2015 European Control Conference (ECC)
July 15-17, 2015. Linz, Austria

Safe and Robust Learning Control with Gaussian Processes

Felix Berkenkamp and Angela P. Schoellig

Abstract— **This paper introduces a learning-based robust control algorithm that provides robust stability and performance guarantees during learning. The approach uses Gaussian process (GP) regression based on data gathered during operation to update an initial model of the system and to gradually decrease the uncertainty related to this model. Embedding this data-based update scheme in a robust control framework guarantees stability during the learning process. Traditional robust control approaches have not considered online adaptation of the model and its uncertainty before. As a result, their controllers do not improve performance during operation. Typical machine learning algorithms that have achieved similar high-performance behavior by adapting the model and controller online do not provide the guarantees presented in this paper. In particular, this paper considers a stabilization task, linearizes the nonlinear, GP-based model around a desired operating point, and solves a convex optimization problem to obtain a linear robust controller. The resulting performance improvements due to the learning-based controller are demonstrated in experiments on a quadrotor vehicle.**

I. Introduction

Mathematical models are an important prerequisite when designing model-based controllers using, for example, techniques such as Linear Quadratic Regulators [1] or Model Predictive Control [2]. The performance of the resulting controllers directly depends on the accuracy of the model. While established methods for system modeling and identification exist [3], [4], the resulting models are always an approximation of the real system behavior.

The goal of this paper is to use data gathered during operation to improve the model and, consequently, the control performance while providing stability guarantees during the learning process. Guaranteeing stability during learning has recently been stated as an open problem in robotics [5]. The proposed approach combines robust control theory with machine learning, where the latter updates the model and model uncertainty based on data obtained during operation. This information is used in the robust controller design to guarantee stability despite the present uncertainty (Fig. 1).

Machine learning methods have been used before to overcome the limitations of approximate models. These methods leverage online measurements to improve the control performance. One such example is iterative learning control (ILC), which achieves high-performance tracking by learning feed-forward inputs through repeated executions of a task [6].

Felix Berkenkamp is with ETH Zurich, Switzerland. Email: befelix@ethz.ch. Angela P. Schoellig is with the Dynamic Systems Lab (www.dynsyslab.org) at the University of Toronto Institute for Aerospace Studies (UTIAS), Canada. Email: schoellig@utias.utoronto.ca.

This work was supported by the Natural Sciences and Engineering Research Council of Canada under the grant RGPIN-2014-04634.

Associated video at: http://tiny.cc/ecc15_video.

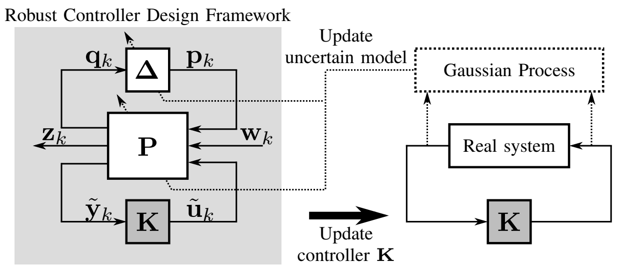

Fig. 1. Framework of the learning-based robust controller design. Using input-output data of the real system, the plant model $\mathbf{P}$ is updated and the uncertainty estimate $\mathbf{\Delta}$ is gradually reduced. As a result, the performance of the regularly updated robust controller $\mathbf{K}$ improves.

Other approaches improve the system model directly, such as Neural Networks (NN) [7]. Recently, the focus of the community has shifted to Gaussian Processes (GPs) [8], as they outperform NNs under certain conditions [9]. Learned GP models have been successfully used for dynamic programming [10] and to improve trajectory tracking of a robot traversing unknown, rough terrain [11]. GPs also provide uncertainty estimates for their predictions, which have been used in [12] to minimize the expected squared error in a model predictive control setting. While all these methods achieve impressive control performance after learning, they do not guarantee stability during the learning process. The reason for this is that these methods focus on updating the estimated model of the dynamics without explicitly considering the mismatch between the learned model and the real dynamics of the system.

In the control community, model errors have been explicitly considered in the field of robust control, established in the 1980s [13]. Robust control focuses on linear systems, incorporates an *a priori* estimate of the model uncertainty in the controller design, and guarantees stability and performance of the system for all modeled uncertainties [14]. However, this uncertainty estimate is not updated during operation. While the uncertainty specification provides stability guarantees, robust control methods may decrease overall controller performance, because the performance objective is minimized over all possible models that lie within the uncertainty specification.

In this paper, we combine machine learning with robust control to achieve both good controller performance after learning, inherited from machine learning, and stability and performance guarantees, inherited from robust control. The main challenge in combining the two approaches lies in identifying the unknown plant dynamics and the corresponding uncertainty in a way that can be used in a robust controller design framework. Classical offline system identification methods focus on fitting linear models to observed data using either time or frequency domain methods [3], [4]. While these approaches can be extended to include uncertainty estimates [15], [16], fitting a linear model to the unknown, nonlinear dynamics can lead to errors. In [17] a way to avert these errors and quantify the uncertainty was presented involving periodic inputs to the system. Such a solution is generally not possible in an online setting. Thus, to achieve bias-free online system identification, one must turn to nonlinear, nonparametric identification methods [18]. GPs are of particular interest, since they provide uncertainty estimates [8].

Combining robust stability with nonlinear, nonparametric system identification was explored in [19]. There, the learned dynamics were considered to be bounded disturbances and a linear controller for the nominal plant was used to guarantee robustness. Compared to our approach, this work did not adapt the *a priori* model uncertainty. In [20] robust stability guarantees were obtained for a nonlinear model predictive controller by switching between safe and nominal control actions based on level sets. This work only considered nominal performance. As a result, the performance of the uncertain system is not guaranteed.

Another large field of research related to our approach is (robust) adaptive control; because of space limitations, we do not refer to specific papers but want to highlight that the majority of work in this area considers the adaptation of a finite number of control parameters, while our approach considers nonparametric, nonlinear model learning based on GPs.

This paper uses GPs for online learning of the system dynamics. The learned GP model and its uncertainty estimates are linearized using properties of GPs that have not been applied to robust control before. The linearization point itself is considered to be uncertain and identified as well. Robust controller design methods are applied to the obtained uncertain model resulting in a convex optimization problem that is solvable online [21]. Initial simulation results of this approach were presented in [22]. This paper presents experimental results on quadrotor vehicles, see associated video at: http://tiny.cc/ecc15_video.

The remainder of this paper is structured as follows: in Sec. II the robust control problem is stated. GPs are introduced in Sec. III and their application to online model learning is shown in Sec. IV. Sec. V presents a robust control design based on the GP model of Sec. IV. A discussion of the presented approach is provided in Sec. VI. Finally, the method is applied to a quadrotor vehicle in Sec. VII, and conclusions are drawn in Sec. VIII.

II. Problem Statement

This section introduces the system model and the robust control problem considered in this paper.

Let $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{u} \in \mathbb{R}^m$ and $\mathbf{y} \in \mathbb{R}^p$ denote the system states, inputs and measured outputs, respectively. The system dynamics are separated into two components, a *known* function, $\mathbf{f}(\mathbf{x},\mathbf{u})$, derived, for example, from first principles, and an *unknown* function, $\mathbf{g}(\mathbf{x},\mathbf{u})$. The function $\mathbf{g}(\mathbf{x},\mathbf{u})$ represents unknown, deterministic dynamics that are not captured by the *a priori* model $\mathbf{f}(\mathbf{x},\mathbf{u})$. Both functions are assumed to be continuously differentiable, $\mathbf{f},\mathbf{g} \in C^1$. Measurements are made via a known matrix $\mathbf{C} \in \mathbb{R}^{p\times n}$ and are corrupted by zero-mean Gaussian noise with covariance matrix $\boldsymbol{\Sigma}_\omega$, $\boldsymbol{\omega} \sim \mathcal{N}(\mathbf{0},\boldsymbol{\Sigma}_\omega)$. The corresponding discrete-time system dynamics are given by

$$\begin{aligned} \mathbf{x}_{k+1} &= \underbrace{\mathbf{f}(\mathbf{x}_k,\mathbf{u}_k)}_{\textit{a priori}\text{ model}} + \underbrace{\mathbf{g}(\mathbf{x}_k,\mathbf{u}_k)}_{\text{unknown model}} \\ \mathbf{y}_k &= \mathbf{C}\mathbf{x}_k + \boldsymbol{\omega}_k, \end{aligned} \tag{1}$$

where k refers to the kth time step.

The goal of this paper is to find an estimate of the function $\mathbf{g}(\mathbf{x},\mathbf{u})$ and an associated confidence interval of this estimate from measurement data. This information is used to design a robust controller that stabilizes the system despite the uncertainty in the estimate. As the estimate of $\mathbf{g}(\mathbf{x},\mathbf{u})$ becomes more certain and accurate, the robust controller performance improves. In this paper, we model the unknown dynamics $\mathbf{g}(\mathbf{x},\mathbf{u})$ as a GP (see Sec. III) and consider linear robust control around a specific operating point, $\mathbf{x}_s$, with a corresponding steady-state input, $\mathbf{u}_s$, where

$$\mathbf{x}_s = \mathbf{f}(\mathbf{x}_s,\mathbf{u}_s) + \mathbf{g}(\mathbf{x}_s,\mathbf{u}_s). \tag{2}$$

Both $\mathbf{x}_s$ and $\mathbf{u}_s$ are initially unknown and are also estimated from measurement data, see Sec. IV-A.

Throughout this paper, we assume that noisy full-state information is available; that is, $\mathbf{C}$ is full-rank or, without loss of generality, $\mathbf{C} = \mathbf{I}$. While it is possible to extend the obtained results to partial-state information, this is outside of the scope of this paper.

III. Gaussian Processes (GPs)

In this section, we introduce the basic properties of GPs. GPs are used in Sec. IV to model the unknown dynamics, $\mathbf{g}(\mathbf{x},\mathbf{u})$, see (1).

GPs are a popular choice for nonparametric regression in machine learning, where the goal is to find an approximation of a nonlinear map, $g(\mathbf{a}) : \mathbb{R}^{\dim(\mathbf{a})} \mapsto \mathbb{R}$, from an input vector $\mathbf{a}$ to the function value $g(\mathbf{a})$. This is accomplished by assuming that function values $g(\mathbf{a})$, associated with different values of $\mathbf{a}$, are random variables and that any finite number of these random variables have a joint Gaussian distribution depending on the values of $\mathbf{a}$ [8].

For the nonparametric regression we need to define a prior for the mean of $g(\mathbf{a})$ and for the covariance between any two function values, $g(\mathbf{a}_i)$ and $g(\mathbf{a}_j)$, $i,j \in \mathbb{N}$. The latter is also known as the kernel. In this work, the mean is assumed to be zero, since GPs are used to approximate the dynamics $\mathbf{g}(\mathbf{x},\mathbf{u})$ in (1) for which no prior knowledge is available.

In the following discussion, we choose the often used squared-exponential function as the covariance function, [8]; however, any kernel with continuous first derivative can be used in general. For a squared-exponential function, input vectors $\mathbf{a}$ that are close to each other (in terms of their scaled, squared distance) are assumed to have similar function values. The covariance between two data points, $g(\mathbf{a}_i)$ and $g(\mathbf{a}_j)$, is given by

$$k(\mathbf{a}_i, \mathbf{a}_j) = \sigma_\eta^2 \exp\left(-\frac{1}{2}(\mathbf{a}_i - \mathbf{a}_j)^T \mathbf{M}^{-2}(\mathbf{a}_i - \mathbf{a}_j)\right) + \delta_{ij}\sigma_\omega^2, \tag{3}$$

where $\delta_{ij} = 1$ if $i = j$ and 0 otherwise. The covariance is parameterized by three hyperparameters: measurement noise, σ_ω^2, process variation, σ_η^2, and length scales, $\mathbf{l} \in \mathbb{R}^{\dim(\mathbf{a})}$, which are the diagonal elements of the diagonal matrix $\mathbf{M}$, $\mathbf{M} = \mathrm{diag}(\mathbf{l})$, and correspond to the rate of change of the function g with respect to $\mathbf{a}$. These hyperparameters are learned from observed data by solving a maximum log-likelihood problem using gradient ascent methods [8].

A. Prediction

The GP framework above can be used to predict the function value of $g(\mathbf{a}^*)$ at an arbitrary input, $\mathbf{a}^*$, based on a set of N past observations, $\mathcal{D} = \{\mathbf{a}_i, \hat{g}(\mathbf{a}_i)\}_{i=1}^N$. We assume that observations are noisy measurements of the true function value, $g(\mathbf{a})$; that is, $\hat{g}(\mathbf{a}) = g(\mathbf{a}) + \omega$ with $\omega \sim \mathcal{N}(0, \sigma_\omega^2)$. The joint probability distribution of the function value $g(\mathbf{a}^*)$ and the observed data is given by

$$\begin{bmatrix} \hat{\mathbf{g}} \\ g(\mathbf{a}^*) \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}_N, \begin{bmatrix} \overline{\mathbf{K}} & \mathbf{k}^{\mathrm{T}}(\mathbf{a}^*) \\ \mathbf{k}(\mathbf{a}^*) & k(\mathbf{a}^*, \mathbf{a}^*) \end{bmatrix}\right), \tag{4}$$

where $\hat{\mathbf{g}} = \left[\hat{g}(\mathbf{a}_1), \dots, \hat{g}(\mathbf{a}_N)\right]^{\mathrm{T}}$ is the vector of observed function values and $\mathbf{0}_N \in \mathbb{R}^N$ is a vector of zeros. The covariance matrix $\overline{\mathbf{K}} \in \mathbb{R}^{N \times N}$ has entries $\overline{\mathbf{K}}_{(i,j)} = k(\mathbf{a}_i, \mathbf{a}_j)$, $i, j \in \{1, \dots, N\}$, and $\mathbf{k}(\mathbf{a}^*) = \left[k(\mathbf{a}^*, \mathbf{a}_1), \dots, k(\mathbf{a}^*, \mathbf{a}_N)\right]$ contains the covariances between the new input $\mathbf{a}^*$ and the observed data points in $\mathcal{D}$. Using the properties of joint Gaussian distributions, the prediction of $g(\mathbf{a}^*)$ conditioned on the data set $\mathcal{D}$ is given by $g(\mathbf{a}^*)|\mathcal{D} \sim \mathcal{N}(\mu(\mathbf{a}^*), \sigma^2(\mathbf{a}^*))$ (cf. [8]) with

$$\mu(\mathbf{a}^*) = \mathbf{k}(\mathbf{a}^*)\overline{\mathbf{K}}^{-1}\hat{\mathbf{g}}, \tag{5}$$

$$\sigma^2(\mathbf{a}^*) = k(\mathbf{a}^*, \mathbf{a}^*) - \mathbf{k}(\mathbf{a}^*)\overline{\mathbf{K}}^{-1}\mathbf{k}^{\mathrm{T}}(\mathbf{a}^*). \tag{6}$$

The prediction depends on the inverse of the $(N \times N)$ matrix $\overline{\mathbf{K}}$, which is an expensive $\mathcal{O}(N^3)$ computation. However, in order to make predictions, the inverse must be computed only once for a given data set.

So far, we have considered a scalar function, $g(\mathbf{a})$. The extension to a vector-valued function, as required for approximating $\mathbf{g}(\mathbf{x}, \mathbf{u})$ in (1), can be done by extending (4), resulting in a matrix $\overline{\mathbf{K}}$ of size $(nN \times nN)$. Such a matrix is usually too big to be inverted efficiently. To maintain computational feasibility, independent GPs are trained for each dimension of $\mathbf{g}(\mathbf{x}, \mathbf{u})$. As a result, any additional information that could be gained from the correlation between the components in $\mathbf{g}(\mathbf{x}, \mathbf{u})$ (for example, introduced by the measurement noise) are neglected.

B. Derivatives

An additional characteristic of the GP framework important for Sec. IV is that the derivative of a GP is a GP as well [8]. This follows from the fact that the derivative is a linear operator. From (4), we get

$$\begin{bmatrix} \hat{\mathbf{g}} \\ \frac{\partial g(\mathbf{a})}{\partial \mathbf{a}}\Big|_{\mathbf{a}^*} \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}_N, \begin{bmatrix} \overline{\mathbf{K}} & \frac{\partial \mathbf{k}^{\mathrm{T}}(\mathbf{a})}{\partial \mathbf{a}}\Big|_{\mathbf{a}^*} \\ \frac{\partial \mathbf{k}(\mathbf{a})}{\partial \mathbf{a}}\Big|_{\mathbf{a}^*} & \frac{\partial^2 k(\mathbf{a}, \mathbf{a})}{\partial \mathbf{a} \partial \mathbf{a}}\Big|_{\mathbf{a}^*} \end{bmatrix}\right), \tag{7}$$

where $h(\mathbf{a})|_{\mathbf{a}^*}$ means $h(\mathbf{a})$ evaluated at $\mathbf{a} = \mathbf{a}^*$. The derivatives of the squared-exponential function are given by

$$\frac{\partial \mathbf{k}_{(i)}(\mathbf{a})}{\partial \mathbf{a}}\Big|_{\mathbf{a}^*} = \mathbf{M}^{-2}(\mathbf{a}_i - \mathbf{a}^*)k(\mathbf{a}^*, \mathbf{a}_i), \tag{8}$$

$$\frac{\partial^2 k(\mathbf{a}, \mathbf{a})}{\partial \mathbf{a} \partial \mathbf{a}}\Big|_{\mathbf{a}^*} = \sigma_\eta^2 \mathbf{M}^{-2}, \tag{9}$$

where $\mathbf{k}_{(i)}(\mathbf{a})$ is the ith element of $\mathbf{k}(\mathbf{a})$ and (8) represents the ith column of $\partial \mathbf{k}(\mathbf{a})/\partial \mathbf{a}$.

In the literature, this property has mainly been used to include derivative observations; however, following the same reasoning as in the previous section predictions of the derivatives are given by $\frac{\partial g(\mathbf{a})}{\partial \mathbf{a}}\big|_{\mathbf{a}^*}|\mathcal{D} \sim \mathcal{N}(\boldsymbol{\mu}'(\mathbf{a}^*), \boldsymbol{\Sigma}'(\mathbf{a}^*))$ with

$$\boldsymbol{\mu}'(\mathbf{a}^*) = \frac{\partial \mathbf{k}(\mathbf{a})}{\partial \mathbf{a}}\Big|_{\mathbf{a}^*} \overline{\mathbf{K}}^{-1}\hat{\mathbf{g}}, \tag{10}$$

$$\boldsymbol{\Sigma}'(\mathbf{a}^*) = \frac{\partial^2 k(\mathbf{a}, \mathbf{a})}{\partial \mathbf{a} \partial \mathbf{a}}\Big|_{\mathbf{a}^*} - \frac{\partial \mathbf{k}(\mathbf{a})}{\partial \mathbf{a}}\Big|_{\mathbf{a}^*} \overline{\mathbf{K}}^{-1} \left(\frac{\partial \mathbf{k}(\mathbf{a})}{\partial \mathbf{a}}\Big|_{\mathbf{a}^*}\right)^{\mathrm{T}}. \tag{11}$$

From (11) and (9) it can be seen that $\sigma_\eta^2 \mathbf{M}^{-2}$ is the prior variance of the first derivative, which confirms the interpretation of the length-scale hyperparameters as corresponding to the rate of change of g.

IV. Model Update

In this section, we show how GPs, introduced in Sec. III as a means to approximate $\mathbf{g}(\mathbf{x}, \mathbf{u})$ in (1), can be used to derive a linear model of the unknown dynamics $\mathbf{g}(\mathbf{x}, \mathbf{u})$ around a linearization point. This linear model and the associated uncertainty estimates are used in Sec. V to update the robust controller, see Fig. 1.

We use separate GPs, g_i, to identify each dimension of the unknown dynamics $\mathbf{g}(\mathbf{x}, \mathbf{u})$; that is, g_i corresponds to the ith element of the model error, $\mathbf{g}_{(i)}(\mathbf{x}, \mathbf{u})$. The input vector, $\mathbf{a}_k = (\mathbf{x}_k, \mathbf{u}_k)$, is the same for each GP and consists of the current state and input, while the training target of the GPs is the model error, $\hat{\mathbf{g}}(\mathbf{a}_k) = \mathbf{x}_{k+1} - \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)$, obtained from measurements, cf. (1). As a result, we can predict the function value and the derivative of the unknown dynamics $\mathbf{g}(\mathbf{x}, \mathbf{u})$ using (5)-(6) and (10)-(11). As mentioned in Sec. II, this requires the full state $\mathbf{x}$ to be known. Extensions to partial-state measurements can be found in [23], [24].

A. Operating Point

The goal of the control problem considered in this paper is to stabilize the system around a desired operating point. Since the *a priori* operating point of the known dynamics $\mathbf{f}$

with $\bar{\mathbf{x}}_s = \mathbf{f}(\bar{\mathbf{x}}_s, \bar{\mathbf{u}}_s)$ is not necessarily equal to (2), an *optional* step that reduces steady-state errors and further improves the control performance is to regularly update the operating point as the unknown dynamics are learned.

Given the prior operating point $\bar{\mathbf{x}}_s$, an updated steady-state operating point that satisfies (2) with the mean estimate for $\mathbf{g}$ from (5) is found via the optimization problem

$$\begin{aligned}&\min_{\mathbf{x}_s, \mathbf{u}_s} \|\bar{\mathbf{x}}_s - \mathbf{x}_s\|_{\mathbf{Q}} \\ &\text{subject to } \; \mathbf{x}_s = \mathbf{f}(\mathbf{x}_s, \mathbf{u}_s) + \boldsymbol{\mu}(\mathbf{x}_s, \mathbf{u}_s),\end{aligned} \tag{12}$$

where $\boldsymbol{\mu} \in \mathbb{R}^n$ is the learned mean consisting of the individual predictions for each dimension according to (5) and $\|\mathbf{x}\|_{\mathbf{Q}} = \mathbf{x}^{\mathrm{T}}\mathbf{Q}\mathbf{x}$ refers to the 2-norm with a positive-definite weighting matrix $\mathbf{Q}$ that accounts for the prior uncertainty in the different states and for scaling. If $\mathbf{x}_s$ is known, a simpler optimization problem can be formulated, which is solvable via gradient descent:

$$\min_{\mathbf{u}_s} \|\mathbf{x}_s - \mathbf{f}(\mathbf{x}_s, \mathbf{u}_s) - \boldsymbol{\mu}(\mathbf{x}_s, \mathbf{u}_s)\|_{\mathbf{Q}}. \tag{13}$$

B. Linearization

In the previous section, an operating point including steady-state input was found. The second step of the model update (see Fig. 1) linearizes system (1) around this operating point and obtains elementwise confidence intervals.

The dynamics (1) are linearized around the operating point $\mathbf{a}^* = (\mathbf{x}_s, \mathbf{u}_s)$ using the properties of GPs presented in Sec. III-B. We denote the derivative of the ith GP, g_i, with respect to $\mathbf{a}$, $\mathbf{a} = (\mathbf{x}, \mathbf{u})$, by $\frac{\partial g_i(\mathbf{a})}{\partial \mathbf{a}}\big|_{\mathbf{a}^*} \sim \mathcal{N}(\boldsymbol{\mu}_i'(\mathbf{a}^*), \boldsymbol{\Sigma}_i'(\mathbf{a}^*))$, and obtain

$$\begin{bmatrix}\mathbf{A} & \mathbf{B}\end{bmatrix} = \left.\frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{u})}{\partial \mathbf{a}}\right|_{\mathbf{a}^*} + \begin{bmatrix}\boldsymbol{\mu}_1'(\mathbf{a}^*), \ldots, \boldsymbol{\mu}_n'(\mathbf{a}^*)\end{bmatrix}^{\mathrm{T}}. \tag{14}$$

The uncertainty is represented by

$$\begin{bmatrix}\mathbf{A}_u & \mathbf{B}_u\end{bmatrix} = 2 \begin{bmatrix}\sqrt{\operatorname{diag}(\boldsymbol{\Sigma}_1'(\mathbf{a}^*))} \\ \vdots \\ \sqrt{\operatorname{diag}(\boldsymbol{\Sigma}_n'(\mathbf{a}^*))}\end{bmatrix}, \tag{15}$$

where $\operatorname{diag}(\mathbf{M})$ is a row vector consisting of the diagonal elements of the matrix $\mathbf{M}$, and the square root in (15) acts elementwise. Each element of the matrices $\mathbf{A}_u$ and $\mathbf{B}_u$ in (15) is equal to the 2σ (95%) confidence interval for the corresponding element in $\mathbf{A}$ and $\mathbf{B}$. Truncating the elementwise uncertainty to 2σ may lead to problems in very rare cases when the disturbance is outside of the 2σ interval. Consequently, the linearized dynamics are given by

$$\tilde{\mathbf{x}}_{k+1} = (\mathbf{A} + \mathbf{A}_u \circ \boldsymbol{\Delta}_x)\tilde{\mathbf{x}}_k + (\mathbf{B} + \mathbf{B}_u \circ \boldsymbol{\Delta}_u)\tilde{\mathbf{u}}_k, \tag{16}$$

where $\tilde{\mathbf{x}}_k = \mathbf{x}_k - \mathbf{x}_s$ and $\tilde{\mathbf{u}}_k = \mathbf{u}_k - \mathbf{u}_s$ are deviations around the linearization point, $\circ$ is the elementwise matrix product, and $\boldsymbol{\Delta}_x$ and $\boldsymbol{\Delta}_u$ represent interval uncertainties with matrix entries in $[-1, 1]$; that is, the state matrix is $(\mathbf{A} + \mathbf{A}_u)$, where every element in $\mathbf{A}_u$ is independently scaled by a factor in $[-1, 1]$.

V. Robust Control

We derive a robust controller of the form $\mathbf{u}_k = \mathbf{K}(\mathbf{x}_k - \mathbf{x}_s) + \mathbf{u}_s$ for the linearized system in (16), which stabilizes the system despite the uncertainties introduced by $\boldsymbol{\Delta}_x$ and $\boldsymbol{\Delta}_u$.

In the general framework of robust control (see Fig. 1), the objective is to minimize an error signal, $\mathbf{z} \in \mathbb{R}^r$, caused by a disturbance, $\mathbf{w} \in \mathbb{R}^q$, for all possible uncertainties introduced via an uncertain signal, $\mathbf{p} \in \mathbb{R}^f$, cf. [14].

$$\tilde{\mathbf{x}}_{k+1} = \mathbf{A}\tilde{\mathbf{x}}_k + \mathbf{B}\tilde{\mathbf{u}}_k + \mathbf{B}_w\mathbf{w}_k + \mathbf{B}_p\mathbf{p}_k \tag{17}$$

$$\mathbf{z}_k = \mathbf{C}_z\tilde{\mathbf{x}}_k + \mathbf{D}_z\tilde{\mathbf{u}}_k \tag{18}$$

$$\mathbf{q}_k = \mathbf{C}_q\tilde{\mathbf{x}}_k + \mathbf{D}_q\tilde{\mathbf{u}}_k \tag{19}$$

$$\mathbf{p}_k = \alpha\boldsymbol{\Delta}\mathbf{q}_k, \tag{20}$$

where $\boldsymbol{\Delta} = \operatorname{diag}(\delta_1, \ldots, \delta_f)$ with $|\delta_i| \leq 1, \; i = 1, \ldots, f$, together with $\mathbf{C}_q$, $\mathbf{D}_q$, and $\mathbf{B}_p$ represent the uncertainty in the system (16), $(\mathbf{A}, \mathbf{B})$ is the nominal system from (14), and $\alpha = 1$. The uncertainty (15) can be represented by choosing

$$\mathbf{C}_q = \begin{bmatrix}\operatorname{diag}((\mathbf{A}_u)_{(1,:)}) \\ \vdots \\ \operatorname{diag}((\mathbf{A}_u)_{(n,:)}) \\ \mathbf{0}_{nm\times n}\end{bmatrix}, \mathbf{D}_q = \begin{bmatrix}\mathbf{0}_{n^2\times m} \\ \operatorname{diag}((\mathbf{B}_u)_{(1,:)}) \\ \vdots \\ \operatorname{diag}((\mathbf{B}_u)_{(n,:)})\end{bmatrix}, \tag{21}$$

where $(\mathbf{A}_u)_{(i,:)}$ is the ith row of $\mathbf{A}_u$, $\mathbf{I}_n \in \mathbb{R}^{n\times n}$ is the identity matrix, $\mathbf{0}_{i\times j}, \mathbf{1}_{i\times j} \in \mathbb{R}^{i\times j}$ are matrices with entries zero or one, respectively, and $\mathbf{B}_p = \begin{bmatrix}\mathbf{I}_n \otimes \mathbf{1}_{1\times n}, \mathbf{I}_n \otimes \mathbf{1}_{1\times m}\end{bmatrix}$, where $\otimes$ denotes the Kronecker product.

The matrices $\mathbf{B}_w, \mathbf{C}_z$ and $\mathbf{D}_z$ define the control objective to be minimized by the robust controller. Possible choices can be found in [14]. The disturbance input matrix, $\mathbf{B}_w$, can be interpreted as the expected disturbances at the operating point $(\mathbf{x}_s, \mathbf{u}_s)$. We represent increasing knowledge about the operating point by $\mathbf{B}_w = \operatorname{diag}(\sigma_1, \ldots, \sigma_n)$, where σ_i is the variance of the ith GP, g_i, from (6).

A popular measure for the error signal, $\mathbf{z}$, is the $\mathcal{H}_2$ system norm, which is a generalization of the well-known Linear Quadratic Regulator. The corresponding optimization problem is $\min_{\mathbf{K}} \max_{\boldsymbol{\Delta}} \|T_{zw}\|_2$, where the transfer function from $\mathbf{w}_k$ to $\mathbf{z}_k$, T_{zw}, depends on the controller $\mathbf{K}$ and on $\boldsymbol{\Delta}$.

Methods to solve this kind of problem are given in [14]. The computations in this paper are based on [21], which derives a controller that guarantees stability and performance for all possible $\boldsymbol{\Delta}$. While the method in [21] is potentially more conservative than others in [14], it obtains the discrete-time controller by solving a convex optimization problem in terms of linear matrix inequalities (LMIs). Efficient solvers for LMIs exist [25], making this technique applicable to online applications as considered in this paper. In fact, in [26] this technique was applied to find a model predictive controller, which required solving an LMI at every time step.

We modify the uncertainty representation in [21] to find an $\mathcal{H}_2$ state-feedback controller for our uncertainty definition in (17)-(20). In the following, symmetric matrix elements are denoted by $\bullet$ and $\mathbf{P} > 0$ means that $\mathbf{P}$ is positive definite.

Theorem 1: System (17)-(20) is robustly stable under the state feedback law $\tilde{\mathbf{u}}_k = \mathbf{K}\tilde{\mathbf{x}}_k$ with $\mathbf{K} = \mathbf{R}\mathbf{Q}^{-1}$ and $\|T_{zw}\|_2^2 < \gamma$ if the following optimization problem is feasible:

$$\min_{\mathbf{W}=\mathbf{W}^{\mathrm{T}},\mathbf{Q}=\mathbf{Q}^{\mathrm{T}},\mathbf{R},\mathbf{\Lambda}=\mathrm{diag}(\tau_1,\ldots,\tau_f),\gamma,\beta=1} \gamma \qquad (22)$$

subject to

$$\mathrm{trace}(\mathbf{W}) < \gamma, \qquad \begin{bmatrix} \mathbf{W} & \mathbf{C}_z\mathbf{Q} + \mathbf{D}_z\mathbf{R} \\ \bullet & \mathbf{Q} \end{bmatrix} > 0,$$

$$\begin{bmatrix} \mathbf{Q} & \bullet & \bullet & \bullet & \bullet \\ \mathbf{0}_{q\times n} & \mathbf{I}_q & \bullet & \bullet & \bullet \\ \mathbf{0}_{f\times n} & \mathbf{0}_{f\times q} & \mathbf{\Lambda} & \bullet & \bullet \\ \mathbf{A}\mathbf{Q} + \mathbf{B}\mathbf{R} & \mathbf{B}_w & \mathbf{B}_p\mathbf{\Lambda} & \mathbf{Q} & \bullet \\ \mathbf{C}_q\mathbf{Q} + \mathbf{D}_q\mathbf{R} & \mathbf{0}_{f\times q} & \mathbf{0}_{f\times f} & \mathbf{0}_{f\times n} & \beta\mathbf{\Lambda} \end{bmatrix} > 0.$$

Proof: The proof can be found in [27]. ∎

For $\beta = \alpha^{-2} = 1$, the resulting closed-loop system is robustly stable for all uncertainties. Whenever this optimization problem is infeasible, a line search maximizing α with $0 \leq \alpha < 1$ leads to the best possible controller given the assumptions in [21]. The control matrix, $\mathbf{K}$ from Thm. 1, is recalculated regularly based on the most up-to-date model obtained from Sec. IV. The control input for the nonlinear system (1) is given by $\mathbf{u}_k = \mathbf{K}(\mathbf{x}_k - \mathbf{x}_s) + \mathbf{u}_s$.

VI. Discussion

To initialize the learning-based robust controller, the hyperparameters of the kernel function (3) must be specified as they provide the initial model uncertainty. There are two options to do this: first, it is typical for robust control to assume that bounds for the system uncertainty are known *a priori*. This prior information can be used to determine the hyperparameters and to calculate the initial controller. Second, for robotics applications it is typical to calculate the initial hyperparameters based on experimental data obtained from either an expert controlling the system or from autonomous operation where the controller is allowed to fail. As for all system identification methods, the system needs to be excited sufficiently, so that the hyperparameters reflect the true uncertainty in the system. Whichever method is chosen, in the long term additional experimental data becomes available and can be used for controller improvements.

The proposed approach learns a linear model around a desired operating point. The controller only stabilizes the system in the vicinity of the operating point. For global stability, one would have to turn to nonlinear robust control. However, since we use a nonlinear method for the model update the approach presented in this paper is extendable to robust tracking of nonlinear systems or gain scheduling.

Lastly, if the functional form of $\mathbf{g}(\mathbf{x},\mathbf{u})$ in (1) is known, other methods that take this additional information into account, such as (extended) Kalman Filters, may be better suited for the system identification step. However, if $\mathbf{g}(\mathbf{x},\mathbf{u})$ is unknown and nonlinear, the proposed method has an advantage: the hyperparameters define a region around the operating point in which the system behaves linearly and the GPs only learn from data points within this linear domain. Linear methods, such as the Kalman filter, would try to fit a linear model to the entire state space, which would degrade the accuracy of the estimated model.

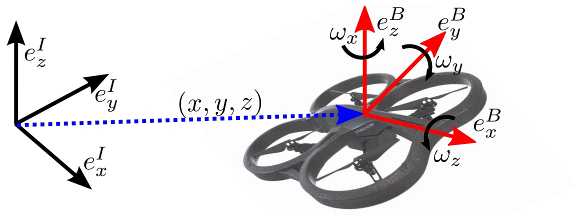

Fig. 2. Coordinate frames and variable definitions for the quadrotor.

VII. Quadrotor Experiments

In this section, we demonstrate our approach on a commercial quadrotor, the AR.Drone 2.0 from Parrot. The quadrotor is well suited as a test platform for our algorithm. Its dynamics are highly nonlinear and include aerodynamic effects that are difficult to model. Moreover, the AR.Drone comes with an unknown on-board controller. Both effects are learned and compensated for by our robust learning controller. A video of the results can be found at: http://tiny.cc/ecc15_video.

The quadrotor dynamics are described by 12 states (see Fig. 2): positions in the global frame I, (x,y,z); velocities, $(\dot{x},\dot{y},\dot{z})$; ZYX-Euler angles, (ψ,θ,ϕ); and, angular velocities in the body frame B, $(\omega_x,\omega_y,\omega_z)$. Measurements of all states are available from an overhead motion capture camera system. The quadrotor's on-board controller has four inputs: desired roll, ϕ_{des}; desired pitch, θ_{des}; desired angular velocity around e_z^B, $\omega_{z,\mathrm{des}}$; and, desired z-velocity, $\dot{z}_{\mathrm{des}}$.

The goal is to robustly stabilize the position of the quadrotor. For the sake of simplicity, we focus on the x,y-position control, while the z-position and the yaw are kept constant by separately learned robust controllers. To show the broad applicability of the presented approach, we make the (unrealistic) assumption that no prior model information is available; that is, $\mathbf{f}(\mathbf{x}_k,\mathbf{u}_k) = \mathbf{x}_k$. To account for the dynamics of the on-board controller and delays in the command transmission, the GP input vector is extended with two past inputs, $\mathbf{a}_k = (\mathbf{x}_k,\mathbf{u}_k,\mathbf{u}_{k-1},\mathbf{u}_{k-2})$.

The quadrotor is flown manually, and state and input data is collected at 80 Hz. The hyperparameters are learned using the first 800 data points. A Gaussian Process is trained using $N = 800$, 1000, 2000 and 3000 data points following the procedure in Sec. III. For each of these data sets a robust $\mathcal{H}_2$ controller is obtained, cf. Sec. V. We chose a simple quadratic cost to balance tracking errors and input magnitude: $\|\mathbf{x}-\mathbf{x}_s\|_{\mathbf{Q}'} + \|\mathbf{u}-\mathbf{u}_s\|_{\mathbf{R}'}$, where the positive-definite, diagonal matrices $\mathbf{Q}'$ and $\mathbf{R}'$ correspond to a weighting of state and input costs. The cost matrices in Sec. V then are

$$\mathbf{C}_z = \begin{bmatrix} \mathbf{Q}'^{\frac{1}{2}} \\ \mathbf{0}_{m\times n} \end{bmatrix} \quad \text{and} \quad \mathbf{D}_z = \begin{bmatrix} \mathbf{0}_{n\times m} \\ \mathbf{R}'^{\frac{1}{2}} \end{bmatrix}. \qquad (23)$$

Calculating the control law from (22) takes $\sim$ 1s on a 2GHz dual core processor. Updates to the controller can be done at low frequencies, since any controller obtained from Thm. 1 locally stabilizes the true system in (1).

The step responses of the different learned controllers are shown in Fig. 3. Given the cost function above, the tracking

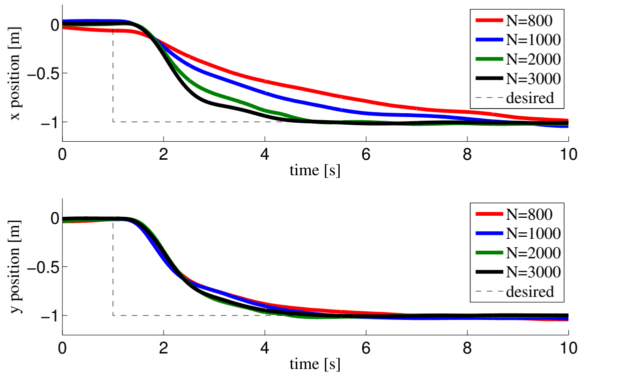

Fig. 3. Quadrotor response to a simultaneous 1 m step in the x- and y-reference position. The performance of the controller increases (that is, lower tracking error) as the Gaussian process learns the system dynamics from an increasing number of input-output samples N. After 3000 samples (corresponding to 37.5 s of flight time) the controller knows the quadrotor dynamics well and achieves a low tracking error (black, solid line).

error decreases as we learn a more accurate model of the system dynamics, resulting in a more aggressive controller with decreasing settling time as N increases. During the first 800 samples the y-direction of the system was excited more than the x-direction. As a result, the GP is is more certain about the dynamics in the y-direction, leading to an initial controller with higher performance in y-direction than in x-direction. During the learning process, the GP obtains more information about the x-direction and finally achieves similar performance in both x- and y-direction, which is expected for the symmetric quadrotor platform. Overall, the system remains robustly stable, while its performance increases.

VIII. Conclusion

In this paper, a method that combines online learning with robust control theory has been introduced with the goal of designing a learning controller that guarantees stability while gradually improving performance. A Gaussian Process (GP) is used to learn a nonlinear model of the unknown dynamics and corresponding uncertainty estimates. Based on this model, the operating point of the system is updated, and a linearization of the learned system model about this point is obtained including uncertainty information. Finally, a controller that is robust to the learned model uncertainties is calculated by solving a convex optimization problem. This controller is updated as more accurate and certain models of the system become available. Experiments on a quadrotor vehicle showed that the controller performance improved as more data became available. Ultimately, the GP framework has proven to be a powerful tool to combine nonlinear learning methods with standard robust control theory.

References

[1] B. Anderson and J. B. Moore, *Optimal control: linear quadratic methods*. Courier Dover Publications, 2007.

[2] E. F. Camacho and C. B. Alba, *Model predictive control*. Springer, 2013.

[3] L. Ljung, *System Identification: Theory for the User*. Pearson Education, 1998.

[4] R. Pintelon and J. Schoukens, *System identification: a frequency domain approach*. John Wiley & Sons, 2012.

[5] S. Schaal and C. G. Atkeson, "Learning control in robotics," *IEEE Robotics & Automation Magazine*, vol. 17, no. 2, pp. 20–29, 2010.

[6] D. Bristow, M. Tharayil, and A. Alleyne, "A survey of iterative learning control," *IEEE Control Systems*, vol. 26, no. 3, pp. 96–114, 2006.

[7] K. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Transactions on Neural Networks*, vol. 1, no. 1, pp. 4–27, 1990.

[8] C. E. Rasmussen and C. K. Williams, *Gaussian processes for machine learning*. Cambridge [MA]: MIT Press, 2006.

[9] R. C. Grande, G. Chowdhary, and J. P. How, "Experimental validation of bayesian nonparametric adaptive control using Gaussian processes," *Journal of Aerospace Information Systems*, vol. 11, no. 9, pp. 565–578, 2014.

[10] M. P. Deisenroth, J. Peters, and C. E. Rasmussen, "Approximate dynamic programming with Gaussian processes," in *Proc. of the IEEE American Control Conference (ACC)*, 2008, pp. 4480–4485.

[11] C. J. Ostafew, A. P. Schoellig, and Timothy D. Barfoot, "Learning-based nonlinear model predictive control to improve vision-based mobile robot path-tracking in challenging outdoor environments," in *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, 2014, pp. 4029–4036.

[12] R. Murray-Smith and D. Sbarbaro, "Nonlinear adaptive control using non-parametric Gaussian process prior models," in *Proc. of the IFAC World Congress on Automatic Control*, 2002, pp. 1038–1038.

[13] K. Zhou and J. C. Doyle, *Essentials of robust control*. Prentice hall Upper Saddle River, NJ, 1998, vol. 104.

[14] S. Skogestad and I. Postlethwaite, *Multivariable Feedback Control: Analysis and Design*. Wiley New York, 2007, vol. 2.

[15] P. M. J. Van Den Hof and R. J. P. Schrama, "Identification and control – closed-loop issues," *Automatica*, vol. 31, no. 12, pp. 1751–1770, 1995.

[16] S. G. Douma and P. M. J. Van den Hof, "Relations between uncertainty structures in identification for robust control," *Automatica*, vol. 41, no. 3, pp. 439–457, 2005.

[17] J. Schoukens, T. Dobrowiecki, and R. Pintelon, "Parametric and non-parametric identification of linear systems in the presence of nonlinear distortions – a frequency domain approach," *IEEE Transactions on Automatic Control*, vol. 43, no. 2, pp. 176–190, 1998.

[18] J. Sjöberg, Q. Zhang, L. Ljung, A. Benveniste, B. Delyon, P.-Y. Glorennec, H. Hjalmarsson, and A. Juditsky, "Nonlinear black-box modeling in system identification: a unified overview," *Automatica*, vol. 31, no. 12, pp. 1691–1724, 1995.

[19] A. Aswani, H. Gonzalez, S. S. Sastry, and C. Tomlin, "Provably safe and robust learning-based model predictive control," *Automatica*, vol. 49, no. 5, pp. 1216–1226, 2013.

[20] A. K. Akametalu, S. Kaynama, J. F. Fisac, M. N. Zeilinger, J. H. Gillula, and C. J. Tomlin, "Reachability-based safe learning with Gaussian processes," in *Proc. of the IEEE Conference on Decisions and Control (CDC)*, 2014.

[21] N. Bedioui, S. Salhi, and M. Ksouri, "H_2 performance via static output feedback for a class of nonlinear systems," in *Proc. of the IEEE International Conference on Signals, Circuits and Systems (SCS)*, 2009, pp. 1–6.

[22] F. Berkenkamp and A. P. Schoellig, "Learning-based robust control: Guaranteeing stability while improving performance," in *Workshop on Machine Learning in Planning and Control of Robot Motion, Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2014.

[23] J. Ko and D. Fox, "Learning gp-BayesFilters via gaussian process latent variable models," *Autonomous Robots*, vol. 30, no. 1, pp. 3–23, 2011.

[24] R. D. Turner, M. P. Deisenroth, and C. E. Rasmussen, "State-space inference and learning with Gaussian processes," in *Proc. of the International Conference on Artificial Intelligence and Statistics*, 2010, pp. 868–875.

[25] S. P. Boyd, *Linear matrix inequalities in system and control theory*. Philadelphia: Society for Industrial and Applied Mathematics (SIAM), 1994, vol. 15.

[26] M. V. Kothare, V. Balakrishnan, and M. Morari, "Robust constrained model predictive control using linear matrix inequalities," *Automatica*, vol. 32, no. 10, pp. 1361–1379, 1996.

[27] F. Berkenkamp and A. P. Schoellig, "Derivation of a linear, robust $\mathcal{H}_2$ controller for systems with parametric uncertainty," *ETH Zürich*, Tech. Rep., 2015, DOI: 10.3929/ethz-a-010405770.