in \{1, \ldots, M\}$ and $\boldsymbol{K}_{\mathrm{uu}}$ is the corresponding covariance matrix. For the posterior distribution, when using the VFE technique, one now obtains $\mathcal{N}\left(\mu_{\mathrm{s}}(\boldsymbol{x}^*), \sigma_{\mathrm{s}}^2(\boldsymbol{x}^*)\right)$ with

$$\mu_{\mathrm{s}}(\boldsymbol{x}^*) = \boldsymbol{Q}_{*\mathrm{f}}\left(\boldsymbol{Q}_{\mathrm{ff}} + \sigma_{\mathrm{n}}^2 \boldsymbol{I}\right)^{-1} \boldsymbol{y}, \tag{6a}$$

$$\sigma_{\mathrm{s}}^2(\boldsymbol{x}^*) = \boldsymbol{K}_{**} - \boldsymbol{Q}_{*\mathrm{f}}\left(\boldsymbol{Q}_{\mathrm{ff}} + \sigma_{\mathrm{n}}^2 \boldsymbol{I}\right)^{-1} \boldsymbol{Q}_{\mathrm{f}*}. \tag{6b}$$

After some pre-computations, the order of evaluating the mean and variance is reduced to $\mathcal{O}(M)$ and $\mathcal{O}(M^2)$, respectively. The location of the pseudo-inputs can be chosen as a subset of the actual training inputs, for example, by Greedy optimization of their location, or they can be included in the overall hyperparameter optimization.

4. Data-based model of the mobile robot

A kinematic model of the planar omnidirectional mobile robot is introduced as a nominal baseline. This model is commonly used for robotics applications and will subsequently serve as a starting point for learning purposes. Thus, the kinematic discrete-time relationship that describes the position at time step $t \in \mathbb{N}_0$ in the inertial frame of reference is given by

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + T \boldsymbol{S}_{\theta_t} \boldsymbol{u}_t =: \boldsymbol{f}_{\mathrm{n}}(\boldsymbol{x}_t, \boldsymbol{u}_t), \tag{7}$$

where $\boldsymbol{x}_t := [x_t\ y_t\ \theta_t]^{\mathsf{T}}$ is the pose of the mobile robot at time t, that is, its position $[x_t\ \ y_t]^{\mathsf{T}}$ in the inertial frame of reference and its orientation θ_t. The rotation matrix

$$\boldsymbol{S}_{\theta_t} := \begin{bmatrix} \cos(\theta_t) & -\sin(\theta_t) & 0 \\ \sin(\theta_t) & \cos(\theta_t) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{8}$$

https://doi.org/10.1017/S0263574723000863 Published online by Cambridge University Press

describes a positive rotation about the robot's vertical axis by the angle θ_t. The notation $\boldsymbol{v}_t := \boldsymbol{v}(t)$ is used as a shorthand, unless the explicit time dependency needs to be emphasized. A sampling time of $T = 0.2\text{s}$ is selected. The simple kinematic model (7) is only suitable for reasonable input combinations. Thus, the wheel speeds and their rates of change are restricted

$$\|\boldsymbol{\Omega}_t\|_\infty \overset{(1)}{=} \|\boldsymbol{D}\boldsymbol{u}_t\|_\infty \leq \Omega_{\max}, \tag{9a}$$

$$\|\boldsymbol{\Omega}_t - \boldsymbol{\Omega}_{t-1}\|_\infty \overset{(1)}{=} \|\boldsymbol{D}(\boldsymbol{u}_t - \boldsymbol{u}_{t-1})\|_\infty \leq \Delta\Omega_{\max}, \tag{9b}$$

$$|\omega_t| \leq \omega_{\max}, \tag{9c}$$

where $\Omega_{\max} = 20\ 1/\text{s}$ and $\Delta\Omega_{\max} = 3.3\ 1/\text{s}$ are the maximum wheel speed and the wheel speeds' maximal change between two time instances, respectively. Additionally, the maximum magnitude of the angular velocity ω_t of the robot is limited to $\omega_{\max} = 0.5\ \text{rad/s}$. The maximum norm is indicated by $\|\bullet\|_\infty$. These constraints are imposed point-wise at each time step t. If a different type of mobile robot is used, (7) has to be replaced with the corresponding (kinematic) model.

4.1. Including data as prior knowledge

In the hardware experiments, it was observed that the mobile robot exhibited interesting nonlinear behavior that was not captured by the simple kinematic model (7). To account for this, an augmented model was introduced that includes a data-based nonlinear function $\boldsymbol{g} = \begin{bmatrix} g^{\text{x}} & g^{\text{y}} & g^{\omega} \end{bmatrix}^\mathsf{T}$. This function captures systematic mismatches between the actual system dynamics and the nominal model in the robot's frame of reference. The unknown nonlinear function $\boldsymbol{g}(\boldsymbol{a}_t)$ is assumed to be dependent on the current and previous inputs, that is,

$$\boldsymbol{a}_t^\mathsf{T} := \begin{bmatrix} \boldsymbol{u}_t^\mathsf{T} & \boldsymbol{u}_{t-1}^\mathsf{T} \end{bmatrix}. \tag{10}$$

This choice is motivated by experience with the robotic hardware. Based on (9), the set of admissible inputs $\mathcal{A} \subset \mathbb{R}^6$ to the function $\boldsymbol{g}$ can be defined as

$$\mathcal{A} := \left\{ \begin{bmatrix} \boldsymbol{u}_t \\ \boldsymbol{u}_{t-1} \end{bmatrix} \middle| \begin{array}{c} \boldsymbol{u}_t \in \mathbb{R}^3, \\ \boldsymbol{u}_{t-1} \in \mathbb{R}^3, \\ \|\boldsymbol{D}\boldsymbol{u}_t\|_\infty \leq \Omega_{\max}, \\ \|\boldsymbol{D}\boldsymbol{u}_{t-1}\|_\infty \leq \Omega_{\max}, \\ \|\boldsymbol{D}(\boldsymbol{u}_t - \boldsymbol{u}_{t-1})\|_\infty \leq \Delta\Omega_{\max}, \\ [\boldsymbol{u}_t]_3| \leq \omega_{\max}, \\ [\boldsymbol{u}_{t-1}]_3| \leq \omega_{\max} \end{array} \right\}, \tag{11}$$

that is, two consecutive inputs that each satisfy the condition for the maximum wheel speed and angular velocity as well as the condition for the change of wheel speed.

The training data $\mathcal{D}_g := \left\{ \left(\boldsymbol{a}(i), \boldsymbol{e}_g(i) \right), i \in \left\{ 1, \ldots, N_g \right\} \right\}$ are obtained by considering the model mismatch to the nominal model between two consecutive time steps in the robot frame of reference

$$\boldsymbol{e}_g(t) = \frac{1}{T} \boldsymbol{S}_{\theta(t)}^\mathsf{T} \left(\boldsymbol{x}(t+1) - \boldsymbol{f}_{\text{n}}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \right). \tag{12}$$

Measurements of the position and orientation of the robot are taken via an optical tracking system with high accuracy. These measurements could be available, for example, from the prior operation of the hardware. In our case, to ensure a good generalization to all possible input combinations satisfying (9) while sufficiently covering the corresponding input space, we generate this data using a special automatic model tuning procedure from refs. [18, 19]. To generate the training data in the hardware experiment, the hit-and-run sampler from ref. [20] is used to generate uniformly distributed random points inside the six-dimensional convex polytope $\mathcal{A}$, see Fig. 2. These samples are then concatenated in a time series and applied to the robot. For more information on the specific training procedure, it is referred to ref. [19]. Some of the trajectories that are used in the training process are depicted in Fig. 3.

https://doi.org/10.1017/S0263574723000863 Published online by Cambridge University Press

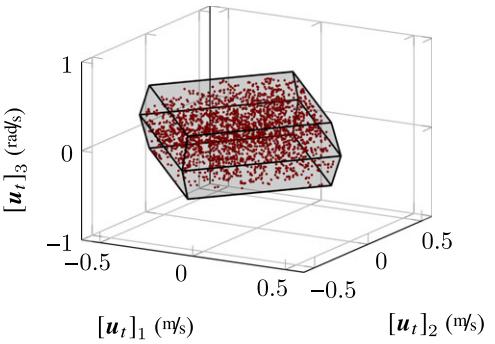


Figure 2. *The set $\mathcal{A}$ for the input $\boldsymbol{u}_t$ (its first three components) with the samples used for the training procedure in red. The set for the corresponding input $\boldsymbol{u}_{t-1}$ and the input rate $\boldsymbol{u}_t - \boldsymbol{u}_{t-1}$, while conceptually similar, are not depicted.*

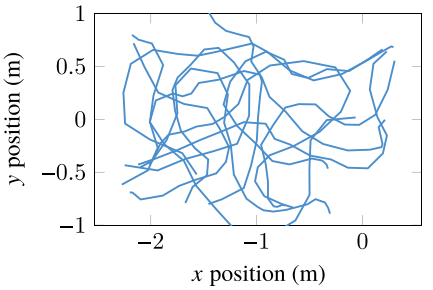


Figure 3. *Selection of some of the training trajectories used for the data generation. The orientation of the robot is not depicted.*

Each component of the unknown function is then approximated using an individual Gaussian process

$$\boldsymbol{g}(\boldsymbol{a}_t) \sim \mathcal{N}\big(\boldsymbol{\mu}_g(\boldsymbol{a}_t), \boldsymbol{\Sigma}_g(\boldsymbol{a}_t)\big) \tag{13}$$

with posterior mean $\boldsymbol{\mu}_g = \left[\mu_g^{\mathrm{x}}\ \mu_g^{\mathrm{y}}\ \mu_g^{\omega}\right]^{\mathsf{T}}$ and diagonal variance $\boldsymbol{\Sigma}_g$, computed using (5).

In the following, $M_{\mathrm{x}} = M_{\mathrm{y}} = 150$ pseudo-inputs are used (see Section 3) for g^{x} and g^{y}, whereas for g^{ω}, fewer points $M_{\omega} = 70$ suffice, based on a convergence analysis of the available data. The size of the overall data set is $N_g \approx 2500$. We opt for the SE-ARD kernel from (3) as covariance function. The GPML Matlab toolbox [21] is used to train the sparse GPs. Thereby, the location of the pseudo-inputs is included in the optimization. The gradient-based hyperparameter optimization is randomly initialized various times to avoid over- or under-fitting caused by local optima.

The data-based model of the omnidirectional robot using the information gained during previous operation (which was generated using the training procedure in this case) is defined as

$$\boldsymbol{x}_{t+1} = \boldsymbol{f}_{\mathrm{n}}(\boldsymbol{x}_t, \boldsymbol{u}_t) + T\boldsymbol{S}_{\theta_t}\boldsymbol{\mu}_g(\boldsymbol{a}_t) =: \boldsymbol{f}_g(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{a}_t). \tag{14}$$

Here, only the mean predictions of the GPs are used. While, in this paper, Gaussian process regression and the special automatic model tuning procedure from ref. [18] is used, the model mismatch could be trained using almost any other regression technique. After several hundred hours of operational time of the hardware on different trajectories and millions of data points in $\mathcal{D}_g$ for example, it is more efficient to switch to parametric regression approaches that can cope with large amounts of data such as neural networks [11].

https://doi.org/10.1017/S0263574723000863 Published online by Cambridge University Press

Using data-based dynamical models of the mobile robot

Throughout this study, kinematic models are employed. The primary reason is that not all (commercially available) mobile robots allow the user to command specific wheel torques or motor currents directly. In order to keep the results applicable to a wide range of robots, this work focuses on kinematic models with (wheel) speeds as inputs. It is also possible to extend the framework to dynamical robot models, for example, obtained when modeling the robot as a rigid multi-body system via the Newton–Euler approach or Lagrange formulation [22]. However, with dynamical models, the specific hardware setup and the sources of the model mismatch have a more significant influence on the approach one may take, for example, whether the model mismatch $\boldsymbol{g}$ is only added on position or also on velocity level. In order to obtain an accurate model, it may also be necessary to measure or estimate the velocity of the robot chassis and the wheels' angular velocities to predict the robot's state precisely.

4.2. Learning trajectory-specific mismatches

The model (14) can predict the robot's position for general input combinations. In the upcoming section, trajectory tracking will be considered. To that end, it is assumed that the robot needs to follow a specific reference trajectory $\boldsymbol{x}_t^{\mathrm{r}}$ for the state and $\boldsymbol{u}_t^{\mathrm{r}}$ for the input. To obtain an even higher accuracy for specific, a priori known trajectories, subsequently a more specialized model based on the data-driven model (14) is developed. Thus, a new function denoted as $\boldsymbol{h}(\boldsymbol{p}_t)$ with corresponding input $\boldsymbol{p}_t$ is introduced to address discrepancies that are unique to the current trajectory, such as the impact of uneven terrain or discrepancies that arise over longer time periods and cannot be accounted for by $\boldsymbol{g}$. This will further improve the accuracy of the model for the specific reference trajectory.

To learn the trajectory-dependent model mismatch, one may generally assume that the current mismatch might, classically, depend on the robot's current state and the corresponding input $\boldsymbol{p}_t^{\mathsf{T}} = \begin{bmatrix}\boldsymbol{x}_t^{\mathsf{T}} & \boldsymbol{u}_t^{\mathsf{T}}\end{bmatrix}$, which would not directly fit the idea of learning a trajectory-specific mismatch or mismatches that cannot be represented using only the chosen set of states [3, 23]. However, assuming that, due to the choice and consideration of $\boldsymbol{g}$, the robot's state and input will generally already evolve in the vicinity of the state and input reference $\left(\boldsymbol{x}_t \approx \boldsymbol{x}_t^{\mathrm{r}} \text{ and } \boldsymbol{u}_t \approx \boldsymbol{u}_t^{\mathrm{r}}\right)$, we instead subsequently assume a dependency on the current input reference and time, that is, $\boldsymbol{p}_t^{\mathsf{T}} := \begin{bmatrix}\boldsymbol{u}_t^{\mathrm{r}\mathsf{T}} & t\end{bmatrix}$. Thus, any state dependency of the mismatch (and other, more structurally profound model mismatches) will be purely taken into account through an explicit dependency on time; a dependency on the actual input is replaced by a dependency on the input reference. However, the input and state references need to be calculated with a model of the robot, preferably with the most accurate one available. Hence, it makes sense to use the most detailed model, taking into account trajectory-specific model mismatches, for the calculation of the reference. Yet, as designed, that model depends on the reference. Therefore, subsequently, an iterative approach is used. As more and more sets of data become available, a candidate for the trajectory-specific mismatch is calculated, which is used to calculate a new input and state reference. On the basis of the latter, with new data from a trajectory tracking experiment using a controller with the updated reference and model, an updated candidate for the trajectory-specific mismatch is calculated, and so forth. The resulting iterative procedure is visualized in Fig. 4.

Each component of the function $\boldsymbol{h} = \begin{bmatrix}h^{\mathrm{x}} & h^{\mathrm{y}} & h^{\omega}\end{bmatrix}^{\mathsf{T}}$ is, again, approximated by a Gaussian process

$$\boldsymbol{h} \sim \mathcal{N}\left(\boldsymbol{\mu}_{\boldsymbol{h}}(\boldsymbol{p}_t), \boldsymbol{\Sigma}_{\boldsymbol{h}}(\boldsymbol{p}_t)\right) \tag{15}$$

with mean prediction $\boldsymbol{\mu}_{\boldsymbol{h}} = \begin{bmatrix}\mu_{\boldsymbol{h}}^{\mathrm{x}} & \mu_{\boldsymbol{h}}^{\mathrm{y}} & \mu_{\boldsymbol{h}}^{\omega}\end{bmatrix}^{\mathsf{T}}$ and diagonal posterior variance $\boldsymbol{\Sigma}_{\boldsymbol{h}} = \mathrm{diag}\left(\sigma_{\mathrm{x}}^2, \sigma_{\mathrm{y}}^2, \sigma_{\omega}^2\right)$, each computed using (5). Apart from their high data efficiency, the advantage of using GPs as the regression technique that one also obtains an estimate of the remaining uncertainty of the model. This will become useful for the exploration-based approach taken in Section 5.1.

For the dynamics, taking the trajectory-specific mismatches into account, only the mean prediction $\boldsymbol{\mu}_{\boldsymbol{h}}$ of the GPs is considered

$$\boldsymbol{x}_{t+1} = \boldsymbol{f}_{\boldsymbol{g}}(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{a}_t) + T\boldsymbol{S}_{\theta_t}\boldsymbol{\mu}_{\boldsymbol{h}}(\boldsymbol{p}_t) := \boldsymbol{f}_{\boldsymbol{gh}}(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{a}_t, \boldsymbol{p}_t). \tag{16}$$

https://doi.org/10.1017/S0263574723000863 Published online by Cambridge University Press

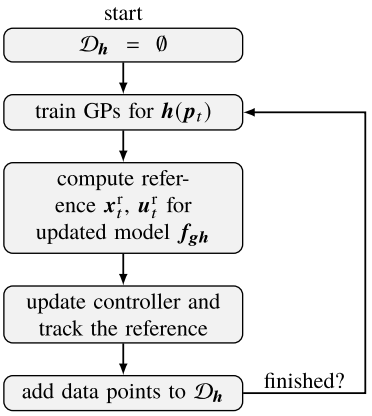


Figure 4. *Visualization of the proposed iterative procedure.*

The GPs for the function $\boldsymbol{h}$ are trained using the corresponding model mismatch $\boldsymbol{e_h}(t)$, that is,

$$\boldsymbol{e_h}(t) := \frac{1}{T}\boldsymbol{S}_{\theta(t)}^{\mathsf{T}}\big(\boldsymbol{x}(t+1) - \boldsymbol{f_g}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{a}(t))\big)\,. \tag{17}$$

It should be noted that the error $\boldsymbol{e_h}(t)$, calculated in the robot's frame of reference, pertains to the baseline model $\boldsymbol{f_g}$ rather than the deviation from the state reference. For the training data $\mathcal{D}_h := \big\{\big(\boldsymbol{p}(i), \boldsymbol{e_h}(i)\big), i \in \{1, \ldots, N_h\}\big\}$ containing N_h data points, the robot drives N^{lap} laps for each reference trajectory to reduce the effect of random disturbances. Training data that was gathered using a specific input reference can be added to $\mathcal{D}_h$. The GPs approximating $\boldsymbol{h}$ can be trained, and the input $\boldsymbol{u}^{\text{r}}$ and state $\boldsymbol{x}^{\text{r}}$ references are subsequently adapted for the updated model $\boldsymbol{f_{gh}}$, see Fig. 4. The idea is that, as more data $\mathcal{D}_h$ for different references become available, the model $\boldsymbol{f_{gh}}$ is improved, further reducing the tracking error. As in the first iteration, no trajectory-specific information is available ($\mathcal{D}_h = \emptyset$), one obtains $\boldsymbol{\mu_h} \equiv \boldsymbol{0}$, and therefore $\boldsymbol{f_{gh}} \equiv \boldsymbol{f_g}$. In Section 5.2, the state and input reference are determined using an optimization-based approach.

5. Trajectory tracking of periodic references

When it comes to trajectory tracking, robots are typically expected to adhere to a predetermined trajectory $\boldsymbol{x}_t^{\text{des}}$ that is specified in both space and time. This is a commonplace task for robots in real-world applications. To later obtain a functioning controller, it is generally advantageous to calculate a corresponding input reference as well. This input sequence is computed based on the specific underlying control model of the robot. Using this feedforward solution, the robot will be able to roughly follow its trajectory while the controller merely needs to compensate for small deviations from the desired reference. The input and state reference can be computed analytically for the nominal dynamics via

$$\boldsymbol{x}_t^{\text{r}} = \boldsymbol{x}_t^{\text{des}}, \tag{18a}$$

$$\boldsymbol{u}_t^{\text{r}} = \frac{1}{T}\boldsymbol{S}_{\theta_t^{\text{des}}}^{\mathsf{T}}\big(\boldsymbol{x}_{t+1}^{\text{des}} - \boldsymbol{x}_t^{\text{des}}\big)\,. \tag{18b}$$

Here, $\theta_t^{\text{des}} := \big[\boldsymbol{x}_t^{\text{des}}\big]_3$ refers to the desired orientation of the robot, which is the third component of the desired trajectory. When computing the nominal reference solution that way, it is important to ensure that the nominal reference input adheres to the constraints (9) on the control inputs. This work considers periodic reference trajectories, which satisfy the following properties

$$\boldsymbol{x}_t^{\text{r}} = \boldsymbol{x}_{t+N}^{\text{r}}, \tag{19a}$$

$$\boldsymbol{u}_t^{\text{r}} = \boldsymbol{u}_{t+N}^{\text{r}} \quad \forall t \in \mathbb{N}_0, \tag{19b}$$

https://doi.org/10.1017/S0263574723000863 Published online by Cambridge University Press

that is, the final state and input should loop back to the initial state and input reference. However, it is not necessary for the orientation of the robot to be exactly the same from one lap to the next as long as it changes by a whole number of full rotations. It is worth noting that the methods presented in this work can also be applied to more general trajectories, as long as (19) holds for at least some sufficiently long time intervals of the overall trajectory. For instance, if the same corridor is passed with the same velocity profile at irregular intervals but the trajectories do not coincide otherwise.

5.1. Exploration-exploitation-based reference generation

In the presence of input constraints (9) and the nonlinear dynamics (14) or (16), it is often difficult to achieve perfect tracking of the desired trajectory $\boldsymbol{x}_t^{\mathrm{des}}$. This also means that the reference cannot be computed in closed form like in (18). In this subsection, a method is proposed to compute reference trajectories $\boldsymbol{u}_t^{\mathrm{r}}$, $\boldsymbol{x}_t^{\mathrm{r}}$ for input and state, respectively, automatically solving the exploration-exploitation trade-off of following a desired trajectory $\boldsymbol{x}_t^{\mathrm{des}}$ as close as possible (exploitation), while at the same time reducing the posterior variance of the underlying Gaussian process model by exploring input sequences along the desired trajectory that may have the potential to further reduce the tracking error (exploration).

Without loss of generality, $\boldsymbol{x}_0^{\mathrm{r}} := \boldsymbol{x}_0^{\mathrm{des}}$ is assumed. The notation $\boldsymbol{v}_{t+i|t}$ is used to indicate the model-based prediction at time t of the quantity $\boldsymbol{v}$ for time $t+i$. The reference trajectories are computed in a receding horizon fashion, where for each time step, the trajectory is optimized over a horizon $H_{\mathrm{ff}} < N$. This approach is computationally more efficient than optimizing the entire reference at once but possibly suboptimal. Additionally, this method scales linearly with the period of the reference N, making it viable for longer periods in the first place. The so-called performance cost is defined to minimize the deviations to the input reference

$$J_t\left(\boldsymbol{x}_{\bullet|t}^{\mathrm{r}}, \boldsymbol{u}_{\bullet|t}^{\mathrm{r}}\right) := \sum_{k=t}^{t+H_{\mathrm{ff}}-1} \left\|\boldsymbol{x}_k^{\mathrm{des}} - \boldsymbol{x}_{k|t}^{\mathrm{r}}\right\|_{\boldsymbol{Q}_{\mathrm{ff}}}^2 + \left\|\boldsymbol{u}_{k-1|t}^{\mathrm{r}} - \boldsymbol{u}_{k|t}^{\mathrm{r}}\right\|_{\boldsymbol{R}_{\mathrm{ff}}}^2 \tag{20}$$

with $\|\boldsymbol{v}\|_{\boldsymbol{V}}^2 := \boldsymbol{v}^{\mathsf{T}}\boldsymbol{V}\boldsymbol{v}$. Here, $\boldsymbol{Q}_{\mathrm{ff}}$ and $\boldsymbol{R}_{\mathrm{ff}}$ denote the symmetric positive definite weighting matrices. The input change in (20) is penalized to encourage smooth input references. For $\boldsymbol{f}_{\boldsymbol{gh}}$ the reference will be iteratively updated as more data becomes available. On its own, the performance cost J_t does not incentivize the robot to venture into areas of the input space that have not yet been visited. Because $\boldsymbol{h}$ is inferred from GPs, this can be done by considering the posterior variance $\boldsymbol{\Sigma}_{\boldsymbol{h}}$ [24]. The model $\boldsymbol{f}_{\boldsymbol{gh}}$ reliably predicts the position of the robot around previously seen input sequences, that is, if the variances of the corresponding GP predictions are low. Input sequences that have not yet been applied to the robot, however, show a large posterior variance of the GP predictions. Therefore, the learning cost function is defined as

$$L(\boldsymbol{p}_{\bullet|t}) := \sum_{k=t}^{t+H_{\mathrm{ff}}-1} \sum_{j\in\{\mathrm{x},\mathrm{y},\omega\}} \beta^{k-t} w_j \sigma_j\left(\boldsymbol{p}_{k|t}\right). \tag{21}$$

Here, w_j are weighting parameters for the standard deviations $\sigma_j\left(\boldsymbol{p}_{k|t}\right)$ which when squared are the diagonal entries of the posterior variance $\boldsymbol{\Sigma}_{\boldsymbol{h}}\left(\boldsymbol{p}_{k|t}\right)$ of the GPs approximating $\boldsymbol{h}$ from (15). The scalar parameter $0 < \beta < 1$ is called the forgetting factor [2]. This exponential discounting puts more emphasis on the learning cost towards the beginning of the prediction horizon. This can be useful for the convergence of the reference generation as nominal 'exploitative' behavior is incentivized towards the end of the prediction horizon, which would otherwise conflict with the performance cost J_t from (20). The so-called exploration-exploitation cost function [25] used for the reference generation is now defined as

$$V_t\left(\boldsymbol{x}_{\bullet|t}^{\mathrm{r}}, \boldsymbol{u}_{\bullet|t}^{\mathrm{r}}, \boldsymbol{p}_{\bullet|t}\right) := J_t\left(\boldsymbol{x}_{\bullet|t}^{\mathrm{r}}, \boldsymbol{u}_{\bullet|t}^{\mathrm{r}}\right) - \gamma L\left(\boldsymbol{p}_{\bullet|t}\right). \tag{22}$$

Note that L enters V_t negatively since the learning cost should be maximized compared to the performance cost J_t, that is, the deviation to the desired reference. Minimizing V_t allows the reference generator

https://doi.org/10.1017/S0263574723000863 Published online by Cambridge University Press

Table I. *Different modifications to the optimization problem (23) to obtain a periodic state and input reference.*

time step t	**solution $\boldsymbol{x}^{\mathrm{r}}, \boldsymbol{u}^{\mathrm{r}}$**	**additions to (23)**
$t = 0$	$\boldsymbol{x}^{\mathrm{r}}_0 := \boldsymbol{x}^{\mathrm{des}}_0$, $\boldsymbol{u}^{\mathrm{r}}_0 := \boldsymbol{u}^{\mathrm{r}}_{0\|0}$, $\boldsymbol{x}^{\mathrm{r}}_1 := \boldsymbol{x}^{\mathrm{r}}_{1\|0}$, $\boldsymbol{u}^{\mathrm{r}}_{N-1} := \boldsymbol{u}^{\mathrm{r}}_{-1\|0}$	opt. variable $\boldsymbol{u}^{\mathrm{r}}_{-1} = \boldsymbol{u}^{\mathrm{r}}_{N-1}$, constraint $\boldsymbol{x}^{\mathrm{r}}_{0\|0} = \boldsymbol{x}^{\mathrm{des}}_0$
$0 < t < N - H_{\mathrm{ff}}$	$\boldsymbol{x}^{\mathrm{r}}_{t+1} := \boldsymbol{x}^{\mathrm{r}}_{t+1\|t}$, $\boldsymbol{u}^{\mathrm{r}}_t := \boldsymbol{u}^{\mathrm{r}}_{t\|t}$	-
$N - H_{\mathrm{ff}} \leq t < N - 1$	$\boldsymbol{x}^{\mathrm{r}}_{t+1} = \boldsymbol{x}^{\mathrm{r}}_{t+1\|N-H_{\mathrm{ff}}}$, $\boldsymbol{u}^{\mathrm{r}}_t = \boldsymbol{u}^{\mathrm{r}}_{t\|N-H_{\mathrm{ff}}}$	constraints $\boldsymbol{u}^{\mathrm{r}}_{N-1\|N-H_{\mathrm{ff}}} = \boldsymbol{u}^{\mathrm{r}}_{N-1}$, $\boldsymbol{x}^{\mathrm{r}}_{N\|N-H_{\mathrm{ff}}} = \boldsymbol{x}^{\mathrm{r}}_0$

to explore new input sequences in unseen regions of the input space (exploration) while at the same time refining the solution for previously scouted input sequences (exploitation). Thereby, the parameter $\gamma \geq 0$ controls the exploration-exploitation trade-off. This cost function is less prone to getting stuck in local optima, which can possibly reduce the tracking error compared to a purely exploitative approach merely minimizing J_t. For each time step $0 < t < N - H_{\mathrm{ff}}$, the reference can be computed via

$$\min_{\boldsymbol{x}^{\mathrm{r}}_{\bullet|t},\, \boldsymbol{u}^{\mathrm{r}}_{\bullet|t}} V_t(\boldsymbol{x}^{\mathrm{r}}_{\bullet|t}, \boldsymbol{u}^{\mathrm{r}}_{\bullet|t}, \boldsymbol{p}_{\bullet|t}) \tag{23a}$$

$$\text{s.t. } \boldsymbol{u}^{\mathrm{r}}_{t-1|t} = \boldsymbol{u}^{\mathrm{r}}_{t-1}, \tag{23b}$$

$$\boldsymbol{x}^{\mathrm{r}}_{t|t} = \boldsymbol{x}^{\mathrm{r}}_t, \tag{23c}$$

$$\boldsymbol{x}^{\mathrm{r}}_{k+1|t} = \boldsymbol{f}_{g/\boldsymbol{gh}}(\boldsymbol{x}^{\mathrm{r}}_{k|t}, \boldsymbol{u}^{\mathrm{r}}_{k|t}, \bullet), \tag{23d}$$

$$\boldsymbol{a}^{\mathrm{r}}_{k|t} = \left[\boldsymbol{u}^{\mathrm{r}}_{k|t}{}^{\mathsf{T}} \boldsymbol{u}^{\mathrm{r}}_{k-1|t}{}^{\mathsf{T}}\right]^{\mathsf{T}} \in k_\varepsilon \mathcal{A}, , \tag{23e}$$

$$\boldsymbol{p}_{k|t} = \left[\boldsymbol{u}^{\mathrm{r}}_{k|t}{}^{\mathsf{T}} k\right]^{\mathsf{T}}, \tag{23f}$$

$$|\boldsymbol{u}^{\mathrm{r}}_{k|t} - \boldsymbol{u}^{\mathrm{b}}_k| \leq \boldsymbol{\Delta}_{\mathrm{br}}, \tag{23g}$$

$$k \in \{t, \ldots, t + H_{\mathrm{ff}} - 1\}.$$

The expression $\boldsymbol{f}_{g/\boldsymbol{gh}}(\boldsymbol{x}^{\mathrm{r}}_{k|t}, \boldsymbol{u}^{\mathrm{r}}_{k|t}, \bullet)$ depends on the application and can refer to either (14) or (16), where the state and input are replaced with their respective reference values. A constant $0 < k_\varepsilon < 1$ is introduced to ensure that the controller can still compensate for any mismatches that may occur during closed-loop operation, as $\mathbf{0}$ is in the interior of $\mathcal{A}$. In (23e). The notation $\varepsilon\mathcal{A} := \left\{\varepsilon \boldsymbol{a} \in \mathbb{R}^6 \mid \boldsymbol{a} \in \mathcal{A}\right\}$ for some $\varepsilon \in \mathbb{R}$ is used to denote a scaled set. The constraint (23g) limits the deviation between the input reference and a baseline input sequence $\boldsymbol{u}^{\mathrm{b}}_t$ to $\boldsymbol{\Delta}_{\mathrm{br}} = \left[0.05\ \mathrm{m/s}\ 0.05\ \mathrm{m/s}\ 0.15\ \mathrm{rad/s}\right]^{\mathsf{T}}$. Here $\boldsymbol{u}^{\mathrm{b}}_t$ is set to the input sequence computed for the nominal system (18), which restricts the solutions to inputs near a reasonable guess. This also limits the amplitude of possible oscillations that can occur due to the exploratory nature of the cost function (22).

The optimization problem (23) is solved for time steps $0 < t < N - H_{\mathrm{ff}}$, and the state and input reference are set to $\boldsymbol{x}^{\mathrm{r}}_{t+1} := \boldsymbol{x}^{\mathrm{r}}_{t+1|t}$ and $\boldsymbol{u}^{\mathrm{r}}_t := \boldsymbol{u}^{\mathrm{r}}_{t|t}$. However, for the initial time step $t = 0$ and the final time steps, the optimization problem (23) needs to be modified as shown in Table I. At $t = 0$, the input $\boldsymbol{u}^{\mathrm{r}}_{-1} = \boldsymbol{u}^{\mathrm{r}}_{N-1}$ is added as an additional optimization variable. For $t = N - H_{\mathrm{ff}}$, the optimization problem is modified with the additional constraints $\boldsymbol{u}^{\mathrm{r}}_{N-1|N-H_{\mathrm{ff}}} = \boldsymbol{u}^{\mathrm{r}}_{N-1}$ and $\boldsymbol{x}^{\mathrm{r}}_{N|N-H_{\mathrm{ff}}} = \boldsymbol{x}^{\mathrm{r}}_0$ to ensure that the periodicity constraint (19) is satisfied. The reference states and inputs for time steps $N - H_{\mathrm{ff}} \leq t < N - 1$ are set to the corresponding open-loop solution obtained from time step $N - H_{\mathrm{ff}}$, that is, $\boldsymbol{x}^{\mathrm{r}}_{t+1} = \boldsymbol{x}^{\mathrm{r}}_{t+1|N-H_{\mathrm{ff}}}$ and $\boldsymbol{u}^{\mathrm{r}}_t = \boldsymbol{u}^{\mathrm{r}}_{t|N-H_{\mathrm{ff}}}$, respectively. Alternatively, instead of setting the tail end of the reference to the open-loop solution of time $N - H_{\mathrm{ff}}$, the prediction horizon H_{ff} could be decreased as time progresses. The constants are assigned as

https://doi.org/10.1017/S0263574723000863 Published online by Cambridge University Press

$\boldsymbol{Q}_{\mathrm{ff}} = \mathrm{diag}(1000, 1000, 50)$, $\boldsymbol{R}_{\mathrm{ff}} = \mathrm{diag}(1, 1, 5)$, $H_{\mathrm{ff}} = 5$, and $k_e = 0.95$. The optimization problem (23) is set up in Matlab using CasADi [26] and is solved using the interior-point optimizer IPOPT [27].

Reducing the effective number of data points

The number of training points $N_{\boldsymbol{h}}$ is constantly increasing due to the iterative nature of this procedure. Since, as detailed in Section 3, additional training points increase the time it takes for a prediction of the posterior mean and variance, we aim to mitigate this subsequently.

One suitable approach to reduce the effective number of data points for the GPs is called local approximate Gaussian process models [16]. Since, for each input at a given time step t, the data points that were recorded for a time step far into the future or in the past in previous iterations will likely have little effect on the approximation. Therefore, only the points that have the most influence on the Gaussian process approximation in the current time interval could be selected from $\mathcal{D}_{\boldsymbol{h}}$.

Several approaches were proposed in the literature [3, 15, 28] on how to select a subset of points from $\mathcal{D}_{\boldsymbol{h}}$ for local GP approximations, ranging from Greedy optimization [29] to simple nearest neighbor approaches [16]. Note, however, that the scheme to select these points should be reasonably quick, as this would otherwise conflict with the overall goal of reducing the computational time. Generally, it has been observed that a simple (and fast) nearest neighbor approach works well [16] and will therefore be used in the following. Contrary to what was proposed in refs. [16, 29], however, instead of actually reducing the training points to a subset of $\mathcal{D}_{\boldsymbol{h}}$, sparse GPs are trained on the full data set and pseudo-inputs are placed at the specific locations [28, 30] which results in the same computational effort for the evaluation of the posterior mean and variance. Since an input is comprised of translational speeds, a rotational velocity, and a time step, the kernel $k_j, j \in \{\mathrm{x}, \mathrm{y}, \omega\}$, is used as a similarity/distance measure [30]. This procedure has to be done prior to solving optimization problem (23) for each time step.

Since for every iteration N^{lap} laps are driven, for each time step, several data points with slightly different model mismatches are obtained. However, the corresponding inputs of each iteration repeat N^{lap} times due to the assumption that $\boldsymbol{h}$ only depends on the reference, which is fixed for each iteration. Since adding the same pseudo-input twice will not improve the approximation further [31], the possible locations of the pseudo-inputs are restricted to the unique set of inputs in $\mathcal{D}_{\boldsymbol{h}}$. This also means that for the choice of $\boldsymbol{p}_t$, driving several laps with the same reference input sequence will not further increase the effective number of inputs.

To ensure a reliable approximation, one should at least include all the data points that were recorded for the time steps $\{t, \ldots, t + H_{\mathrm{ff}} - 1\}$ of the current time interval. Any additional points then improve the approximation accuracy near the boundary of the interval. This results in $N^{\mathrm{loc}} \leq N$ effective data points. Especially, the setup of the optimization problem is significantly sped up that way. This approximation makes the optimization-based scheme also viable for very long trajectories, as the effective number of training points is independent of N.

The procedure combining the two aforementioned approaches for generating a local GP from the full GP trained on the full data set $\mathcal{D}_{\boldsymbol{h}}$ is summarized in Algorithm 1. Using this procedure reduces the effective number of inputs from $N_{\boldsymbol{h}} = NN^{\mathrm{lap}}N^{\mathrm{iter}}$ (if the inputs are selected naïvely) to N^{loc}. As $N_{\boldsymbol{h}}$ is increasing with each iteration, the number of pseudo-inputs $N^{\mathrm{loc}} \propto N^{\mathrm{iter}}$ is increased accordingly to retain a good approximation with a growing number of data points. Here, N^{iter} denotes the number of iterations that are included in $\mathcal{D}_{\boldsymbol{h}}$. The more laps are driven for each iteration, and the longer the trajectories are, the more effective these approximations are.

5.2. Model predictive trajectory tracking controller

To follow the computed reference trajectory, a MPC [32] is employed, because it can consider the input constraints (9) as well as leverage the nonlinear data-based model of our robot.

https://doi.org/10.1017/S0263574723000863 Published online by Cambridge University Press

Algorithm 1 Proposed local approximation $\mathcal{GP}_{\text{loc}}$ of a Gaussian process $\mathcal{GP}_{\text{full}}$ trained on a data set $\mathcal{D}_{\boldsymbol{h}}$.

input: t, H^{ff}, $\mathcal{D}_{\boldsymbol{h}}$, N^{loc}, $\mathcal{GP}_{\text{full}}$
$\mathcal{T} = \{t, \dots, t + H^{\text{ff}} - 1\}$
$\tilde{\mathcal{X}}$ = unique set of inputs of $\mathcal{D}_{\boldsymbol{h}}$
$\mathcal{X}$ = all inputs $\left[\boldsymbol{u}_k^{\text{r}\,\mathsf{T}} \;\; k\right]^{\mathsf{T}}$ from $\tilde{\mathcal{X}}$ such that $k \in \mathcal{T}$
for each point $\tilde{\boldsymbol{x}}_i$ in $\tilde{\mathcal{X}}$ **do**
  for each point $\boldsymbol{x}_j$ in $\mathcal{X}$ **do**
    $[\boldsymbol{q}]_j = k(\tilde{\boldsymbol{x}}_i, \boldsymbol{x}_j)$ ▷ evaluate the kernel
  $q_i^{\max} = \max(\boldsymbol{q})$ ▷ get max. value for each $\tilde{\boldsymbol{x}}_i$
$\mathcal{X}^{\text{loc}}$ = set of N^{loc} points $\tilde{\boldsymbol{x}}_i$ with the biggest corresponding value $q_i^{\max}$
train sparse GP $\mathcal{GP}_{\text{loc}}$ on data set $\mathcal{D}_{\boldsymbol{h}}$ with hyperparameters from $\mathcal{GP}_{\text{full}}$ and pseudo-inputs located at $\mathcal{X}^{\text{loc}}$
return $\mathcal{GP}_{\text{loc}}$ ▷ local approximation of $\mathcal{GP}_{\text{full}}$

Starting from an initial state measurement $\boldsymbol{x}_t$ at time step t, the MPC predicts the future poses for the following H time steps using either the nominal or data-based models. The input sequence is thereby chosen such that the weighted squared error between the states and inputs to their corresponding reference values computed in Subsection 5.1 is minimized. State and input constraints can easily be integrated into the optimization problem. The optimal control problem is then solved in real time to find the optimal control action for each time step. Using the more accurate data-based control model within the optimal control problem instead of the nominal one will improve the prediction accuracy and, subsequently, the control performance.

The optimal control problem for trajectory tracking can be formulated as

$$\min_{\boldsymbol{x}_{\bullet|t}, \boldsymbol{u}_{\bullet|t}} \sum_{k=t}^{t+H-1} \left\| \boldsymbol{x}_k^{\text{r}} - \boldsymbol{x}_{k|t} \right\|_{\boldsymbol{Q}}^2 + \left\| \boldsymbol{u}_k^{\text{r}} - \boldsymbol{u}_{k|t} \right\|_{\boldsymbol{R}}^2 \tag{24a}$$

$$\text{s. t.} \quad \boldsymbol{u}_{t-1|t} = \boldsymbol{u}_{t-1}, \tag{24b}$$

$$\boldsymbol{x}_{t|t} = \boldsymbol{x}_t, \tag{24c}$$

$$\boldsymbol{x}_{k+1|t} = \boldsymbol{f}_{\text{n}/\boldsymbol{g}/\boldsymbol{gh}}\left(\boldsymbol{x}_{k|t}, \boldsymbol{u}_{k|t}, \bullet\right), \tag{24d}$$

$$\boldsymbol{a}_{k|t} = \left[\boldsymbol{u}_{k|t}^{\mathsf{T}} \boldsymbol{u}_{k-1|t}^{\mathsf{T}}\right]^{\mathsf{T}} \in \mathcal{A},$$
$$k \in \{t, \dots, t + H - 1\}, \tag{24e}$$

$$\boldsymbol{x}_{t+H|t} = \boldsymbol{x}_{t+H}^{\text{r}}, \tag{24f}$$

$$\left\| \boldsymbol{D}(\boldsymbol{u}_{t+H}^{\text{r}} - \boldsymbol{u}_{t+H-1|t}) \right\|_{\infty} \leq \Delta\Omega_{\max} \tag{24g}$$

with positive definite weighting matrices $\boldsymbol{Q} = \boldsymbol{I}$ and $\boldsymbol{R} = \frac{1}{30}\boldsymbol{I}$, the current state measurement $\boldsymbol{x}_t$ and the prediction horizon $H = 20$. This corresponds to a prediction of the robot pose $HT = 4$ s into the future. The constraint (24f) is an equality constraint for the terminal state of the prediction, that is, the state prediction should coincide with the reference at the end of the horizon. Together with the constraint (24g), this makes sure that applying the nominal reference trajectory is feasible for all future time steps, which is a straightforward way of guaranteeing convergence to the reference for the undisturbed system [33, 34]. After (24) is solved, the current input is set to the solution $\boldsymbol{u}_t := \boldsymbol{u}_{t|t}$ for each time step. The online MPC problem (24) is implemented using the ACADO Toolkit [35], which implements a real-time iteration scheme with Gauss-Newton Hessian approximation [36]. The thereby arising quadratic programs are solved using qpOASES [37]. The average solving time for the OCP is around 200 and 700 μs using the nominal and the data-based models, respectively. All computations in this study are done using a laptop computer with an Intel Core i7-9750H CPU. Note that, as in (24) the trajectory-specific mismatch $\boldsymbol{h}$ is neither explicitly state nor input dependent, for fixed $\boldsymbol{u}_t^{\text{r}}$ and t the term $\boldsymbol{h}(\boldsymbol{p}_t)$ is constant. Therefore, the computational effort of solving the OCP for $\boldsymbol{f}_{\boldsymbol{g}}$ and $\boldsymbol{f}_{\boldsymbol{gh}}$ is similar.

One way to improve the tracking performance using the nominal model $\boldsymbol{f}_{\text{n}}$ would be to select the weighting matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ of the MPC in (24a) to obtain a stiffer control, however, in turn making

https://doi.org/10.1017/S0263574723000863 Published online by Cambridge University Press

Table II. *The kernel hyperparameters for the three different GPs approximating the function* $\boldsymbol{h}$.

GP	$\sigma_{\mathrm{s},j}$	$\sigma_{\mathrm{n},j}$	$\boldsymbol{A}_j$	$\ell_{\mathrm{PER},j}$
x	0.020	0.005	$0.025\boldsymbol{I}$	0.080
y	0.020	0.005	$0.025\boldsymbol{I}$	0.080
ω	0.120	0.030	$0.025\boldsymbol{I}$	0.080

the controller more sensitive to measurement noise. A stiffer control can only reduce systematic model mismatches but never eliminate them entirely. In general, high-gain position feedback should be avoided for robotic applications to be more compliant, especially around human personnel [38]. Contrary to the nominal model, as systematic mismatches are accounted for using the data-based model, a high tracking accuracy can be achieved without requiring a stiff controller.

Note that, when a differential-drive or car-like mobile robot is used instead of an omnidirectional mobile robot, some modifications to the cost function (24a) might be necessary to retain some theoretical properties, see [6] for details.

6. Experimental results

For the functions h_j, the corresponding kernels $k_j, j \in \{\mathrm{x}, \mathrm{y}, \omega\}$

$$k_j\left(\begin{bmatrix}\boldsymbol{x}\\ t\end{bmatrix}, \begin{bmatrix}\boldsymbol{x}'\\ t'\end{bmatrix}\right) = \sigma_{\mathrm{s},j}^2 \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{x}')^{\mathsf{T}}\boldsymbol{A}_j(\boldsymbol{x}-\boldsymbol{x}')\right) \exp\left(-\frac{2\sin^2\left(\frac{\pi}{N}(t-t')\right)}{\ell_{\mathrm{PER},j}^2}\right) \tag{25}$$

are used, where $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^3$ and $t, t' \in \mathbb{N}_0$. This choice of the kernel is a product of the SE kernel (3) and the periodic kernel (4), which respects the periodicity of the considered trajectories, see (19). The kernel hyperparameters $\left(\sigma_{\mathrm{s},j}, \sigma_{\mathrm{n},j}, \boldsymbol{A}_j, \ell_{\mathrm{PER},j}\right)$ are set to the values from Table II. The selection of the hyperparameters requires experience with the system, the environment, and the measurement setup. However, once a (nominal) model of the robot is available, a large portion of the relevant parameters can be tuned in simulation before moving to real hardware, including the parameters of the reference generator and the MPC. Optimizing the hyperparameters can be a good first step in building some intuition for reasonable values of the kernel hyperparameters in case no prior experience with the hardware setup is available.

The parameters for the learning cost function are set to $\beta = 0.8$, and the individual weights for the posterior variances are set to $w_j = \frac{\sigma_{\mathrm{s},\omega}}{\sigma_{\mathrm{s},j}}, j \in \{\mathrm{x}, \mathrm{y}, \omega\}$. This choice of w_j is made as an effort to lower the influence of GPs with high signal variances. A value of $\gamma = 0.01$ is used unless stated otherwise.

To compare the experimental results for different cost functions (22), the cumulative learning cost is defined as

$$L_\Sigma(\boldsymbol{p}_t) := \sum_{t=0}^{N-1} \sum_{j\in\{\mathrm{x},\mathrm{y},\omega\}} w_j \sigma_j(\boldsymbol{p}_t) \geq L_\Sigma^{\mathrm{lb}} \tag{26}$$

as an indicator of the uncertainty of the current reference solution. Note that since σ_j^2 is lower bounded by the signal noise variance $\sigma_{\mathrm{n},j}^2$ the term L_Σ also has a lower bound of $L_\Sigma^{\mathrm{lb}} = N \sum_{j\in\{\mathrm{x},\mathrm{y},\omega\}} w_j \sigma_{\mathrm{n},j}$ which is, however, only reached in the limiting case $N_{\boldsymbol{h}} \to \infty$, see ref. [39] for details. Since the approximation from Subsection 5.1 is used, each $\sigma_j(\boldsymbol{p}_t)$ is evaluated using the local GP approximation corresponding to time step t. The interval for the approximation at time step t (see Algorithm 1) includes $N^{\mathrm{loc}} = N^{\mathrm{iter}}(H^{\mathrm{ff}} + 24)$ points, that is, all the points corresponding to the time steps from the current prediction interval $\{t, \ldots, t + H_{\mathrm{ff}} - 1\}$ as well as $24N^{\mathrm{iter}}$ additional points to improve the prediction quality at the boundary. With this setup, the performance of the proposed method is intently analyzed for different trajectories.

https://doi.org/10.1017/S0263574723000863 Published online by Cambridge University Press

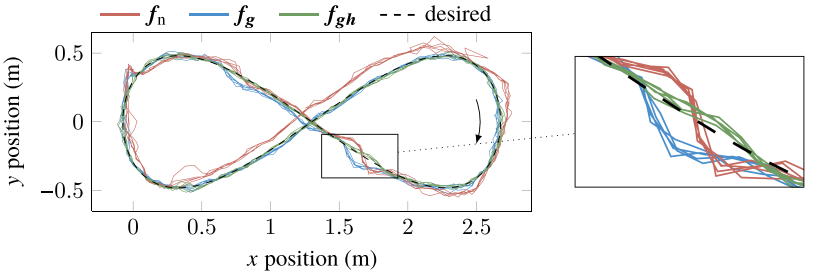


Figure 5. *For the hardware experiments, the robot drives five consecutive laps for each model on the ∞-shaped trajectory. The errors to the desired trajectory $\boldsymbol{x}_t^{des}$ (dashed line) are shown using a magnification of 10 for better visibility. The orientation of the robot is not depicted.*

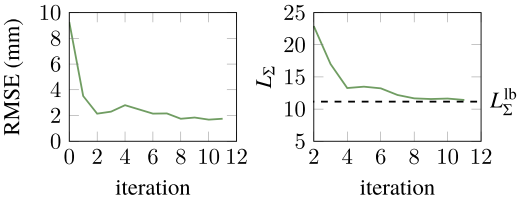


Figure 6. *The RMSE for the position of the robot as well as the term L_Σ for the ∞-shaped trajectory. The initial (0th) is the nominal solution, and the first iteration is the solution corresponding to no additional trajectory-specific information, that is, $\boldsymbol{f}_g$.*

Infinity-shaped trajectory

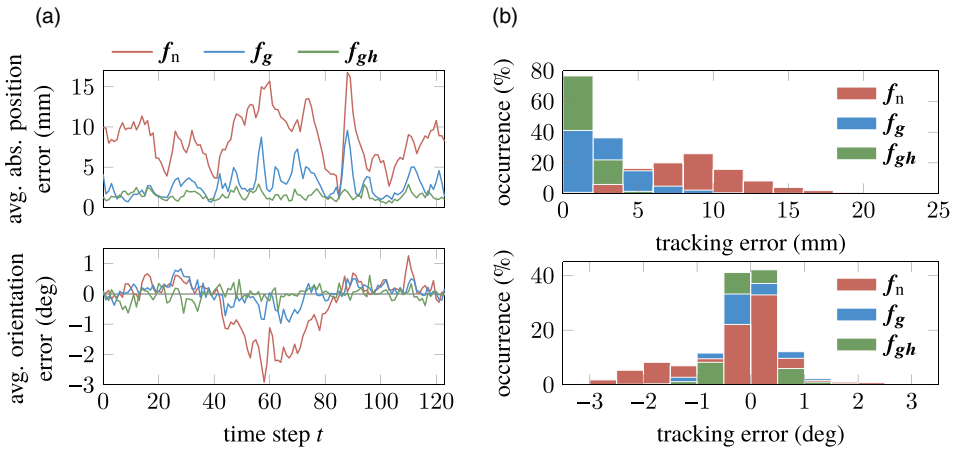
In the first scenario, the custom-built robot drives five consecutive laps of a ∞-shaped trajectory with a length of $N = 124$ time steps and a desired orientation of $\theta_t^{\text{des}} \equiv 45°$, compare Fig. 5. Here, the error to the desired trajectory is depicted using a magnification of ten. Clearly, the tracking error shows systematic behavior, which is especially noticeable using the nominal model. Contrary to any random, non-systematic mismatches, this behavior can be compensated when using a more accurate, data-based model.

In Fig. 6, the root mean square error (RMSE) of the position is depicted over the iterations. The initial (0th) iteration is the one corresponding to the nominal solution. The five laps that are driven using $\boldsymbol{f}_g$ are considered the first iteration and are used to train the trajectory-specific mismatch for the second iteration. Evidently, the error is not decreasing monotonically. This is, on the one hand, due to the exploitative behavior driving the robot away from the desired trajectory and, on the other hand, due to a wrong and uncertain model $\boldsymbol{f}_{gh}$, reflected by higher values of L_Σ. Over the iterations, more knowledge is gathered, and the algorithm becomes more confident in its solution, compare Fig. 6 (right). This exploration-exploitation trade-off is done fully automatically thanks to the cost function (22).

In the tracking error histogram in Fig. 7b, the control error for the nominal model $\boldsymbol{f}_n$ persists due to unmodeled mismatches, resulting in a RMSE of 9.27 mm or 1.00°, in terms of position and orientation, respectively. The general GP-based model $\boldsymbol{f}_g$ significantly reduces the RMSE to 3.50 mm or 0.53°. However, the model specifically tailored to the trajectory, $\boldsymbol{f}_{gh}$, manages to further reduce the tracking RMSE to 1.68 mm and 0.39° in the best (10th) iteration. This is shown in the histogram in Fig. 7b. The error in position is consistently reduced for most time steps, as seen in Fig. 7a (left). The improvement in orientation error between $\boldsymbol{f}_g$ and $\boldsymbol{f}_{gh}$ is only marginal.

An explanation might be that the reference-specific term may primarily be explained by effects due to local ground changes that might not have as big of an impact on the orientation as on the position. These effects are averaged out for $\boldsymbol{f}_g$ due to the way the data $\mathcal{D}_g$ was collected.

https://doi.org/10.1017/S0263574723000863 Published online by Cambridge University Press



Average errors over time. *Histogram of tracking errors.*

Figure 7. *Error for the deviations from the desired ∞-shaped trajectory $\boldsymbol{x}_t^{des} - \boldsymbol{x}_t$ on position level (left) and orientation level (right) using the three different robot models.*

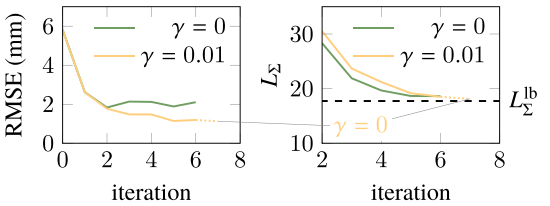


Figure 8. *The RMSE for the position of the robot as well as the term L_Σ with $\gamma = 0$ and $\gamma = 0.01$ for the rectangular trajectory. The initial (0th) is the nominal solution, and the first iteration is the solution corresponding to no additional trajectory-specific information, that is, $\boldsymbol{f}_g$. The seventh and final iteration for the exploration-based approach $\gamma = 0$ (pure exploitation) was used.*

In the magnification in Fig. 5, it also becomes evident that even the trajectory-specific model can struggle to perfectly track the desired trajectory. One explanation could be that it is impossible to perfectly track the desired $\boldsymbol{x}_t^{\text{des}}$ without some local modifications, for example, due to the saturation of the input limits (9). Another explanation would be that the algorithm could not find a potential input sequence with better tracking performance within the limits of its possibilities, for example, due to the selection of the parameters γ or $\boldsymbol{\Delta}_{\text{br}}$.

Rectangular trajectory and the influence of explorative behavior

In the second scenario, the robot shall follow a rectangular trajectory. To that end, the robot drives trapezoidal velocity profiles on each rectangle edge with a top speed of about 0.45 ms, $N = 197$ and $\theta_t^{\text{des}} \equiv 0°$. At each corner, the robot halts for a time period of 1 s. In Fig. 8, five consecutive laps for four different input sequences are depicted. The nominal model struggles to accurately predict the robot's behavior which is noticeable in the overshoot in the braking phase at each corner of the rectangle. The data-based model without any trajectory-specific information is able to reduce the RMSE by more than 50% (from 5.8 to 2.6 mm). Both of the reference solutions for $\boldsymbol{f}_{\text{n}}$ and $\boldsymbol{f}_{gh}$ were generated with $\gamma = 0$. When including trajectory-specific information incorporating a learning term by choosing $\gamma = 0.01$ outperforms the pure 'exploitative' approach of $\gamma = 0$. The uncertainty decreases for both values of γ (Fig. 8, right). This can indicate that for $\gamma = 0$, the algorithm is stuck in a local optimum quickly decreasing L_Σ by

https://doi.org/10.1017/S0263574723000863 Published online by Cambridge University Press

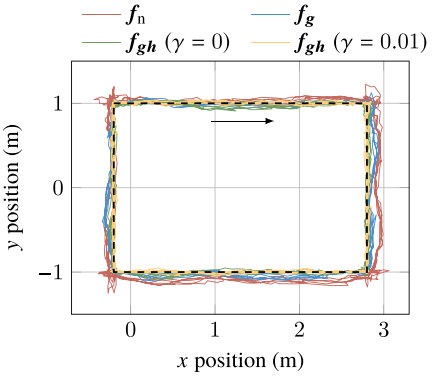


Figure 9. *Final tracking results for the rectangle trajectory using different robot models. The errors to the desired trajectory $\boldsymbol{x}_t^{des}$ (dashed line) are shown using a magnification of 25 for better visibility. For $\boldsymbol{f}_{gh}$, only the iterations with the lowest position error are depicted.*

revisiting previous input sequences with low variances. For $\gamma = 0.01$, the first iteration is about as good as with $\gamma = 0$ but, in the following iterations, the error, as well as the uncertainty, decrease slowly but steadily until iteration 6 where the uncertainty is about as low as for $\gamma = 0$ but the error in the position is about 45% lower (1.9 vs. 1.1 mm). Note that the RMSE also includes any non-systematic noise, which means achieving an RMSE of zero is essentially impossible in any real-world experiment. For the final seventh iteration, the exploration-based cost function, $\gamma = 0$ was set as enough information was gathered. Otherwise, at the beginning of this scheme, the choice of $\gamma \neq 0$ can hurt the tracking performance as it can increase the tracking error in exchange for exploratory behavior. Therefore, the tracking performance can generally be expected to be worse compared to taking $\gamma = 0$ in the first few iterations, especially for very large values of γ. The error for the orientation is about 0.2° regardless of the value of γ and the iteration and is therefore not depicted in Fig. 8. While this difference in tracking performance between the two different cost functions seems small, when looking at the tracking error in more detail in Fig. 9, there is some systematic error left for $\gamma = 0$ in the top straight especially whereas for $\gamma = 0.01$, there is hardly any visible systematic error even with the employed error magnification of 25.

The time it takes to pre-compute the references for $\boldsymbol{f}_g$ is about 50 ms per time step, see Fig. 10 (iteration 1). Due to the increasing number of data points, the computational times for $\boldsymbol{f}_{gh}$ increase with each iteration. While the computational time per time step increases approximately quadratically, for $\gamma = 0.01$ (or $\gamma \neq 0$ in general) it increases faster than taking $\gamma = 0$. This is due to the optimization problem (23) being harder to solve when exploitation is included in the cost function (22). Moreover, as detailed in Section 3, the time to evaluate the GPs posterior variances also increases quadratically with the number of pseudo-inputs, which in this case scales linearly with the number of iterations. The computational time could be reduced by lowering the number of effective data points N^{loc} in exchange for a worse approximation accuracy of the GPs, which is not done here in order to introduce as little of an error as possible via the local approximations. The nominal model $\boldsymbol{f}_n$ does not require any time-consuming pre-computation, as the reference can be computed in closed form (18).

7. Discussion

This procedure will not replace an experienced control engineer, as the proper selection and identification of the relevant inputs $\boldsymbol{a}_k$ and $\boldsymbol{p}_k$ and the selection of individual parameters in (23) or (24) requires a thorough understanding of the hardware setup and more importantly whether there are sources of systematic errors present. However, this procedure is a semi-automated way to leverage the full potential of the hardware using real-world input–output data, without having to make any changes to the hardware

https://doi.org/10.1017/S0263574723000863 Published online by Cambridge University Press

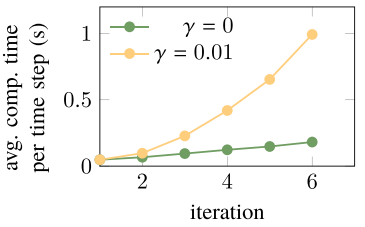


Figure 10. *Average computational time for solving (23) for each iteration using different values of γ. The times were recorded for the rectangular trajectory ($N = 197$). The first iteration is the solution corresponding to no additional trajectory-specific information, that is, $\boldsymbol{f}_g$.*

itself. The framework is also extremely flexible in the sense that prior knowledge can be included on different levels. If the learning factor is set to zero ($\gamma = 0$) much of the experience and prior knowledge needed to select the GP hyperparameters is not necessary, although this might hurt the effectiveness of this procedure. Other simplifications include the optimization of the hyperparameters before each iteration or assuming a pure time dependence of $\boldsymbol{h}$ (for $\gamma = 0$). The right choice of the learning factor γ is also important, as a learning factor close to zero can result in hardly any exploration and a very large learning factor can result in a very slow decrease of the uncertainty and the tracking error. Ideally, γ is chosen large enough such that the algorithm does not get stuck in local minima, but small such that the corresponding solutions stay somewhat close to a reasonable input sequence, like the one obtained with $\boldsymbol{f}_g$ for example.

In order for the method to be viable in practice, the computational times have to be kept small. Several modifications to the presented framework are possible that trade control performance for reduced computational times. For the reference generation, special data management techniques [23, 30] can be used to eliminate irrelevant or old data points over time to keep N_h below a certain value. In order to reduce the computational times for the OCP (24) solved in real time, one could fix the inputs $\boldsymbol{a}_{k|t}$ of $\boldsymbol{g}$ to their corresponding reference values $\boldsymbol{a}_t^{\mathrm{r}}$ resulting in a constant expression for fixed t. In fact, for less intricate systematic mismatches or in case no prior input–output data is available, the function $\boldsymbol{g}$ can be dropped from the procedure altogether.

8. Conclusion and outlook

By combining a robot-specific data-based model and a trajectory-specific data-based correction term, this paper achieves high-accuracy tracking of periodic references. Gaussian process regression was used to model the trajectory-specific term, which was assumed to explicitly depend on time and the reference. Leveraging this data-based model in a model predictive control framework results in a significant reduction of the computational time. Additionally, a method based on iterative optimization was introduced to efficiently calculate periodic input and state references while considering the dynamics of the model. It was shown that exploitative behavior, that is, trying to reduce the posterior variances of the Gaussian process model, can outperform the purely exploitative approach, merely minimizing the tracking error, which may get stuck in local optima. Since this procedure shows impressive control performance within only a few iterations, it can be used for fast online adaptations of the input reference in case of external disturbances, like changes in the environment or the configuration of the robot, without manual re-calibration. In future work, the proposed theory could be extended to high-accuracy path following control by incorporating a path parameter into the disturbance model, or to the distributed case, such as for decentralized high-accuracy formation control [40].

Author contribution. Hannes Eschmann and Henrik Ebel were responsible for the basic concepts. Hannes Eschmann carried out the experiments, gathered the data, and performed analyses of the data. Hannes Eschmann and Henrik Ebel wrote the article. Hannes Eschmann, Henrik Ebel, and Peter Eberhard discussed the results. Henrik Ebel and Peter Eberhard did proofreading.

https://doi.org/10.1017/S0263574723000863 Published online by Cambridge University Press

Financial support. This work was supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Grants 501890093 (SPP 2353) and 433183605, and through Germany's Excellence Strategy (Project PN 4-4 "Learning from Data - Predictive Control in Adaptive Multi-Agent Scenarios") under Grant EXC 2075-390740016. We acknowledge the support by the Stuttgart Center for Simulation Science (SimTech).

Competing interests. The authors declare no conflicts of interest exist.

Ethical approval. Not applicable.

References

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd edition (The MIT Press, Cambridge, MA, 2015). doi: 10.1109/TNN.1998.712192.
[2] V. Gaitsgory, L. Grüne, M. Höger, C. M. Kellett and S. R. Weller, "Stabilization of strictly dissipative discrete time systems with discounted optimal control," *Automatica* **93**, 311–320 (2018). doi: 10.1016/j.automatica.2018.03.076.
[3] L. Hewing, A. Liniger and M. N. Zeilinger, "Cautious NMPC with Gaussian Process Dynamics for Autonomous Miniature Race Cars," *Proceedings of European Control Conference (ECC)*, Limassol, Cyprus (2018) pp. 1341–1348. doi: 10.23919/ECC.2018.8550162.
[4] K.-L. Han and J. S. Lee, "Iterative path tracking of an omni-directional mobile robot," *Adv. Robot.* **25**(13–14), 1817–1838 (2011). doi: 10.1163/016918611X584703.
[5] H. Eschmann, H. Ebel and P. Eberhard, "High Accuracy Data-Based Trajectory Tracking of a Mobile Robot," *Proceedings of Advances in Service and Industrial Robotics*, Klagenfurt, Austria (2022), vol. 31, pp. 420–427. doi: 10.1007/978-3-031-04870-8_49.
[6] M. Rosenfelder, H. Ebel, J. Krauspenhaar and P. Eberhard, "Model predictive control of non-holonomic systems: Beyond differential-drive vehicles," *Automatica* **152**, 110972 (2023). doi: 10.1016/j.automatica.2023.110972.
[7] K. Kanjanawanishkul and A. Zell, "Path Following for an Omnidirectional Mobile Robot Based on Model Predictive Control," *International Conference on Robotics and Automation*, Kobe, Japan (2009) pp. 3341–3346. doi: 10.1109/ROBOT.2009.5152217.
[8] A. S. Huang, E. Olson and D. C. Moore, "LCM: Lightweight Communications and Marshalling," *International Conference on Intelligent Robots and Systems*, Taipei, Taiwan (2010) pp. 4057–4062. doi: 10.1109/IROS.2010.5649358.
[9] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)* (The MIT Press, Cambridge, England, 2005). doi: 10.7551/mitpress/3206.001.0001.
[10] S. L. Brunton, J. L. Proctor and J. N. Kutz, "Sparse identification of nonlinear dynamics with control (SINDYc)," *IFAC-PapersOnLine* **49**(18), 710–715 (2016). doi: 10.1016/j.ifacol.2016.10.249.
[11] T. Hastie, R. Tibshirani and J. Friedman, *The Elements of Statistical Learning*, 2nd edition (Springer, New York City, New York, 2017). doi: 10.1007/978-0-387-84858-7.
[12] D. J. C. MacKay, "Introduction to Gaussian processes," *Neural Netw. Mach. Learn.* **168**, 133–165 (1998).
[13] J. Quiñonero Candela and C. E. Rasmussen, "A unifying view of sparse approximate Gaussian process regression," *J. Mach. Learn. Res.* **6**(65), 1939–1959 (2005).
[14] E. Snelson and Z. Ghahramani, "Sparse Gaussian Processes Using Pseudo-Inputs," *Proceedings of the 18th International Conference on Neural Information Processing Systems*, Vancouver, Canada (2005) pp. 1257–1264.
[15] E. Snelson and Z. Ghahramani, "Local and Global Sparse Gaussian Process Approximations," *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics*, San Juan, Puerto Rico (2007), vol. 2, pp. 524–531.
[16] X. Emery, "The kriging update equations and their application to the selection of neighboring data," *Comput. Geosci.* **13**(3), 269–280 (2009). doi: 10.1007/s10596-008-9116-8.
[17] M. Titsias, "Variational Learning of Inducing Variables in Sparse Gaussian Processes," *Proceedings of Machine Learning Research*, Clearwater Beach, FL (2009), vol. 5, pp. 567–574.
[18] H. Eschmann, H. Ebel and P. Eberhard, "Data-Based Model of an Omnidirectional Mobile Robot Using Gaussian Processes," *Proceedings of IFAC Symposium on System Identification (SYSID)*, Padova, Italy (2021) pp. 13–18. doi: 10.1016/j.ifacol.2021.08.327.
[19] H. Eschmann, H. Ebel and P. Eberhard, "Trajectory tracking of an omnidirectional mobile robot using Gaussian process regression," *Automatisierungstechnik* **69**(8), 656–666 (2021). doi: 10.1515/auto-2021-0019.
[20] D. P. Kroese, T. Taimre and Z. I. Botev, *Handbook of Monte Carlo Methods* (John Wiley & Sons, Hoboken, NJ, 2013).
[21] C. E. Rasmussen and H. Nickisch, "Gaussian processes for machine learning (GPML) toolbox," *J. Mach. Learn. Res.* **11**(100), 3011–3015 (2010). doi: 10.5555/1756006.1953029.
[22] P. Manzl, M. Sereinig and J. Gerstmayr, "Modelling and Experimental Validation for Mecanumwheels," *Proceedings of International Federation for the Promotion of Mechanism and Machine Science D-A-CH Conference*, online conference (2022), vol. 8. doi: 10.17185/duepublico/75419.
[23] C. D. McKinnon and A. P. Schoellig, "Experience-Based Model Selection to Enable Long-Term, Safe Control for Repetitive Tasks Under Changing Conditions," *Proceedings of International Conference on Intelligent Robots and Systems*, Madrid, Spain (2018), vol. 30, pp. 2977–2984. doi: 10.1109/IROS.2018.8593882.

[24] T. Koller, F. Berkenkamp, M. Turchetta and A. Krause, "Learning-Based Model Predictive Control for Safe Exploration," *IEEE Conference on Decision and Control (CDC)*, Miami Beach, Florida (2018), vol. 57, pp. 6059–6066. doi: 10.1109/CDC.2018.8619572.
[25] R. Soloperto, M. A. Müller and F. Allgöwer, "Guaranteed closed-loop learning in model predictive control," *IEEE Trans. Autom. Control* **68**(2), 991–10066 (2023). doi: 10.1109/TAC.2022.3172453.
[26] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings and M. Diehl, "CasADi: A software framework for nonlinear optimization and optimal control," *Math. Program. Comput.* **11**(1), 1–36 (2019). doi: 10.1007/s12532-018-0139-4.
[27] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Math. Program.* **106**(1), 25–57 (2006). doi: 10.1007/s10107-004-0559-y.
[28] A. Carron, E. Arcari, M. Wermelinger, L. Hewing, M. Hutter and M. N. Zeilinger, "Data-driven model predictive control for trajectory tracking with a robotic arm," *IEEE Robot. Autom. Lett.* **4**(4), 3758–3765 (2019). doi: 10.1109/LRA.2019.2929987.
[29] R. B. Gramacy and D. W. Apley, "Local Gaussian process approximation for large computer experiments," *J. Comput. Graph. Stat.* **24**(2), 561–578 (2015). doi: 10.48550/arXiv.1303.0383.
[30] J. Kabzan, L. Hewing, A. Liniger and M. N. Zeilinger, "Learning-based model predictive control for autonomous racing," *IEEE Robot. Autom. Lett.* **4**(4), 3363–3370 (2019). doi: 10.1109/LRA.2019.2926677.
[31] M. Bauer, M. van der Wilk and C. E. Rasmussen, "Understanding Probabilistic Sparse Gaussian Process Approximations," *Proceedings of Advances in Neural Information Processing Systems*, Barcelona, Spain (2016), vol. 29, pp. 1533–1541. doi: 10.48550/arXiv.1606.04820.
[32] J. Rawlings, D. Q. Mayne and M. Diehl, *Model Predictive Control: Theory, Computation, and Design*, 2nd edition (Nob Hill Publishing, Santa Barbara, CA, 2017).
[33] D. Q. Mayne, J. B. Rawlings, C. V. Rao and P. O. M. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica* **36**(6), 789–814 (2000). doi: 10.1016/S0005-1098(99)00214-9.
[34] S. S. Keerthi and E. G. Gilbert, "Optimal infinite-horizon feedback laws for a general class of constrained discrete-time systems: Stability and moving-horizon approximations," *J. Optim. Theory Appl.* **57**(2), 265–293 (1988). doi: 10.1007/BF00938540.
[35] B. Houska, H. J. Ferreau and M. Diehl, "ACADO toolkit–An open source framework for automatic control and dynamic optimization," *Optim. Control Appl. Methods* **32**(3), 298–312 (2011). doi: 10.1002/oca.939.
[36] B. Houska, H. J. Ferreau and M. Diehl, "An auto-generated real-time iteration algorithm for nonlinear MPC in the microsecond range," *Automatica* **47**(10), 2279–2285 (2011). doi: 10.1016/j.automatica.2011.08.020.
[37] H. Ferreau, C. Kirches, A. Potschka, H. Bock and M. Diehl, "qpOASES: A parametric active-set algorithm for quadratic programming," *Math. Program. Comput.* **6**(4), 327–363 (2014). doi: 10.1007/s12532-014-0071-1.
[38] A. De Santis, B. Siciliano, A. De Luca and A. Bicchi, "An atlas of physical human-robot interaction," *Mech. Mach. Theory* **43**(3), 253–270 (2008). doi: 10.1016/j.mechmachtheory.2007.03.003.
[39] C. K. Williams and F. Vivarelli, "Upper and lower bounds on the learning curve for Gaussian processes," *Mach. Learn.* **40**(1), 77–102 (2000). doi: 10.1023/A:1007601601278.
[40] H. Ebel and P. Eberhard, "A comparative look at two formation control approaches based on optimization and algebraic graph theory," *Robot. Auton. Syst.* **136**(7), 103686 (2021). doi: 10.1016/j.robot.2020.103686.

Cite this article: H. Eschmann, H. Ebel and P. Eberhard (2023). "Exploration-exploitation-based trajectory tracking of mobile robots using Gaussian processes and model predictive control", Robotica **41**, 3040–3058. https://doi.org/10.1017/S0263574723000863