Preprint version, final version at http://ieeexplore.ieee.org/ IEEE International Conference on Robotics and Automation 2021

Active Model Learning using Informative Trajectories for Improved Closed-Loop Control on Real Robots

Weixuan Zhang, Marco Tognon, Lionel Ott, Roland Siegwart, and Juan Nieto

***Abstract*— Model-based controllers on real robots require accurate knowledge of the system dynamics to perform optimally. For complex dynamics, first-principles modeling is not sufficiently precise, and data-driven approaches can be leveraged to learn a statistical model from real experiments. However, the efficient and effective data collection for such a data-driven system on real robots is still an open challenge. This paper introduces an optimization problem formulation to find an informative trajectory that allows for efficient data collection and model learning. We present a sampling-based method that computes an approximation of the trajectory that minimizes the prediction uncertainty of the dynamics model. This trajectory is then executed, collecting the data to update the learned model. We experimentally demonstrate the capabilities of our proposed framework when applied to a complex omnidirectional flying vehicle with tiltable rotors. Using our informative trajectories results in models which outperform models obtained from non-informative trajectory by 13.3% with the same amount of training data. Furthermore, we show that the model learned from informative trajectories generalizes better than the one learned from non-informative trajectories, achieving better tracking performance on different tasks.**

I. Introduction

Model-based controllers have shown to be useful in various robotics applications. Especially when accurate models are available, these controllers can exhibit impressive performance [1], [2]. Compared to model-free methods such as reinforcement learning, there is no need of training samples to train a control policy. On the other hand, it can be hard to obtain a good dynamical model for complex systems such as humanoid robots [3], race cars on uneven terrains [4], soft robots [5], and novel fully actuated multi-rotor flying vehicles [6] like the one considered in this work (see Fig. 3).

One approach to solve the modeling problem is to rely on learning techniques: Through interaction with the real-world and data collection a statistical dynamics model is trained, which is either directly fed into a model-based controller, e.g. [4], [7]–[9], or used in simulation to train a control policy [10]. One challenge for these approaches is that often the training data has a different distribution than the test data due to several reasons: first, model uncertainties and feedback controller might lead the system to a state not encountered in a previous data collection routine. Secondly, given partial model knowledge, the region of the data that leads to the best performance is a-priori unknown. Finally, the closed-loop dynamics change as the model used by the controller is updated. One could perform a large number of experiments to cover as much of the input space as possible during training. However, for robotic systems with high-dimensional and continuous state space, the search space typically is too large to be searched exhaustively. Furthermore, the dynamics can change significantly during consecutive experiments, e.g., the crash of a flying vehicle could damage its motors and invalidate the previous training data. Even when considering a specific task, a good model is required in the working area of the state and input spaces, which still might be large. Thus, it is desirable to have an efficient scheme to collect training data locally around the desired task if a precise enough first-principle parametric model is not available or hard to obtain. As these learning techniques are nonparametric, common tools from parametric system identification [11], e.g., persistence of excitation, are not applicable.

One idea is to use the statistical information learned from training data to infer the region where to sample data, thus improving sampling efficiency. This is a well-known approach in machine learning called *active learning* [12]. In this paper we exploit such an idea: we rely on the previously learned statistical model to get an estimate of the region of interest. We then generate an informative trajectory that reduces the overall uncertainty in the estimated region. This trajectory is then executed in the real world to collect data.

More specifically, in a first step, possible informative locations are inferred in simulation from the previously learned model. Then, different informative trajectories are sampled and evaluated according to a cost metric, which is defined as the integral of the predictive uncertainty over these possible locations. The most informative trajectory is then selected and executed on the real robot to collect the data. As a result, the model learned from this informative trajectory should result in improved control performance and a better generalization. The latter is achieved because the informative trajectory reduces the uncertainty over a large region of state and input space.

The contributions of this paper are summarized as follows:

- A formal mathematical formulation of the problem of efficient data collection for learning dynamics model.
- A practical strategy to *efficiently* collect task-relevant data that improves the model-based control performance when used to update the learned model.
- Real experimental results conducted on a complex over-actuated omnidirectional flying system with nonlinear dynamics and 18 actuators. For a figure-8 trajectory, two runs of trajectory flight lead to an angular acceleration tracking error reduction of 54.4%

This work was supported by the NCCR Robotics, NCCR Digital Fabrication and Armasuisse.

Authors are with the Autonomous Systems Lab, ETH Zürich, Leonhardstrasse 21, 8092 Zurich, Switzerland. e-mail: wzhang@mavt.ethz.ch

A. Related work

Active learning in robotics is mostly defined in a regression setting: a regression mapping between an input and an output space is to be learned while the sample complexity is minimized. The exploration of the sample space is typically driven by some metric often consisting in variants of the expected informational gain.

Considering active dynamics model learning, existing work includes the use of information gain on parameter estimates ([13], [14]), Gaussian processes ([15]–[17]), and neural networks [18]. They typically generate trajectories that minimize a defined metric, trading off between exploration and exploitation. Aside from the parameter estimates approach, little work is done on real robots.

We can also distinguish approaches depending whether the trajectory generation is performed online or offline. The online approaches are often done in a receding horizon fashion [19], where trajectories are regenerated at a certain frequency on the fly during experiments. This constant update helps reducing the distance between the desired inputs and achieved ones. However, this approach is computationally intensive. While exploring a state of interest, the robot cannot always stay stationary waiting for a new planned trajectory. Up to date, this method exists only in theoretical works validated in simulation [15], [16], [20].

The offline approach has the shortcoming that the planned trajectory has a larger distance to the executed one, but applicable on real robots. In [18], the trajectory generation is formulated as a variable-constrained problem and validated on a simulated overactuated robotic spacecraft. In [17], the input trajectories are parametrized by consecutive trajectory sections and the most informative and safe trajectory is then executed. Their formulation did not take into account closed-loop control. The method is applied on a high-pressure fluid injection system. Our investigation belongs to this approach: we make use of the previously learned model and simulations to reduce the deviation of the executed trajectory to the desired one. In this work, we demonstrate that this approach works for complex robots and efficiently improve control performance.

II. Modeling and Problem Statement

We consider a generic system whose dynamics in the discrete time domain are described by:

$$\boldsymbol{x}[k+1] = f(\boldsymbol{x}[k], \boldsymbol{u}[k]), \tag{1}$$

where $f(\cdot,\cdot)$ is a Lipschitz-continuous function[1] and represents the *true dynamics*. $\boldsymbol{x}[k] \in \mathcal{X} \subset \mathbb{R}^n$ and $\boldsymbol{u}[k] \in \mathcal{U} \subset \mathbb{R}^m$ describes the state and the control input of the dynamical system at time $k \in \mathbb{N}_{\geq 0}$. To simplify the notation, $\boldsymbol{x}[k]$ denotes $\boldsymbol{x}(kT)$ where $T \in \mathbb{R}_{>0}$ is the sampling time. We remark that a perfect knowledge of $f(\cdot,\cdot)$ is in general not available. We might have only an estimation of it denoted by $\hat{f}(\cdot,\cdot)$.

[1]This is a common assumption that does not limit the validity of the work since most of the considered robotic systems have Lipschitz-continuous dynamics.

The considered task consists in a trajectory tracking problem. A desired *task state trajectory* is defined by the sequence of state values $X_t^r = (\boldsymbol{x}_t^r[0], \ldots, \boldsymbol{x}_t^r[N])$ in the time horizon $N \in \mathbb{N}_{>0}$. Throughout this paper, we use a capitalized letter to indicate a sequence of vectors with a time horizon of N. The subscript $\star_t$ is used to denote the quantities related to the task trajectory tracking problem, while the superscript $\star^r$ denotes reference state or input. We first introduce the following assumption

Assumption 1. *A model-based controller $\pi(\cdot,\cdot,\cdot)$ that is a function of a reference state $\boldsymbol{x}^r[k]$, a state $\boldsymbol{x}[k]$, and an estimated dynamics model $\hat{f}$ is provided*

$$\boldsymbol{u}[k] = \pi(\boldsymbol{x}^r[k], \boldsymbol{x}[k], \hat{f}). \tag{2}$$

Furthermore, if $\boldsymbol{x}[0] = \boldsymbol{x}_t^r[0]$, and $\hat{f}(\boldsymbol{x},\boldsymbol{u}) = f(\boldsymbol{x},\boldsymbol{u})$ for every $(\boldsymbol{x},\boldsymbol{u}) \in \mathcal{Z} = \mathcal{X} \times \mathcal{U}$, then

$$\boldsymbol{x}_t[k+1] = f(\boldsymbol{x}_t[k], \pi(\boldsymbol{x}_t^r[k], \boldsymbol{x}_t[k], \hat{f})) = \boldsymbol{x}_t^r[k+1], \tag{3}$$

for every $k = 0, \ldots, N-1$.

This condition describes a perfect tracking of the desired trajectory given a perfect modeling. We further remark that the given task state trajectory is feasible, so that there exists at least one task input trajectory to achieve it We define the sequence of inputs that provides perfect tracking as $U_t^r = (\boldsymbol{u}_t^r[0], \ldots, \boldsymbol{u}_t^r[N])$, called *task input trajectory*.

Objective 1. *Considering the closed-loop system* (1) *and* (2)*, our objective is to define an active learning method aiming at optimizing the data collection process to*

- *make it more efficient (less experiments and data points),*
- *improve the precision of the learned model,*
- *improve the generalizability of the learned model,*
- *minimize the tracking error.*

We shall show how the learning problem can be reformulated to address such objectives.

Without loss of generality, we can decompose the true dynamics into two components:

$$f(\boldsymbol{x}[k], \boldsymbol{u}[k]) = h(\boldsymbol{x}[k], \boldsymbol{u}[k]) + g(\boldsymbol{x}[k], \boldsymbol{u}[k]), \tag{4}$$

where $h(\cdot,\cdot)$ is called *first principles dynamics*, corresponding to the model reflecting physical laws. We consider $h(\cdot,\cdot)$ to be known. $g(\cdot,\cdot)$ is called *residual dynamics*, corresponds to all other elements not modeled by h. g is assumed unknown and we only have an estimation denoted by $\hat{g}$.

This modeling allows to exploit the knowledge we already have about the system, reducing the learning effort and making it possible to employ several model-based controllers.

Once again, it is clear that, considering the control law (2) with $\hat{f} = h + \hat{g}$, the closed loop system achieves perfect tracking if $\hat{g}(\boldsymbol{z}) = g(\boldsymbol{z})$ for every $\boldsymbol{z} := (\boldsymbol{x},\boldsymbol{u}) \in \mathcal{Z}$. For simplicity we use $\boldsymbol{z}$ to denote the state-input pair $(\boldsymbol{x},\boldsymbol{u})$. We assume that a Bayesian prior model [21] over the residual dynamics is given. That is, for a given test point $\boldsymbol{z}$, the belief of the value of $g(\boldsymbol{z})$ follows a Gaussian probability distribution $\mathcal{N}_{\boldsymbol{z}}\big(\mu(\boldsymbol{z}), \sigma^2(\boldsymbol{z})\big)$. We denote the mean and variance of $\mathcal{N}_{\boldsymbol{z}}(\cdot,\cdot)$ as $\mu(\boldsymbol{z}) \in \mathbb{R}^n$ and $\sigma^2(\boldsymbol{z}) \in \mathbb{R}^{n\times n}_{\geq 0}$,

respectively. Note that the distribution is a function of the test point $\boldsymbol{z}$. We consider the estimation of the residual dynamics as $\hat{g}(\boldsymbol{z}) = \mu(\boldsymbol{z})$, which brings to $\hat{f}(\boldsymbol{z}) = h(\boldsymbol{z}) + \mu(\boldsymbol{z})$.

Let Z denote the state and input trajectory pair (X, U). The prior model can be updated to a posterior model from trajectory data subsampled from Z. In particular, the updated model is described by the posterior mean $\mu(\boldsymbol{z}|Z)$ and posterior variance $\sigma^2(\boldsymbol{z}|Z)$.

We then introduce the following assumption for the Bayesian model:

Assumption 2. *Given two sets of data from trajectory Z_1 and Z_2, for all $\boldsymbol{z} \in \mathcal{Z}$ and $j = 0, \dots, n$, $\sigma_j^2(\boldsymbol{z}|Z_1) < \sigma_j^2(\boldsymbol{z}|Z_2)$ leads to $|\mu_j(\boldsymbol{z}|Z_1) - g_j(\boldsymbol{z})| < |\mu_j(\boldsymbol{z}|Z_2) - g_j(\boldsymbol{z})|$. Furthermore, if $\sigma_j^2(\boldsymbol{z}|Z_1)$ approaches zero, $|\mu_j(\boldsymbol{z}|Z_1) - g_j(\boldsymbol{z})|$ approaches zero. The subscript $*_j$ is used to denote the j-th vector element or j-th diagonal element.*

The intuition behind this practical assumption is that a high-quality observation (which is possible in robotic applications) near the test point reduces the uncertainty at the test point and therefore reduces the estimation error.

To improve the knowledge of $\hat{g}$, suitable data must be collected. A possible solution is to simply run the task trajectory, over and over, until sufficient data is collected to obtain a good model around (X_t^r, U_t^r), or Z_t^r. However, this would require many trials to ensure the collected data is informative enough.

Departing from this basic approach, here we aim to design an algorithm that automatically derives reference state trajectories X_i^r, called *informative state trajectories*. These trajectories aim to efficiently collect data to improve the prior model, thus reducing the task trajectory tracking error when using the control law (2). Let $\boldsymbol{x}_i[k]$ and $\boldsymbol{u}_i[k]$ denote the inputs and states obtained letting the closed-loop system evolve using X_i^r as reference trajectory. In details

$$\begin{aligned} \boldsymbol{x}_i[k+1] &= f(\boldsymbol{x}_i[k], \boldsymbol{u}_i[k]) \\ \boldsymbol{u}_i[k] &= \pi(\boldsymbol{x}_i^r[k], \boldsymbol{x}_i[k], \hat{f}), \end{aligned} \tag{5}$$

with $\boldsymbol{x}_i[0] = \boldsymbol{x}_i^r[0]$. The subscript $*_i$ is used to denote quantities related to the informative trajectory tracking problem.

The following problem is then formulated:

Problem 1. *Find X_i^r as solution of:*

$$\begin{aligned} \min_{X_i^r} \quad & \sum_{k=0}^{N} \|\boldsymbol{x}_t^r[k] - \boldsymbol{x}_t[k]\|_2^2 \\ s.t. \quad & \boldsymbol{x}_t[k+1] = f(\boldsymbol{x}_t[k], \boldsymbol{u}_t[k]) \\ & \boldsymbol{u}_t[k] = \pi(\boldsymbol{x}_t^r[k], \boldsymbol{x}_t[k], \hat{f}_{\text{posterior}}) \\ & \hat{f}_{\text{posterior}} = h + \hat{g} \\ & \hat{g}(\boldsymbol{z}) = \mu(\boldsymbol{z}|Z_i),\ Z_i \text{ as in (5)}. \end{aligned} \tag{6}$$

III. Generation of Informative Trajectories

This section introduces an optimization problem aiming at minimizing an informative cost metric, the solution of which is equivalent to the solution of (6). The problem is solved by practical approximations leading to a sampling-based trajectory generation algorithm.

A. Minimization of the informative cost

Solving (6) is definitely not a trivial problem, even using sampling-based methods. In fact, since we do not know f, solving (6) would require to run two experiments for every sampled informative state trajectory X_i^r, using as reference firstly X_i^r and then X_t^r.

In order to make the problem feasible from a practical point of view, let us recall that using the model-based controller (2), we can achieve perfect tracking by having the perfect knowledge of $\hat{g}$ for all $\boldsymbol{z} \in \mathcal{Z}_t^r$ where

$$\mathcal{Z}_t^r = \{\boldsymbol{z} \in \mathcal{Z} \,|\, \exists\, k \in (0, \dots, N) \text{ s.t. } \boldsymbol{z} = \boldsymbol{z}_t^r[k]\}, \tag{7}$$

contains the pairs state/input that achieve perfect tracking of the task trajectory.

According to Assumption 2, a possible idea is to improve the model by minimizing the uncertainty of the prior model, i.e., $\sigma^2(\boldsymbol{z})$ for all $\boldsymbol{z} \in \mathcal{Z}_t^r$. Thus, we reformulate (6) as

$$\min_{X_i^r} \sum_{\mathcal{Z}_t^r} \sigma^2(\boldsymbol{z}|Z_i). \tag{8}$$

Recall that Z_i are computed as in (5). Note that the solution of (8) allows to minimize the modeling error (Assumption 2) which in turns leads to the minimization of tracking error (Assumption 1). Therefore, the solution of (8) is also the solution of Problem 1.

Notice that we focus on reducing the informative cost on the space relevant to the task instead of the entire state/input space $\mathcal{Z}$. However, from experimental considerations, we remark that improving the model only in $\mathcal{Z}_t^r$ is not enough to achieve good tracking performance. In fact, initial errors, noisy measurements, and external disturbances might make the system deviate from Z_t^r, visiting pairs input/state not included in $\mathcal{Z}_t^r$ for which the model could be imprecise. Therefore, to achieve good tracking also in these non-ideal and more realistic conditions, we propose to improve the learning of the model by solving (8) not only for the points in $\mathcal{Z}_t^r$, but also for the ones that are sufficiently close, i.e., for all $\boldsymbol{z} \in \mathcal{Z}_{\Delta_t^r}$ where

$$\mathcal{Z}_{\Delta_t^r} = \{\boldsymbol{z} \in \mathcal{Z} \,|\, \exists\, \boldsymbol{z}_t^r \in \mathcal{Z}_t^r \text{ s.t. } \|\boldsymbol{z} - \boldsymbol{z}_t^r\| \leq \epsilon\}, \tag{9}$$

with[2] $\epsilon \in \mathbb{R}_{\geq 0}$ being a heuristic that can be tuned to control the exploratory behavior of the informative trajectory. Problem (8) becomes:

$$\min_{X_i^r} \int_{\mathcal{Z}_{\Delta_t^r}} \sigma^2(\boldsymbol{z}|Z_i) d\boldsymbol{z} \tag{10}$$

From now on, we refer to the objective function to be minimized as *informative cost*.

The problem cannot be solved in a closed-form way. Thus, we propose to use a sampling-based optimization method [22] that consists in sampling different informative state trajectories X_i^r and choose the one that shows the smallest informative cost. However, this approach cannot be directly employed due to some practical issues:

[2]With an abuse of notation, we consider $\|\boldsymbol{z} - \boldsymbol{z}_\star\| = \|[\boldsymbol{x}^\top\ \boldsymbol{u}^\top]^\top - [\boldsymbol{x}_\star^\top\ \boldsymbol{u}_\star^\top]^\top\|$. A weighted norm can also be used to normalize the components of state and input vectors.

1) To compute the informative cost for every sampled informative trajectory we should theoretically run an experiment. This is clearly time consuming and does not meet the goals of Objective 1.
2) We do not know U_t^r. From its definition, we should know f to compute U_t^r given X_t^r. Therefore, we cannot directly compute $\mathcal{Z}_{\Delta_t^r}$.
3) It is not straightforward how we can compute the integral of the posterior variance over $\mathcal{Z}_{\Delta_t^r}$.
4) It is not straightforward how we can efficiently sample informative trajectories.

Each of these four problems are individually addressed below proposing a few approximations that make (10) solvable from a practical point of view. This allows for deploying the method on real robots.

B. Approximations of the optimization problem

1) Approximation of the dynamical constraints: During the search for the optimal informative state trajectory, given a candidate informative state trajectory $X_{i,\text{cand}}^r$, instead of computing the posteriori variance based on the data collected from a real experiment, Z_i, we compute it based on the data collected from a simulation of the system, $\bar{Z}_i$. In details, $\bar{Z}_i$ is the output of the simulated closed-loop system using $X_{i,\text{cand}}^r$ as reference trajectory, i.e.,

$$\begin{aligned} \bar{\boldsymbol{x}}_i[k+1] &= h(\bar{\boldsymbol{x}}_i[k], \bar{\boldsymbol{u}}_i[k]) + \boldsymbol{g}'(\bar{\boldsymbol{z}}_i^j[k]) \\ \bar{\boldsymbol{u}}_i[k] &= \pi(\boldsymbol{x}_{i,\text{cand}}^r[k], \bar{\boldsymbol{x}}_i[k], \hat{f}), \end{aligned} \tag{11}$$

where $\boldsymbol{g}'(\bar{\boldsymbol{z}}_i^j[k])$ is a sample of the Bayesian model of the residual dynamics, using the probability distribution $\mathcal{N}_{\bar{\boldsymbol{z}}_i^j[k]}(\cdot,\cdot)$. The bar $\bar{*}$ is used to denote quantities related to the simulation throughout this paper.

2) Approximation of $\mathcal{Z}_{\Delta_t^r}$: Since we do not know f, we cannot compute U_t^r, and therefore neither $\mathcal{Z}_{\Delta_t^r}$. In this section we show how we can get an estimation of $\mathcal{Z}_{\Delta_t^r}$, denoted by $\hat{\mathcal{Z}}_{\Delta_t^r}$, exploiting the current estimation of f.

We firstly uniformly sample the state and input spaces, $\mathcal{X}$ and $\mathcal{U}$, creating the sets $\mathcal{X}' \subset \mathcal{X}$ and $\mathcal{U}' \subset \mathcal{U}$, respectively. We then simulate the closed-loop system M times using X_t^r as reference, We obtain M state and input trajectories $\bar{Z}_t^j$ where

$$\begin{aligned} \bar{\boldsymbol{x}}_t^j[k+1] &= h(\bar{\boldsymbol{x}}_t^j[k], \bar{\boldsymbol{u}}_t^j[k]) + \boldsymbol{g}'(\bar{\boldsymbol{z}}_t^j) \\ \bar{\boldsymbol{u}}_t^j[k] &= \pi(\boldsymbol{x}_t^r[k], \bar{\boldsymbol{x}}_t^j[k], \hat{f}). \end{aligned} \tag{12}$$

Finally, we compute $\hat{\mathcal{Z}}_{\Delta_t^r}$ as

$$\begin{aligned} \hat{\mathcal{Z}}_{\Delta_t^r} = \{ &\boldsymbol{z} \in \mathcal{X}' \times \mathcal{U}' \,|\, \exists\, k \in (0,\dots,N) \text{ and} \\ &j \in (1,\dots,M) \text{ s.t. } \left\| \boldsymbol{z} - \bar{\boldsymbol{z}}_t^j[k] \right\| \leq \epsilon \}. \end{aligned} \tag{13}$$

Similar to (9), the threshold ϵ is a heuristic that controls the exploration of the informative trajectory. With large ϵ, the optimal trajectory should show a more exploratory behavior.

3) Approximation of the informative cost: We replace $\mathcal{Z}_{\Delta_t^r}$ with $\hat{\mathcal{Z}}_{\Delta_t^r}$ in (10) and this integral can be approximately solved using numerical integration such as Monte-Carlo integration: we uniformly sample S pairs $\boldsymbol{z}^j$ in $\mathcal{Z}_{\Delta_t^r}$, where $j = 1,\dots,S$, creating the set $\mathcal{Z}'_{\Delta_t^r}$. We then approximate the informative cost in (10) as

$$\frac{V}{S} \sum_{\boldsymbol{z}^j \in \hat{\mathcal{Z}}'_{\Delta_t^r}} \sigma^2(\boldsymbol{z}^j | Z_i), \tag{14}$$

where V is the volume $\int_{\hat{\mathcal{Z}}_{\Delta_t^r}} d\boldsymbol{z}$. Notice that for the different sampled informative trajectories, V and S remain constant and therefore can be omitted in the optimization.

4) Parametrization of the informative trajectory: Since we want to improve the knowledge of the model in $\mathcal{Z}_{\Delta_t^r}$, it is natural to think that the informative state trajectory X_i^r should be "close" to the task state trajectory X_t^r. Therefore, given a generic k, we define $\boldsymbol{x}_i^r[k]$ such that

$$\boldsymbol{x}_i^r[k] = \boldsymbol{x}_t^r[k] + \delta\boldsymbol{x}[k]. \tag{15}$$

Now, sampling informative state trajectories means sampling "deviations" from the task state trajectory. To reduce the sampling space, which has the same dimension of $\mathcal{X}$, we parametrize $\delta\boldsymbol{x}$ using the Discrete Fourier Transform (DFT)

$$\delta\boldsymbol{x}[k] = \frac{1}{P} \sum_{p=0}^{P-1} \boldsymbol{\Theta}_{\boldsymbol{x}}^\top \boldsymbol{e}_p e^{j\frac{2\pi p}{P}k}, \tag{16}$$

where $P \in \mathbb{N}_{>0}$, j is the complex operator, $\boldsymbol{e}_p \in \mathbb{R}^P$ is a vector with 1 in place p and 0 elsewhere, and $\boldsymbol{\Theta}_{\boldsymbol{x}} \in \mathcal{O}_{\boldsymbol{x}} \subset \mathbb{R}^{n \times P}$ is the state parameter matrix.

From sampling every state of the informative trajectory, we now samples only fewer parameters. Furthermore, the rationale behind the use of DFT parametrization is that it gives us a more intuitive control of the frequencies of excitation. We can use fewer parameters to generate excitation signals that are spread through frequencies of interest. Intuitively, the deviation signal can be seen as an excitation signal added around the task state trajectory. As a result, the algorithm inherently explores locally around the task trajectory.

C. Sampling-based optimization algorithm

Considering the previous simplifications, (10) becomes

$$\begin{aligned} \min_{\boldsymbol{\Theta}_{\boldsymbol{x}}} \quad & \sum_{\boldsymbol{z}^j \in \hat{\mathcal{Z}}'_{\Delta_t^r}} \sigma^2(\boldsymbol{z}^j | \bar{Z}_i) \\ \text{s.t.} \quad & \hat{\mathcal{Z}}_{\Delta_t^r} \text{ as in (13)}, \ \boldsymbol{z}^j \text{ as in (12)}, \\ & \boldsymbol{x}_i^r[k] = \boldsymbol{x}_t^r[k] + \delta\boldsymbol{x}[k], \ \delta\boldsymbol{x}[k] \text{ as in (16)}. \end{aligned} \tag{17}$$

Practically, to solve (17) we used a Monte-Carlo sampling-based method. The algorithm follows the next steps which require the simulation of the system only:

1) Uniformly sample a set of parameters $\boldsymbol{\Theta}_{\boldsymbol{x}} \in \mathcal{O}_{\boldsymbol{x}}$ and compute several informative state trajectories as in (15) and (16);
2) Simulate multiple times the system with the sampled residual model $\boldsymbol{g}'$ according to the prior model. Each informative state trajectories computed at step 1 is used as reference;
3) For every simulation, collect the data relative to the performed trajectory, $\bar{Z}_i$, and update the Bayesian model of the residual dynamics;

4) Compute $\hat{\mathcal{Z}}_{\Delta_t^r}$ as explained in III-B.2;
5) Evaluate the information cost in $\hat{\mathcal{Z}}_{\Delta_t^r}$ according to (14) associated to every new updated model;
6) Select the informative state trajectory corresponding to the minimum information cost.

Once the informative state trajectory supposed to provide the best model update is selected, it is used as reference in a real experiment. The relative collected data, Z_i, is then employed to update the prior model. The full process can be repeated from step 1), to find a new state informative trajectory that would allow to further improve the model accuracy, and in turn to reduce the tracking error.

Remark: Note that the quality of the approximate solution depends on the quality of the prior model. Therein lies the purpose of this algorithm: Within each iteration, the quality of the prior model improves, and the solution to the approximated problem converges towards the true optimum. Consequently, this helps improving the prior model.

IV. Application to an Aerial Robot: The OMAV

This section shows how the above framework is applied on an omnidirectional flying vehicle, called *omav* [6]. The omav (Fig. 3) is an overactuated omnidirectional flying vehicle with six tiltable arms in a hexagonal arrangement. A coaxial rotor configuration is rigidly attached to the end of each arm. The rotation of each arm can be actively controlled by a servo motor, which results in a total of 18 actuators. Although the setup enhances the motion and interaction capabilities, aerodynamic disturbances among the rotors, unknown servo dynamics, backlashes, and other mechanical inaccuracy are difficult to be modeled and included in standard model-based controller. This makes omav a suitable testbed to validate the proposed method for active model learning.

The state of the omav is given by $\boldsymbol{x} = [\boldsymbol{p}^\top \ \boldsymbol{\eta}^\top \ \dot{\boldsymbol{p}}^\top \ \boldsymbol{\omega}^\top]^\top \in \mathcal{X} \subset \mathbb{R}^{12}$. In order, $\boldsymbol{x}$ includes the position, attitude (expressed in Euler angles), linear velocity, and angular velocity of the vehicle. As input of the system we consider the commanded wrench, i.e., the total force and moment commanded to the vehicle[3], $\boldsymbol{u} = [\boldsymbol{f}_{\text{cmd}}^\top \ \boldsymbol{\tau}_{\text{cmd}}^\top]^\top \in \mathcal{U} \subset \mathbb{R}^6$. We assume that an allocation policy is implemented to transform $\boldsymbol{u}$ into low level commands for the servos and the motors [6]. Finally, the dynamics of the omav can be written as in (4), where h is derived using standard Newton-Euler equations. Notice that h is linear with respect to the input and can be written as

$$h(\boldsymbol{z}) = l(\boldsymbol{x}) + \boldsymbol{u}, \tag{18}$$

where $l(\boldsymbol{x})$ includes all the terms that do not depend on $\boldsymbol{u}$.

On the other hand, g includes all previously mentioned unmodeled dynamic behaviors that cannot be easily captured with first principles. Considering the last six row of the dynamics (the linear and angular accelerations), we can consider $g(\boldsymbol{z})$ as the mismatch between the commanded wrench and the actuated one.

The controller tries to implement a feedback linearization control law with a PID action on the position and attitude errors. In particular, given a reference task trajectory, X_t^r, and a priori model for g, the controller $\pi(\boldsymbol{x}_t^r[k], \boldsymbol{x}[k], \hat{f})$ tries to find the input $\boldsymbol{u}[k]$ that solves the following optimization problem

$$\min_{\boldsymbol{u}[k]} \left\| \boldsymbol{x}^\star[k] - l(\boldsymbol{x}[k]) - \boldsymbol{u}[k] - \hat{g}(\boldsymbol{u}[k]) \right\|, \tag{19}$$

where $\boldsymbol{x}^\star[k] = \boldsymbol{K}(\boldsymbol{x}_t^r[k] - \boldsymbol{x}[k]) + \boldsymbol{K}_I \sum_{j=0}^{k} (\boldsymbol{x}_t^r[j] - \boldsymbol{x}[j])$ is the PID action, with $\boldsymbol{K}, \boldsymbol{K}_I \in \mathbb{R}^{12\times 12}$ positive definite matrices. For the details about the implementation of such an optimization, we refer the interested reader to [8].

From experimental observations, we remark that the residual dynamics regarding the differential kinematics and linear acceleration (first nine rows) is almost negligible with respect to the one regarding the angular acceleration (last three rows). In other words, the mismatch between commanded and actual force is much smaller than the one between commanded and actual torque. For this reason, in this first work, we focus on the attitude dynamics, applying the proposed active dynamics learning only on the last three rows of the system dynamics. These mismatches are modeled as three independent single-output Gaussian processes with $\boldsymbol{u}$ as the training input and the torque model mismatch as training output. We neglect the rotational drag torque acting on the vehicle since the vehicle is mostly operating with low angular velocities, thus $\hat{g}$ is modeled independent of the state.

V. Experimental Results

The experimental platform is the omnidirectional aerial vehicle in Fig. 3: the omav. The omav weighs $4\,\text{kg}$ and is equipped with a NUC i7 computer and a PixHawk flight controller. This configuration allows to run all the necessary algorithms onboard implemented in a ROS framework. A motion capture system provides pose estimates at $100\,\text{Hz}$. For a more complete description of the testbed see [6].

As stated in Section IV, the proposed method has been implemented and evaluated focusing on the rotational dynamics. For the learned Gaussian process model, data points are subsampled from the experimental data using the k-medoids [23] algorithm where the Euclidean squared distance between the inputs is used as the distance metric. Throughout the experiments, squared exponential kernels are used. The deviation $\delta\boldsymbol{x}[k]$ is sampled around x, y, z-axis on the angular acceleration level, constraining to be below 2 Hz. Note that this is equivalent to giving $\delta\boldsymbol{x}[k]$ on the angular velocity. For simplicity, we limit the number of frequencies P to 2 and allow the frequency locations to be sampled along with its magnitude. This yields a total of 12 coefficients to be sampled. The simulation framework is set up using RotorS Gazebo simulator [24]. In this section we use "non-informative trajectory" to describe the case where only the task trajectory is used to collect the data to update the model.

A. Correlation between informative cost and tracking error

An experiment is conducted to investigate whether the tracking error defined in problem (6) is correlated to the informative cost defined in (17). The omav is asked to follow a pitching trajectory up to 60 degrees in pitch and 1 rad/s^2 in pitch angular acceleration, similar to previous work [8]. A

[3]For simplicity, we consider force and moment scaled by mass and inertia, respectively.

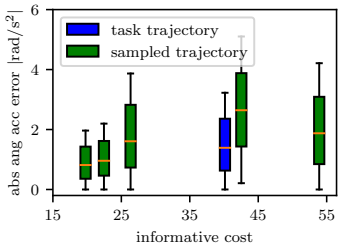

Fig. 1: A comparison of the tracking performance using the model learned from sampled trajectories and task trajectory.

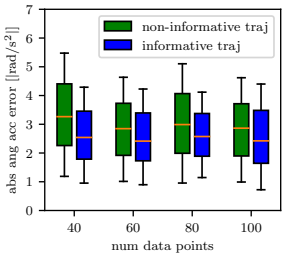

Fig. 2: A comparison of the tracking performance between informative trajectory and task trajectory for the same number of data points.

Fig. 3: The omnidirectional flying vehicle (omav) used to experimentally validate our method.

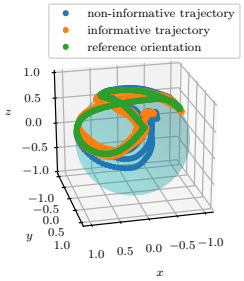

Fig. 4: Tracking of a body-fixed unit vector $(1,1,1)/\sqrt{3}$ is plotted on a unit sphere.

prior model is built by collecting the data from executing the task trajectory. Next, five sampled candidate informative trajectories and the task trajectory are executed and six learned models are built accordingly. They are then evaluated on the test data generated by the prior model. ϵ is heuristically tuned by simulation computing the average distance between the commanded wrench and the achieved wrench. The tracking performance of the angular acceleration[4] along the y-axis using these models are shown in Fig. 1. It can be seen that there is a clear correspondence between the informative cost and the tracking error. Furthermore, the model learned from the task trajectory does not yield the best tracking performance.

B. Comparison between informative and task trajectory

To compare the efficiency of the informative and non-informative trajectory, a figure-8 in attitude (with roll and pitch up to 26 degrees) with constant position is given as a task trajectory (see Fig. 4). We compute the prior model running the task trajectory for the first time. Then 20 trajectories are randomly generated and evaluated in simulation as explained in Section III-C using the prior model. The most informative trajectory (lowest informative cost) and the task trajectory are then executed and the data are recorded for both trajectories. We subsampled 20, 40, 60, 80 data points from the experiments running each trajectory and built a model for each of these combinations by augmenting the prior model with these data points. The hyperparameters of the Gaussian processes are reoptimized. The models are then used in the controller to track the task trajectory in real experiments for validation. Tracking performance are evaluated in Fig. 2 as the average of the absolute angular acceleration over all three axes. It can be noted that for the same amount of data points, the informative trajectory always outperforms the non-informative trajectory in term of both mean tracking error and corresponding variance. On average the performance[5] of informative trajectories outperforms the non-informative one by 13.3%.

[4]Notice that evaluating the angular acceleration tracking is equivalent to evaluate the error between actual and commanded torque which strongly depends on the model accuracy.

[5]By performance of a trajectory we mean the tracking performance using the updated controller with the data collected from that trajectory.

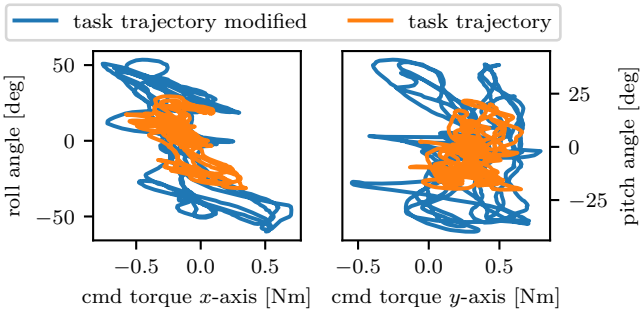

Fig. 5: Phase plots of the task trajectory and modified task trajectory. It can be observed that although the modified trajectory extend beyond the task trajectory, the model learned from the informative trajectory helps to reduce the tracking.

	x-axis	y-axis	z-axis
non-informative	38.4%	41.7%	23%
informative	43.2%	57.9 %	62%

TABLE I: Angular acceleration tracking error reduction with respect to the case without model learning in percentage.

C. Comparison of the model generalizability

To test the generalizability of the model learned from the informative trajectory, a modified figure-8 trajectory with higher pitch and roll reference angles (up to 43 degrees) is used. As can be seen in the phase plot in Fig. 5, the state input pairs of the modified figure-8 extend up to twice of the original one. In this case, both models from the informative trajectory and the non-informative trajectory have 100 data points. It can be seen from Table I that the model learned from informative trajectory yields better tracking performance, especially around the z-axis.

VI. Conclusion

This work presents a practical framework that effectively and efficiently collects data points for the learning of models used at the control level to significantly improve tracking performance on real robots. We experimentally demonstrate the validity of the method on an overactuated aerial robot, the omav, whose dynamics is complex and difficult to learn. Experimental results show that the learned model from informative trajectories is efficient in data points collection and generalizes on modified trajectories.

REFERENCES

[1] P. Abbeel, A. Coates, and A. Y. Ng, "Autonomous helicopter aerobatics through apprenticeship learning," *The International Journal of Robotics Research*, vol. 29, no. 13, pp. 1608–1639, 2010.

[2] 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, pp. 3–39.

[3] S. Kuindersma, R. Deits, M. Fallon, A. Valenzuela, H. Dai, F. Permenter, T. Koolen, P. Marion, and R. Tedrake, "Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot," *Autonomous robots*, vol. 40, no. 3, pp. 429–455, 2016.

[4] C. J. Ostafew, A. P. Schoellig, and T. D. Barfoot, "Learning-based nonlinear model predictive control to improve vision-based mobile robot path-tracking in challenging outdoor environments," in *IEEE International Conference on Robotics and Automation*, 2014.

[5] M. T. Gillespie, C. M. Best, E. C. Townsend, D. Wingate, and M. D. Killpack, "Learning nonlinear dynamic models of soft robots for model predictive control with neural networks," in *2018 IEEE International Conference on Soft Robotics (RoboSoft)*. IEEE, 2018, pp. 39–45.

[6] K. Bodie, M. Brunner, M. Pantic, S. Walser, P. Pfndler, U. Angst, R. Siegwart, and J. Nieto, "An Omnidirectional Aerial Manipulation Platform for Contact-Based Inspection," *Robotics: Science and Systems XV*, 2019.

[7] J. Kabzan, L. Hewing, A. Liniger, and M. N. Zeilinger, "Learning-based model predictive control for autonomous racing," *IEEE Robotics and Automation Letters*, 2019.

[8] W. Zhang, M. Brunner, L. Ott, M. Kamel, R. Siegwart, and J. Nieto, "Learning dynamics for improving control of overactuated flying systems," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5283–5290, 2020.

[9] D. Nguyen-Tuong and J. Peters, "Using model knowledge for learning inverse dynamics," in *IEEE International Conference on Robotics and Automation*, 2010.

[10] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, 2019.

[11] L. Ljung, *System identification, Theory for the user*. Prentice Hall, 1999.

[12] B. Settles, "Active learning literature survey," University of Wisconsin-Madison Department of Computer Sciences, Tech. Rep., 2009.

[13] P. Schrangl, P. Tkachenko, and L. del Re, "Iterative model identification of nonlinear systems of unknown structure: Systematic data-based modeling utilizing design of experiments," *IEEE Control Systems Magazine*, vol. 40, no. 3, pp. 26–48, 2020.

[14] A. D. Wilson, J. A. Schultz, A. R. Ansari, and T. D. Murphey, "Dynamic task execution using active parameter identification with the baxter research robot," *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 1, pp. 391–397, 2016.

[15] T. Koller, F. Berkenkamp, M. Turchetta, and A. Krause, "Learning-based model predictive control for safe exploration," in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 6059–6066.

[16] M. Buisson-Fenet, F. Solowjow, and S. Trimpe, "Actively learning gaussian process dynamics," *arXiv preprint arXiv:1911.09946*, 2019.

[17] C. Zimmer, M. Meister, and D. Nguyen-Tuong, "Safe active learning for time-series modeling with gaussian processes," in *Advances in Neural Information Processing Systems*, 2018, pp. 2730–2739.

[18] Y. K. Nakka, A. Liu, G. Shi, A. Anandkumar, Y. Yue, and S.-J. Chung, "Chance-constrained trajectory optimization for safe exploration and learning of nonlinear systems," *arXiv preprint arXiv:2005.04374*, 2020.

[19] F. Borrelli, A. Bemporad, and M. Morari, *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[20] A. Capone, G. Noske, J. Umlauft, T. Beckers, A. Lederer, and S. Hirche, "Localized active learning of gaussian process state space models," in *Learning for Dynamics and Control*. PMLR, 2020, pp. 490–499.

[21] P. Congdon, *Bayesian statistical modelling*. John Wiley & Sons, 2007, vol. 704.

[22] T. Homem-de Mello and G. Bayraksan, "Monte carlo sampling-based methods for stochastic optimization," *Surveys in Operations Research and Management Science*, vol. 19, no. 1, pp. 56–85, 2014.

[23] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 14. Oakland, CA, USA, 1967, pp. 281–297.

[24] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, *Robot Operating System (ROS): The Complete Reference (Volume 1)*. Cham: Springer International Publishing, 2016, ch. RotorS—A Modular Gazebo MAV Simulator Framework, pp. 595–625. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-26054-9_23