Data-Driven MPC for Quadrotors

Guillem Torrente*, Elia Kaufmann*, Philipp Föhn, Davide Scaramuzza

***Abstract*—Aerodynamic forces render accurate high-speed trajectory tracking with quadrotors extremely challenging. These complex aerodynamic effects become a significant disturbance at high speeds, introducing large positional tracking errors, and are extremely difficult to model. To fly at high speeds, feedback control must be able to account for these aerodynamic effects in real-time. This necessitates a modelling procedure that is both accurate and efficient to evaluate. Therefore, we present an approach to model aerodynamic effects using Gaussian Processes, which we incorporate into a Model Predictive Controller to achieve efficient and precise real-time feedback control, leading to up to 70% reduction in trajectory tracking error at high speeds. We verify our method by extensive comparison to a state-of-the-art linear drag model in synthetic and real-world experiments at speeds of up to 14m/s and accelerations beyond 4g.**

Supplementary Material

Video: https://youtu.be/FHvDghUUQtc
Code: https://github.com/uzh-rpg/data_driven_mpc

I. Introduction

Accurate trajectory tracking with quadrotors in high-speed and high-acceleration regimes is still a challenging research problem. While autonomous quadrotors have seen a significant gain in popularity and have been applied in a variety of industries ranging from agriculture to transport, security, infrastructure, entertainment, and search and rescue, they still do not exploit their full maneuverability. The ability to precisely control drones during fast and highly agile maneuvers would allow to not only fly fast in known-free environments, but also close to obstacles, humans, or through openings, where already small deviations from the reference have catastrophic consequences.

Operating a quadrotor at high speeds and controlling it through agile, high-acceleration maneuvers requires to account for complex aerodynamic effects acting on the platform. These effects are difficult to model, since they consist of a combination of propeller lift and drag dependent on the induced airstream velocity, fuselage drag, and complex or even turbulent effects due to the interaction between the propellers, the downwash of other propellers, and the fuselage.

Manuscript received: October 15th, 2020; Revised: December 26th, 2020; Accepted: February 2nd, 2021

This paper was recommended for publication by Editor Clement Gosselin upon evaluation of the Associate Editor and Reviewers' comments.

*These two authors contributed equally.

The authors are with the Robotics and Perception Group, Dep. of Informatics, University of Zurich, and Dep. of Neuroinformatics, University of Zurich and ETH Zurich, Switzerland (http://rpg.ifi.uzh.ch). This work was supported by the National Centre of Competence in Research (NCCR) Robotics through the Swiss National Science Foundation (SNSF) and the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No. 871479 (AERIAL-CORE) and the European Research Council (ERC) under grant agreement No. 864042 (AGILEFLIGHT).

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

Fig. 1: Our quadrotor platform reaches its physical limits at a pitch angle of 80 degrees while performing a lemniscate trajectory in our experiments. Throughout the trajectory, the platform reaches speeds of up to $14\,\mathrm{m\,s^{-1}}$ and accelerations beyond 4g.

Furthermore, in the context of model-based feedback control, the model complexity is constrained by the feedback time-scale and computational capabilities of the executing platform. Therefore, it is not sufficient to find the most accurate model, but required to find an applicable trade-off between model accuracy and complexity.

Very little work exists on agile control of quadrotors at speeds beyond $5\,\mathrm{m\,s^{-1}}$ and accelerations above 2g, [1–8]. Even though these works show agile control at various levels, none of them accounts for aerodynamic effects. This is not a limiting assumption when the quadrotor is controlled close to hover conditions, but introduces significant errors when tracking fast and agile trajectories. Other approaches use iterative learning control to perform highly aggressive trajectories [9], but they are constrained to a single maneuver and do not generalize.

The main challenge when performing aggressive flight is to identify a dynamics model of the platform that is capable of describing the aerodynamic effects while still being lightweight enough to guarantee real time performance. While there exist sophisticated computational fluid dynamics simulations that are able to model turbulent aerodynamic effects [10], they require hours of processing on a compute cluster, and still need to be abstracted in simplified models to be tractable in a control loop running at high frequency.

In this work, we propose to learn the aerodynamic effects acting on the platform from data. Inspired by [11, 12], we use Gaussian Processes to learn the residual dynamics with respect to a simplified quadrotor model that does not account for aerodynamic effects. Learning the residual dynamics simplifies the learning problem and allows describing the model augmentation using only a small number of inducing points for Gaussian Processes. Using such a small model allows leveraging the combined dynamics formulation in a Model Predictive Control (MPC) pipeline.

Our experiments, performed in simulation and in the real world, show that the proposed approach can significantly improve control performance for agile trajectories with speeds up to $14\,\mathrm{m\,s^{-1}}$ and accelerations exceeding 4g. We show that the method generalizes between different trajectories and outperforms methods relying on simplified correction terms.

Contributions

In this paper, we present a Model Predictive Control pipeline that is augmented with learned residual dynamics using Gaussian Processes. We extend the approach presented in [11, 12] to three-dimensional GP predictions for the quadrotor platform. By combining the learned GP corrections with the nominal quadrotor dynamics, we can learn an accurate dynamics model from a small number of inducing data points. Such a small model can be efficiently optimized within an MPC pipeline and allows for control frequencies greater than $100\,\mathrm{Hz}$. We show that the augmented MPC improves trajectory tracking by up to 70% with respect to its nominal counterpart. We verify our method by extensive comparison to a state-of-the-art linear drag model in synthetic and real-world experiments at speeds of up to $14\mathrm{m\,s^{-1}}$ and accelerations beyond 4g.

II. Related Work

Performing fast and agile maneuvers with an autonomous robot requires knowledge of an accurate dynamics model of the platform. However, especially at high speeds and accelerations, such a description of the system is difficult to obtain due to hard-to-model effects caused by friction, aerodynamics or varying battery voltage. As modelling these effects significantly increases the problem complexity, controlling a robot under such conditions requires to find a trade-off between model expressivity and computational tractability. Most prior work on control of autonomous robots does not account for higher-order effects at all and treats them as external disturbances [1, 13–16]. While this allows for very efficient and lightweight controller implementations, tracking performance progressively decreases for higher speeds. In [17], Nonlinear Incremental Dynamics Inversion is used to achieve robust tracking of fast trajectories. However, the reactive nature of this approach does not allow to account for future disturbances as the controller does not optimize over a horizon of actions.

A recent line of work [18–21] investigates the application of dynamics learned entirely from data for a variety of applications such as robot arms, cars or fluids. These learned dynamics representations take the form of deep neural networks and substitute the nominal dynamics in the MPC. While the resulting dynamics models are very expressive, their optimization is often intractable due to local minima. A common way to overcome this challenge is to use sampling-based optimizers, which in turn scale poorly to high-dimensional input spaces.

Instead of learning the full dynamics from data, [11, 12, 22, 23] combine a nominal model with a learned correction term. This allows to limit the learned dynamics to have different dimensionality than the nominal system and provides the possibility to learn only specific effects that are difficult to capture with the nominal model.

For the particular case of quadrotor flight, the most prominent source of disturbances are aerodynamic effects originating from drag caused by the rotors and the fuselage, as well as lift effects that act on the platform at high speeds. By conducting controlled experiments in both wind tunnels as well as instrumented tracking volumes, previous work has shown a significant effect of aerodynamic forces already at linear speeds of $5\,\mathrm{m\,s^{-1}}$ [24, 25].

Neglecting other aerodynamic effects, previous work mainly studies the effect of rotor drag [26, 24, 27]. Rotor drag effects originate from blade flapping and induced drag of the rotors. These effects are typically combined as linear effects in a lumped parameter dynamical model [28]. In [24], the authors identify a linear model for the rotor drag and use it to compute feedforward terms of a PID controller. Even though they show improved trajectory tracking performance, they evaluate their linear model only up to $5\,\mathrm{m\,s^{-1}}$. At these speeds, the linear effect of rotor drag dominates the fuselage drag. We integrate the model of [24] in an MPC pipeline to act as baseline of our approach.

Similar to our work, in [11, 12, 29–31], the authors use Gaussian Processes to improve the control performance of a robotic platform. In [29], Gaussian Processes are used on a quadrotor to correct for wind disturbances. Since instead of platform states only observed disturbances are fed to the GPs, this approach does not learn a dynamics model and can only react to disturbances once they have been observed. In [30], the authors separately learn the translational and rotational dynamics of a quadrotor platform. As this approach learns the full model from data, it requires a large number of training points and is computationally very expensive. As a result, the prediction horizon of the MPC needs to be reduced to a single point to achieve near real-time performance. In [31], a robust experience-driven predictive controller (EPC) is proposed that uses Gaussian belief propagation to account for uncertainties in the state estimate. The controller demonstrates robust constraint satisfaction on a quadrotor platform, where it is integrated into an MPC that controls the translational dynamics of the vehicle. In [11, 12], the authors use the predictions of the Gaussian Processes to improve the tracking performance of an autonomous race car by learning the *residual* dynamics of a nominal model. Learning such residual dynamics instead of the full model allows them to simplify the learning problem and as a result reduce the number of training points in the GP.

Our work is inspired by these approaches, but extends [11, 12] to three-dimensional GP predictions for the quadrotor platform. Instead of learning a mapping from observed disturbances to future disturbance as in [29], we focus on fast flight and correct for aerodynamic effects that arise due to the fast ego-motion of the platform. We tightly integrate the predictions of the Gaussian Processes into the MPC formulation. Instead of using virtual inputs such as bodyrates and collective thrust [1], our MPC models the dynamics down to the motor inputs and can therefore account for the true

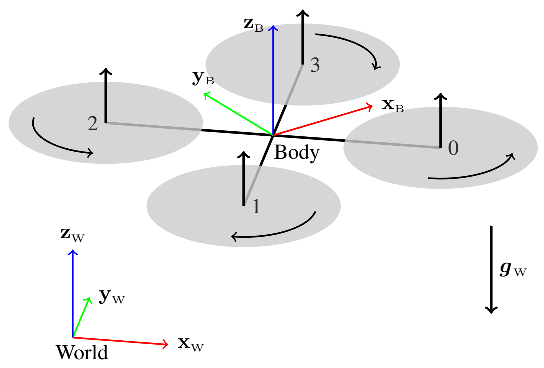

Fig. 2: Diagram of the quadrotor model with the world and body frames and propeller numbering convention.

actuation limits of the platform. Our work is the first to combine Gaussian Processes with a full-state quadrotor MPC formulation to model aerodynamic drag effects while still being able to account for the true actuation limits of the platform.

III. Methodology

A. Notation

We denote scalars in lowercase s, vectors in lowercase bold $\boldsymbol{v}$, and matrices in uppercase bold $\boldsymbol{M}$. We define the World W and Body B frames with orthonormal basis i.e. $\{\boldsymbol{x}_W, \boldsymbol{y}_W, \boldsymbol{z}_W\}$. The frame B is located at the center of mass of the quadrotor. Note that we assume all four rotors are situated in the xy-plane of frame B, as depicted in Fig. 2. A vector from coordinate $\boldsymbol{p}_1$ to $\boldsymbol{p}_2$ expressed in the W frame is written as: ${}_W\boldsymbol{v}_{12}$. If the vector's origin coincide with the frame it is described in, we drop the frame index, e.g. the quadrotor position is denoted as $\boldsymbol{p}_{WB}$. Furthermore, we use unit quaternions $\boldsymbol{q} = (q_w, q_x, q_y, q_z)$ with $\|\boldsymbol{q}\| = 1$ to represent orientations, such as the attitude state of the quadrotor body $\boldsymbol{q}_{WB}$. Finally, full SE3 transformations, such as changing the frame of reference from body to world for a point $\boldsymbol{p}_{B1}$, can be described by ${}_W\boldsymbol{p}_{B1} = {}_W \boldsymbol{t}_{WB} + \boldsymbol{q}_{WB} \odot \boldsymbol{p}_{B1}$. Note the quaternion-vector product denoted by $\odot$ representing a rotation of the vector by the quaternion as in $\boldsymbol{q} \odot \boldsymbol{v} = \boldsymbol{q}\boldsymbol{v}\bar{\boldsymbol{q}}$, where $\bar{\boldsymbol{q}}$ is the quaternion's conjugate.

B. Nominal Quadrotor Dynamics Model

We assume the quadrotor is a 6 degree-of-freedom rigid body of mass m and diagonal moment of inertia matrix $\boldsymbol{J} = \mathrm{diag}(J_x, J_y, J_z)$. Our model is similar to [1, 3] but we write the nominal dynamics $\dot{\boldsymbol{x}}$ up to second order derivatives, leaving the quadrotors individual rotor thrusts $T_i \forall i \in (0, 3)$ as control inputs $\boldsymbol{u}$. The state space is thus 13-dimensional and its dynamics can be written as:

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \dot{\boldsymbol{p}}_{WB} \\ \dot{\boldsymbol{q}}_{WB} \\ \dot{\boldsymbol{v}}_{WB} \\ \dot{\boldsymbol{\omega}}_B \end{bmatrix} = \boldsymbol{f}_{dyn}(\boldsymbol{x}, \boldsymbol{u}) = \begin{bmatrix} \boldsymbol{v}_W \\ \boldsymbol{q}_{WB} \cdot \begin{bmatrix} 0 \\ \boldsymbol{\omega}_B/2 \end{bmatrix} \\ \boldsymbol{q}_{WB} \odot \boldsymbol{T}_B + \boldsymbol{g}_W \\ \boldsymbol{J}^{-1}\left(\boldsymbol{\tau}_B - \boldsymbol{\omega}_B \times \boldsymbol{J}\boldsymbol{\omega}_B\right) \end{bmatrix}, \tag{1}$$

where $\boldsymbol{g}_W = [0, 0, -9.81\,\mathrm{m/s^2}]^\intercal$ denotes Earth's gravity, $\boldsymbol{T}_B$ is the collective thrust and $\boldsymbol{\tau}_B$ is the body torque as in:

$$\boldsymbol{T}_B = \begin{bmatrix} 0 \\ 0 \\ \sum T_i \end{bmatrix} \quad \text{and} \quad \boldsymbol{\tau}_B = \begin{bmatrix} d_y(-T_0 - T_1 + T_2 + T_3) \\ d_x(-T_0 + T_1 + T_2 - T_3) \\ c_\tau(-T_0 + T_1 - T_2 + T_3) \end{bmatrix} \tag{2}$$

where d_x, d_y are the rotor displacements and c_τ is the rotor drag torque constant. To incorporate these dynamics in discrete time algorithms, we use an explicit Runge-Kutta method of 4th order $\boldsymbol{f}_{RK4}(\boldsymbol{x}, \boldsymbol{u})$ to integrate $\dot{\boldsymbol{x}}$ given an initial state $\boldsymbol{x}_k$, input $\boldsymbol{u}_k$ and integration step δt by:

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}_{RK4}(\boldsymbol{x}_k, \boldsymbol{u}_k, \delta t). \tag{3}$$

C. Gaussian Process-Augmented Dynamics

Inspired by [11, 12], we use Gaussian Processes to complement the nominal dynamics of the quadrotor in an MPC pipeline. In this setting, the GPs predict the error of the dynamics and correct them at every time instance t_k. Similar to most GP-based learning problems, we assume the existence of the inaccessible true dynamics $\boldsymbol{f}_{true}$ of the quadrotor, which we measure as $\tilde{\boldsymbol{y}}_{k+1}$ through a noisy process at discrete time instances t_k:

$$\tilde{\boldsymbol{y}}_{k+1} = \boldsymbol{f}_{true}(\boldsymbol{x}_k, \boldsymbol{u}_k) + \boldsymbol{w}_k \tag{4}$$

We further assume that $\boldsymbol{w}_k \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma})$ is Gaussian noise, where $\boldsymbol{\Sigma}$ is the time-invariant and diagonal covariance matrix. This means we can effectively treat each dimension of $\boldsymbol{y}_k$ independently through a separate 1-dimensional output GP. We use the Radial Basis Function (RBF) kernel defined by:

$$\kappa(\boldsymbol{z}_i, \boldsymbol{z}_j) = \sigma_f^2 \exp\left(-\frac{1}{2}(\boldsymbol{z}_i - \boldsymbol{z}_j)^\intercal \boldsymbol{L}^{-2}(\boldsymbol{z}_i - \boldsymbol{z}_j)\right) + \sigma_n^2 \tag{5}$$

where $\boldsymbol{L}$ is the diagonal length scale matrix and σ_f, σ_n represent the data and prior noise variance, respectively, and $\boldsymbol{z}_i$, $\boldsymbol{z}_j$ represent data features.

We redefine the system dynamics as a (corrected, $\boldsymbol{f}_{corr}$) combination of the dynamics (1) plus the mean posterior of a GP, $\boldsymbol{\mu}$. The GP only corrects a subset of the state, determined in the selection matrix $\boldsymbol{B}_d$, using the feature vector $\boldsymbol{z}_k$, determined by selection matrix $\boldsymbol{B}_z$:

$$\boldsymbol{f}_{cor}(\boldsymbol{x}_k, \boldsymbol{u}_k) = \boldsymbol{f}_{dyn}(\boldsymbol{x}_k, \boldsymbol{u}_k) + \boldsymbol{B}_d \boldsymbol{\mu}(\boldsymbol{z}_k) \tag{6}$$

$$\boldsymbol{z}_k = \boldsymbol{B}_z \begin{bmatrix} \boldsymbol{x}_k^\intercal & \boldsymbol{u}_k^\intercal \end{bmatrix}^\intercal. \tag{7}$$

Given the concatenated training feature samples $\boldsymbol{Z}$ and the query feature samples $\boldsymbol{Z}_k$, the mean and covariance of the GP prediction can be recovered as follows:

$$\begin{aligned} &\boldsymbol{\mu}(\boldsymbol{Z}_k) = \boldsymbol{K}_k^\intercal \boldsymbol{K}^{-1} \boldsymbol{Z} && \boldsymbol{\Sigma}_{\mu k} = \boldsymbol{K}_{kk} - \boldsymbol{K}_k^\intercal \boldsymbol{K}^{-1} \boldsymbol{K}_k \\ \text{with} \quad &\boldsymbol{K} = \kappa(\boldsymbol{Z}, \boldsymbol{Z}) + \sigma_n^2 \boldsymbol{I} && \\ &\boldsymbol{K}_k = \kappa(\boldsymbol{Z}, \boldsymbol{Z}_k) && \boldsymbol{K}_{kk} = \kappa(\boldsymbol{Z}_k, \boldsymbol{Z}_k). \end{aligned} \tag{8}$$

where $\boldsymbol{K}_{ij}$, the entry of $\boldsymbol{K}$ with index i, j, is $\boldsymbol{K}_{ij} = \kappa(\boldsymbol{z}_i, \boldsymbol{z}_j)$.

Given the mean and covariance of the GP, not only is it possible to learn the corrected dynamics, but in addition we can also propagate the corrected model forward in time to use it in an MPC. To propagate the state we simply substitute the nominal dynamics $\boldsymbol{f}_{dyn}$ with the corrected model $\boldsymbol{f}_{cor}$ in the Runge-Kutta integration. For the propagation of the covariance, we refer to the formulation in [11, 12].

D. MPC Formulation

In its most general form, MPC stabilizes a system subject to its dynamics $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$ along a reference $\boldsymbol{x}^*(t), \boldsymbol{u}^*(t)$, by minimizing a cost $\mathcal{L}(\boldsymbol{x}, \boldsymbol{u})$ as in:

$$\begin{aligned} &\min_{\boldsymbol{u}} \int \mathcal{L}(\boldsymbol{x}, \boldsymbol{u}) \\ \text{subject to} \quad &\dot{\boldsymbol{x}} = \boldsymbol{f}_{dyn}(\boldsymbol{x}, \boldsymbol{u}) \qquad \boldsymbol{x}(t_0) = \boldsymbol{x}_{init} \\ &\boldsymbol{r}(\boldsymbol{x}, \boldsymbol{u}) = 0 \qquad \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{u}) \leq 0 \end{aligned} \tag{9}$$

where $\boldsymbol{x}_0$ denotes the initial condition and $\boldsymbol{h}$, $\boldsymbol{r}$ can incorporate (in-)equality constraints, such as input limitations.

For our application, and as most commonly done, we specify the cost to be of quadratic form $\mathcal{L}(\boldsymbol{x}, \boldsymbol{u}) = \|\boldsymbol{x} - \boldsymbol{x}^*\|_Q^2 + \|\boldsymbol{u} - \boldsymbol{u}^*\|_R^2$ and discretize the system into N steps over time horizon T of size $dt = T/N$. We account for input limitations by constraining $0 \leq \boldsymbol{u} \leq u_{max}$, and optionally include the GP predictions within the system dynamics.

$$\begin{aligned} &\min_{u} \boldsymbol{x}_N^\intercal Q \boldsymbol{x}_N + \sum_{k=0}^{N} \boldsymbol{x}_k^\intercal Q \boldsymbol{x}_k + \boldsymbol{u}_k^\intercal R \boldsymbol{u}_k \\ \text{subject to} \quad &\boldsymbol{x}_{k+1} = \boldsymbol{f}_{RK4}(\boldsymbol{x}_k, \boldsymbol{u}_k, \delta t) \\ &\boldsymbol{x}_0 = \boldsymbol{x}_{init} \\ &u_{min} \leq \boldsymbol{u}_k \leq u_{max} \end{aligned} \tag{10}$$

where $\boldsymbol{f}_{RK4}$ can be extended to the corrected dynamics $\boldsymbol{f}_{cor}$.

To solve this quadratic optimization problem we construct it using a multiple shooting scheme [32] and solve it through a sequential quadratic program (SQP) executed in a real-time iteration scheme (RTI) [32]. All implementations are done using ACADOS [33] and CasADi [34].

E. Practical Implementation

The implementation of the learned dynamics of our GP-MPC must be designed to maximize the performance while minimizing the computational cost added to the optimization. Note that aerodynamic effects operate on the body reference frame. Likewise, the training dataset is adjusted such that the learning problem setup is to identify such a mapping from body frame velocities ${}_B\boldsymbol{v}$ to body frame acceleration disturbances ${}_B\boldsymbol{a}_e$, so that ${}_B\boldsymbol{a}_e = \boldsymbol{\mu}({}_B\boldsymbol{v})$. Furthermore, to reduce the need for additional training samples, we reduce the dimensionality of our input space such that the mappings are learned axis wise. We can thus write the GP prediction as:

$${}_B\boldsymbol{a}_{ek} = \boldsymbol{\mu}({}_B\boldsymbol{v}_k) = \begin{bmatrix} \mu_{vx}({}_Bv_{xk}) \\ \mu_{vy}({}_Bv_{yk}) \\ \mu_{vz}({}_Bv_{zk}) \end{bmatrix} \tag{11}$$

$$\boldsymbol{\Sigma}_{\mu}({}_B\boldsymbol{v}_k) = \operatorname{diag}\left(\begin{bmatrix} \sigma_{vx}^2({}_Bv_{xk}) \\ \sigma_{vy}^2({}_Bv_{yk}) \\ \sigma_{vz}^2({}_Bv_{zk}) \end{bmatrix} \right) \tag{12}$$

F. Data Collection and Model Learning

To fit the GPs, real-world flight data is collected (as detailed in Sec. IV) using the nominal dynamics model. For each sample at time t_k, the velocity at the next sample point ${}_B\boldsymbol{v}_{k+1}$ and the predicted velocity at the next sample point ${}_B\hat{\boldsymbol{v}}_{k+1}$ are recorded, together with the timestep δt_k. We can then compute the time-normalized velocity error, corresponding to the acceleration error:

$${}_B\boldsymbol{a}_{ek} = \frac{{}_B\boldsymbol{v}_{k+1} - {}_B\hat{\boldsymbol{v}}_{k+1}}{\delta t_k} \tag{13}$$

To select the hyperparameters of the kernel function (l, σ_n, σ_f), we perform maximum likelihood optimization on the collected dataset. Being a non-parametric method, the complexity of GP regression depends on the number of training points. As using the full dataset would make MPC optimization intractable in real time, we subsample the dataset and only use a small number of inducing points. To this end, we leverage the smooth nature of the aerodynamic effects by sampling these points at regular intervals in the ranges of the training set.

IV. Experiments and Results

We design our evaluation procedure to address the following questions: (i) What is the contribution of the learned dynamics of our GP-MPC in a closed-loop tracking task? (ii) How does our GP-MPC compare to an MPC with linear aerodynamic effect compensation, as proposed in [24]? (iii) How does the learned model generalize to unseen trajectories? Finally, we validate our design choices with ablation studies. We refer the reader to the attached video to understand the dynamic nature of our experiments.

A. Experimental Setup

We conduct experiments both in simulation as well as on a real quadrotor platform. To assess our proposed approach, the quadrotor executes three different trajectories (Random, Circle, Lemniscate), illustrated in Fig. 3. The lemniscate trajectory lies in the horizontal plane and is defined by $\left[x(t) = 2\cos\left(\sqrt{2}t\right); y(t) = 2\sin\left(\sqrt{2}t\right)\cos\left(\sqrt{2}t\right)\right]$.

We compare the tracking performance on these trajectories using our MPC with the *Nominal* quadrotor model (1), and the improvement after adding different correction terms identified from data. We study two possible augmentations: *GP-MPC* (ours) and *RDRv*. The RDRv approach was proposed in [24] as a feed-forward PID controller term, which identifies a set of linear drag coefficients along the body axes. We instead incorporate this linear compensation into the nominal dynamics of our MPC pipeline. Note that the three control approaches only differ in the dynamics model used by the MPC, i.e. they use the same control frequency and cost matrices. Furthermore, both the GP-MPC and the RDRv are always trained on the same dataset.

For the simulation experiments, we perform a Nominal run on random polynomial trajectories of high aggressiveness to collect training samples for the GP and the coefficient

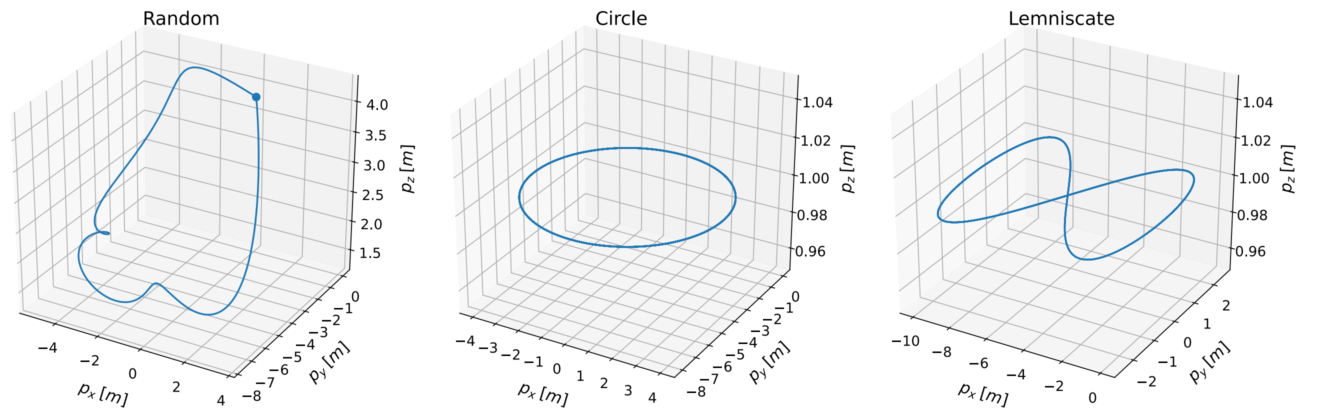

Fig. 3: Trajectories considered in this work. Left: single-loop randomly-generated polynomial trajectory with motion along all axes. The tracking starts and ends at the upper-right corner. Center and right: circular and lemniscate trajectories respectively. Both have zero translation along the z axis, tracking starts at 0 velocity, ramps up until reaching a peak, and ramps down back to hover. The position references remain as shown in the figures in all cases.

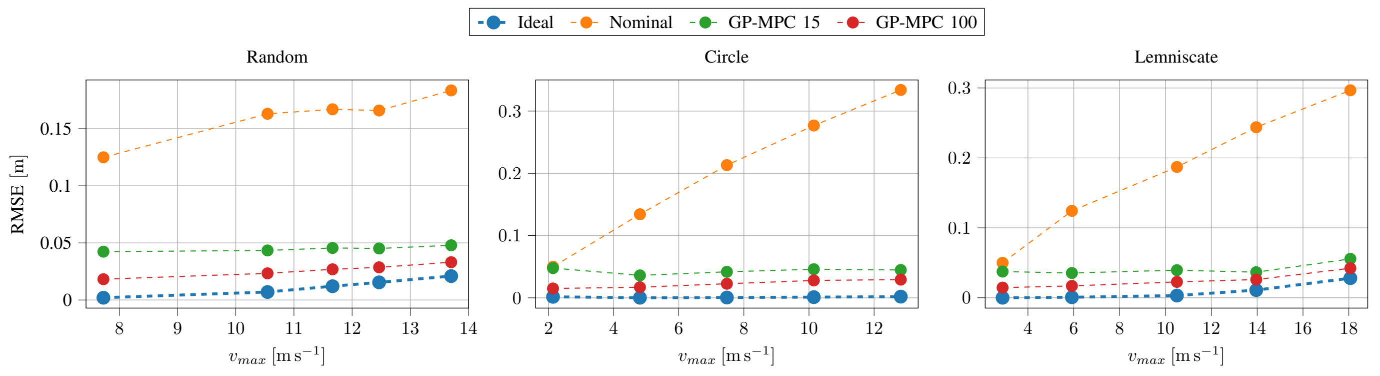

Fig. 4: Closed-loop position tracking error as a function of maximum velocity achieved in our custom simulator. *Ideal* denotes the nominal MPC performance in a disturbance-free scenario. *Nominal* corresponds to the un-augmented MPC, and *GP-MPC 15* and *GP-MPC 100* are our GP-augmented controllers where the GP's have been trained with 15 and 100 training samples.

identification for RDRv. With both models fitted, we deploy all three controllers, Nominal, RDRv and GP-MPC, on the test trajectories without retraining. For the real-world experiments, we first perform a run of the Nominal baseline on both circle and lemniscate trajectories, which is also used for GP training and RDRv coefficient identification. In the subsequent rollouts, we test RDRv and GP-MPC on these two trajectories in different permutations.

In our experimental setup, we only consider obstacle-free scenarios and aim to minimize the tracking error that arises mainly due to aerodynamic disturbance effects. For this reason, the predicted covariance of the GPs is not used in our experiments.

B. Experiments in Simulation

We first evaluate the performance for individual maneuvers in simulation. To isolate the effects of varying MPC computation times for different models, we divide the simulation experiments into two parts: *simplified simulation* and *Gazebo simulation*. This setup allows to compare the predictive performance of arbitrary sized models without the need to correct for varying computation times.

Simplified Simulation This simulation is constituted of a simple forward integration of the system dynamics (1) using an explicit Runge-Kutta method of 4th order with a step size of $0.5\,\mathrm{ms}$. We assume to have access to perfect odometry measurements of the quadrotor, ideal tracking of the commanded single-rotor thrusts, and that the MPC computation is instantaneous. The simulator models drag effects caused both by the rotors as well as the fuselage. Additionally, zero-mean Gaussian noise forces and torques are simulated that act on the quadrotor body, as well as asymmetric noise on the motor voltage signals.

In the simplified simulation, we investigate the influence of the number of training points of the GP on the predictive performance. As the choice of this hyperparameter constitutes a trade-off between model accuracy and computation time, we seek the model with the minimum number of inducing points that surpasses a desired performance threshold. This effect is investigated with two experiments: first, we analyze the trade-off between GP performance and optimization time.

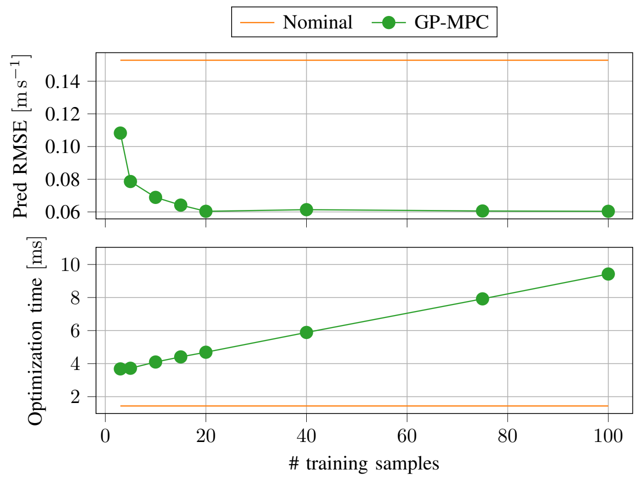

Fig. 5: Trade-off between number of training samples, GP performance (top) and optimization time (bottom). Models are trained and evaluated on data collected in our Simplified Simulation.

In the second experiment, we extend the comparison of different-sized GPs to closed-loop experiments on the three test trajectories.

Having identified the optimal size of the GP, we perform an additional set of experiments to compare the tracking performance on the circle and lemniscate test trajectories between the Nominal and RDRv baselines and our approach. Both trajectories are executed up to varying maximum speeds, where the highest speed pushes the platform to its physical limits. The training set for both the RDRv and the GP models in this simulator is collected by executing random polynomial trajectories of high aggressiveness (such as Fig. 3 left) with the quadrotor, up to $16 \mathrm{m\,s^{-1}}$ axis-wise. This technique works well in simulation since it allows to explore densely all the ranges of operating points without risk of breaking the aircraft if tracking fails.

The results of the GP size analysis are summarized in Fig. 5. As can be seen, the complexity of the optimization problem approximately follows a linear function with respect to the number of training points. Since the predictive performance barely increases when adding more than 20 inducing points, we identify the optimal range to be between 15 to 25 samples, corresponding to $4-5$ ms of optimization time. For comparative purposes, we illustrate in Fig. 4 how two of our GP models perform in closed-loop tracking for 15 and 100 training samples. It can be verified that a larger number of samples is strictly beneficial, but comes at the expense of an increase in optimization time. In fact, such a large model is not usable for a real time application of our pipeline. Based on this evidence, we chose to use 20 inducing points for the rest of this work.

The main results of the closed-loop tracking experiments are summarized in Table I. The table reports position tracking error in millimeters for both maneuvers at varying maximum speeds. While the *Ideal* column indicates the tracking error in case of no unmodelled disturbances (i.e. the MPC dynamics model perfectly fit the actual system), the *Nominal* column represents the baseline when no model augmentations are enabled. Note that even though the MPC controller performs very well in the Ideal scenario, it does not achieve zero tracking error due to discretization effects. Both the RDRv baseline as well as the GP-MPC significantly improve the tracking error compared to the non-augmented Nominal case. However, while RDRv performs comparably to our approach up to speeds of $4\,\mathrm{m\,s^{-1}}$, it starts to fail for higher speeds due to its inability to model higher-order aerodynamic effects such as body drag. Our approach also captures these effects very well and shows consistent improvement for the full range of tested speeds.

Gazebo Simulation To verify the results obtained in the simplified simulation in a well-known quadrotor simulator, we also perform closed-loop tracking experiments in Gazebo [35]. We employ the AscTec Hummingbird quadrotor model using the RotorS extension [36]. To properly evaluate the performance of our pipeline, we also use ground truth odometry measurements instead of a state estimator. We collect a dataset containing velocities in the range $[-12, 12]\,\mathrm{m\,s^{-1}}$ for training our models. This dataset is obtained by tracking randomly generated aggressive trajectories, as with the simplified simulator. We execute the circle and lemniscate trajectories at increasing speeds and compare the tracking performance of the Nominal and RDRv baselines, as well as our approach. Note that once again we use completely independent training and test sets in our setup to ensure our models can generalize to new trajectories.

The main results of the Gazebo experiments are summarized in Fig. 6. In this case, both RDRv as well as our GP-MPC achieve very similar performance over the full range of tested speeds. This is expected, as the RotorS package implementation only simulates rotor drag as aerodynamic effect [36], which follows a linear mapping with respect to the body frame velocity [27]. The true aerodynamic effects acting on a quadrotor however are a combination of rotor drag, body drag and turbulent effects caused by the propellers. We are analyzing these effects in more detail in the following section.

C. Experiments in the Real World

Lastly, we compare the performance of our GP-MPC against both Nominal as well as RDRv controllers on a real quadrotor. We use a custom quadrotor that weighs 0.8kg and has a thrust-to-weight ratio of 5:1. We run the controller on a laptop

TABLE I: Comparison of closed-loop tracking errors on the circle and lemniscate trajectories in simulation.

		Model					
		Ideal	**Nominal**	**RDRv**		**GP-MPC**	
Ref.	$\mathbf{v_{peak}}$ $[\mathrm{m\,s^{-1}}]$	RMSE [mm]	RMSE [mm]	RMSE [mm]	%↓	RMSE [mm]	%↓
Circle	4	0.1	114.1	18.1	84	16.1	**85**
	8	0.4	241.2	56.9	76	25.4	**89**
	12	1.2	338.3	93.0	72	28.4	**93**
Lemn.	4	0.3	104.0	15.5	**85**	16.3	84
	8	1.5	157.7	32.3	79	20.3	**87**
	12	4.2	212.4	60.6	71	24.4	**88**
	Opt. dt [ms]	1.32		1.76		4.13	

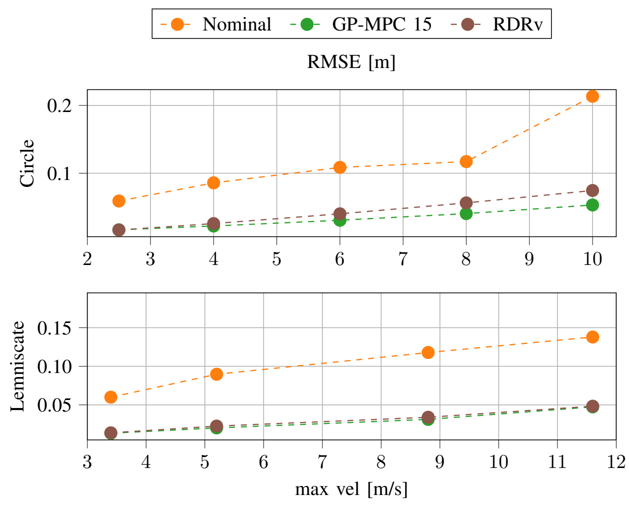

Fig. 6: Closed-loop position tracking error as a function of maximum velocity achieved in the RotorS Gazebo simulator in the circle and the lemniscate trajectories.

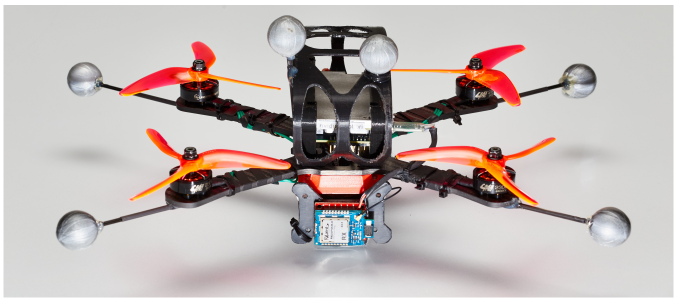

Fig. 7: The quadrotor used for real-world experiments.

computer and send control commands in the form of collective thrust and desired bodyrates at 50 Hz to the quadrotor through a Laird RM024 radio module. A PID controller running on-board the quadrotor tracks the sent commands. The quadrotor flies in an indoor arena equipped with an optical tracking system that provides pose estimates at 100 Hz. Note that our control method also works with state estimates that are obtained differently than with a motion capture system. As in the simulation experiments, we compare the tracking error along both circle and lemniscate trajectories with speeds up to $14\,\mathrm{m\,s^{-1}}$.

To demonstrate that our approach can correct for complex aerodynamic effects, we perform the real world experiments in two settings: in setting (i) we perform all maneuvers with the quadrotor as pictured in Fig. 7, while in setting (ii) we extend the quadrotor body with a vertical drag board. This drag board introduces additional asymmetric aerodynamic disturbance as can be seen in Fig. 8. For the real world experiment, we use 20 inducing points on our GP's.

Table II summarizes the results of our real world experiments in setting (i). We train two GP models on the circle and lemniscate trajectories, and use them at test time in all permutations. As can be seen, our methods as well as the RDRv baseline improve tracking performance by up to 50%, with our approach slightly outperforming the RDRv

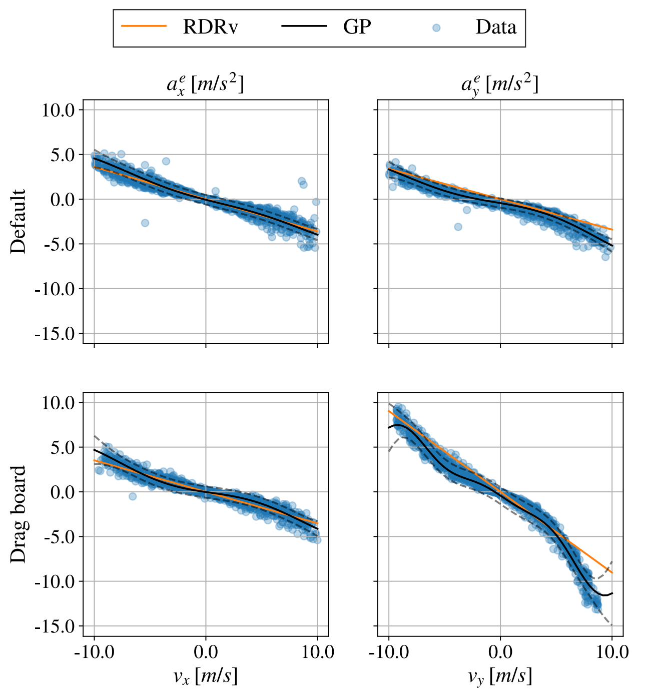

Fig. 8: Aerodynamic effects observed in the real quadrotor platform along body axes x (left column) and y (right column) as a function of body frame velocity. The platform was studied in its default configuration (upper row), and with an additional flat board attached along the body x axis (lower row), resulting in a significantly increased body drag effect in the y direction.

baseline. This result can be explained by the fact that the quadrotor platform used in setting (i) is very compact and powerful, rendering the main source of disturbance being the rotor drag. Rotor drag is a linear effect, which can be well compensated for by the linear RDRv model augmentation.

TABLE II: Comparison of the RDRv and GP-MPC methods in the real world experiments.

	Model RMSE [mm]						
Ref.	**Nomin.**	**GP** (circle)	%↓	**GP** (lemn.)	%↓	**RDRv**	%↓
Circle	319.7	172.9	46	141.0	**56**	168.3	47
Lemn.	396.2	254.2	**36**	266.3	33	269.3	33

TABLE III: Velocity-dependent tracking performance of the augmented MPC methods on the circle trajectory.

		Model				
		Nominal	**RDRv**		**GP-MPC 20**	
Config.	$\mathbf{v}_{\mathbf{range}}$ [m s^{-1}]	RMSE [m]	RMSE [m]	%↓	RMSE [m]	%↓
Default	0-2	0.087	0.130	-49	0.109	-25
	2-4	0.233	0.119	49	0.103	**56**
	4-6	0.329	0.177	46	0.129	**61**
	6-8	0.458	0.210	54	0.154	**66**
	8-10	0.531	0.192	**64**	0.203	62
Drag board	0-2	0.197	0.132	33	0.060	**69**
	2-4	0.346	0.287	17	0.078	**77**
	4-6	0.564	0.381	32	0.141	**75**
	6-8	0.837	0.463	45	0.219	**74**
	8-10	0.912	*Crash*	*?*	0.379	**59**

The slight improvement of GP-MPC over RDRv in this setting can be explained by the ability of the GPs to also account for imperfect thrust mappings.

Finally, we compare the circle tracking for settings (i) and (ii) in Table III. As can be seen, our approach significantly outperforms *RDRv* in setting (ii), where the latter fails to capture the full nonlinearity of aerodynamic effects. This can be verified also in Fig. 8, where the linear fit leads to significant bias.

V. Conclusion

In this work, we propose the usage of Gaussian Processes to augment the nominal dynamics of a quadrotor to compensate for aerodynamic effects. This GP-based model augmentation is integrated in a Model Predictive Controller and the resulting system significantly improves positional tracking error, both in simulation and on a real quadrotor. Using data from previously recorded flights, the GP's are trained to predict the acceleration error of the nominal model given its current velocity in body frame.

In extensive experiments in simulation and the real world, we show that our approach outperforms a state-of-the-art linear drag model. Furthermore, our GP-augmented controller opens up interesting lines of follow-up research for future development. On one hand, we plan to make use of the predicted uncertainty to perform safe agile trajectories close to obstacles. On the other hand, leveraging the fast fitting time of our GP models (in the order of seconds), training and control loop can be executed in parallel on separate threads in real time during flight. This would enable to adapt the dynamics model to varying external or internal conditions such as wind disturbance or battery voltage.

VI. Acknowledgment

The authors thank Leonard Bauersfeld for his help with the photographic illustrations.

References

[1] D. Falanga, P. Foehn, P. Lu, and D. Scaramuzza, "PAMPC: Perception-aware model predictive control for quadrotors," in *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2018.

[2] M. Faessler, F. Fontana, C. Forster, and D. Scaramuzza, "Automatic re-initialization and failure recovery for aggressive flight with a monocular vision-based quadrotor," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2015.

[3] M. Kamel, T. Stastny, K. Alexis, and R. Siegwart, "Model predictive control for trajectory tracking of unmanned aerial vehicles using robot operating system," in *Robot Operating System (ROS)*. Springer, 2017.

[4] M. Neunert, C. de Crousaz, F. Furrer, M. Kamel, F. Farshidian, R. Siegwart, and J. Buchli, "Fast nonlinear model predictive control for unified trajectory optimization and tracking," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2016.

[5] D. Mellinger, N. Michael, and V. Kumar, "Trajectory generation and control for precise aggressive maneuvers with quadrotors," *Int. J. Robot. Research*, 2012.

[6] E. Kaufmann, A. Loquercio, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza, "Deep drone acrobatics," *RSS: Robotics, Science, and Systems*, 2020.

[7] P. Foehn, D. Brescianini, E. Kaufmann, T. Cieslewski, M. Gehrig, M. Muglikar, and D. Scaramuzza, "Alphapilot: Autonomous drone racing," *Robotics: Science and Systems (RSS)*, 2020.

[8] Y. Song and D. Scaramuzza, "Learning high-level policies for model predictive control," in *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2020.

[9] S. Lupashin, A. Schöllig, M. Sherback, and R. D'Andrea, "A simple learning strategy for high-speed quadrocopter multi-flips," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2010.

[10] P. Ventura Diaz and S. Yoon, "High-fidelity computational aerodynamics of multi-rotor unmanned aerial vehicles," in *AIAA Aerospace Sciences Meeting*, 2018.

[11] L. Hewing, J. Kabzan, and M. N. Zeilinger, "Cautious model predictive control using gaussian process regression," *IEEE Trans. Control Sys. Tech.*, 2019.

[12] J. Kabzan, L. Hewing, A. Liniger, and M. N. Zeilinger, "Learning-based model predictive control for autonomous racing," *IEEE Robot. Autom. Lett.*, 2019.

[13] D. Bicego, J. Mazzetto, R. Carli, M. Farina, and A. Franchi, "Nonlinear model predictive control with enhanced actuator model for multi-rotor aerial vehicles with generic designs," *J. Intell. Robot. Syst.*, 2020.

[14] C. Liu, H. Lu, and W.-H. Chen, "An explicit mpc for quadrotor trajectory tracking," in *IEEE Chin. Control Conf. (CCC)*, 2015.

[15] P. Ru and K. Subbarao, "Nonlinear model predictive control for unmanned aerial vehicles," *Aerospace*, 2017.

[16] D. Lunni, A. Santamaria-Navarro, R. Rossi, P. Rocco, L. Bascetta, and J. Andrade-Cetto, "Nonlinear model predictive control for aerial manipulation," in *IEEE Int. Conf. Unm. Aircraft Syst. (ICUAS)*, 2017.

[17] E. Tal and S. Karaman, "Accurate tracking of aggressive quadrotor trajectories using incremental nonlinear dynamic inversion and differential flatness," *IEEE Transactions on Control Systems Technology*, 2020.

[18] G. Williams, N. Wagener, B. Goldfain, P. Drews, J. M. Rehg, B. Boots, and E. A. Theodorou, "Information theoretic MPC for model-based reinforcement learning," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2017.

[19] D. Fan, A. Aghamohammadi, and E. Theodorou, "Deep learning tubes for tube mpc," *Robotics: Science and Systems (RSS)*, 2020.

[20] K. Bieker, S. Peitz, S. L. Brunton, J. N. Kutz, and M. Dellnitz, "Deep model predictive flow control with limited sensor data and online learning," *Theoretical and Computational Fluid Dynamics*, 2020.

[21] I. Lenz, R. A. Knepper, and A. Saxena, "DeepMPC: Learning deep latent features for model predictive control." in *Robotics: Science and Systems (RSS)*, 2015.

[22] M. Saveriano, Y. Yin, P. Falco, and D. Lee, "Data-efficient control policy search using residual dynamics learning," in *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2017.

[23] 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.*, 2019.

[24] M. Faessler, A. Franchi, and D. Scaramuzza, "Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories," *IEEE Robot. Autom. Lett.*, 2017.

[25] S. Sun, C. C. de Visser, and Q. Chu, "Quadrotor gray-box model identification from high-speed flight data," *Journal of Aircraft*, 2019.

[26] P.-J. Bristeau, P. Martin, E. Salaün, and N. Petit, "The role of propeller aerodynamics in the model of a quadrotor uav," in *IEEE Eur. Control Conf. (ECC)*, 2009.

[27] P. Martin and E. Salaün, "The true role of accelerometer feedback in quadrotor control," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2010.

[28] R. Mahony, V. Kumar, and P. Corke, "Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor," *IEEE Robot. Autom. Mag.*, 2012.

[29] M. Mehndiratta and E. Kayacan, "Gaussian process-based learning control of aerial robots for precise visualization of geological outcrops," in *IEEE Eur. Control Conf. (ECC)*, 2020.

[30] G. Cao, E. M.-K. Lai, and F. Alam, "Gaussian process model predictive control of an unmanned quadrotor," *J. Intell. Robot. Syst.*, 2017.

[31] V. R. Desaraju, A. E. Spitzer, C. O'Meadhra, L. Lieu, and N. Michael, "Leveraging experience for robust, adaptive nonlinear mpc on computationally constrained systems with time-varying state uncertainty," *Int. J. Robot. Research*, 2018.

[32] M. Diehl, H. G. Bock, H. Diedam, and P. B. Wieber, "Fast direct multiple shooting algorithms for optimal robot control," in *Fast motions in biomechanics and robotics*. Springer, 2006.

[33] R. Verschueren, G. Frison, D. Kouzoupis, N. van Duijkeren, A. Zanelli, R. Quirynen, and M. Diehl, "Towards a modular software package for embedded optimization," *IFAC-PapersOnLine*, 2018.

[34] 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. Prog. Comp.*, 2019.

[35] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2004.

[36] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, "RotorS–a modular gazebo MAV simulator framework," in *Robot Operating System (ROS)*. Springer, 2016.