Active Learning of Discrete-Time Dynamics for Uncertainty-Aware Model Predictive Control

Alessandro Saviolo[1], Jonathan Frey[2], Abhishek Rathod[1], Moritz Diehl[2], and Giuseppe Loianno[1]
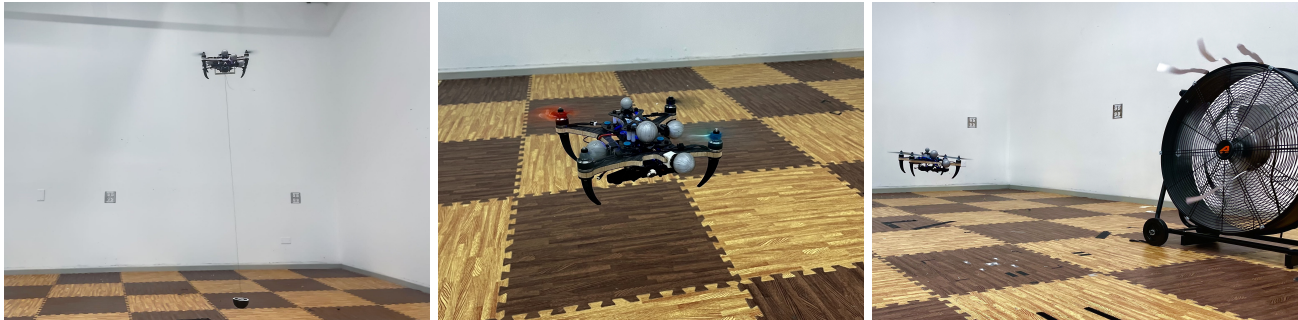
Figure 1: By combining online learning with uncertainty-aware model predictive control, the learned dynamics actively adapt to multiple challenging operating conditions, enabling unprecedented flight control.

Abstract—Model-based control requires an accurate model of the system dynamics for precisely and safely controlling the robot in complex and dynamic environments. Moreover, in presence of variations in the operating conditions, the model should be continuously refined to compensate for dynamics changes. In this paper, we present a self-supervised learning approach that actively models the dynamics of nonlinear robotic systems. We combine *offline* learning from past experience and *online* learning from current robot interaction with the unknown environment. These two ingredients enable a highly sample-efficient and adaptive learning process, capable of accurately inferring model dynamics in real-time even in operating regimes that greatly differ from the training distribution. Moreover, we design an uncertainty-aware model predictive controller that is heuristically conditioned to the aleatoric (data) uncertainty of the learned dynamics. This controller *actively* chooses the optimal control actions that (i) optimize the control performance and (ii) improve the efficiency of online learning sample collection. We demonstrate the effectiveness of our method through a series of challenging real-world experiments using a quadrotor system. Our approach showcases high resilience and generalization capabilities by consistently adapting to unseen flight conditions, while it significantly outperforms classical and adaptive control baselines. Video: https://youtu.be/QmEhSTcWob4

Index Terms—Model Learning for Control; Aerial Systems: Mechanics and Control; Learning and Adaptive Systems; Optimization and Optimal Control

I. INTRODUCTION

MODEL-BASED control is an attractive framework for robot control thanks to its adaptivity, scalability, and data efficiency [1]–[4]. Despite these advantages, the effectiveness of this framework heavily relies on how accurately it can model the robot's interaction with its environment. For example, developing an accurate model for an aerial

[1]The authors are with Tandon School of Engineering, New York University, New York, USA. e-mail: {as16054,arathod,loiannog}@nyu.edu.
[2]The authors are with University of Freiburg, Freiburg, Germany. e-mail: {jonathan.frey,moritz.diehl}@imtek.uni-freiburg.de.

vehicle requires capturing complex factors like wind forces, propeller rotation, aerodynamic effects, and vibrations [5]. Additionally, it is crucial to continually adjust this model to match changes in operating conditions. For example, the platform may be extended with cable-suspended payloads that would significantly change the dynamics by varying mass and moment of inertia (Figure 1). Failing to adapt could result in diminished control performance and even catastrophic failures.

Traditional modeling of the system dynamics is performed using physics-based principles [6]–[8]. While these methods can precisely identify rigid-body systems, they fail to capture complex non-linear phenomena, such as friction, deformation, and aerodynamic effects, that cannot be directly measured and lack explicit analytic equations. To address this, recent studies have explored data-driven approaches for modeling robot dynamics using different learning paradigms, from offline [9]–[18] to online [19]–[22] and active learning [23]–[25].

Offline learning is a setting where the dynamics model is trained on a set of previously collected demonstration data. This learning approach ensures both training over the entire data (i.e., capturing a global dynamical model of the system) and learning without posing any physical risk to the robot. However, offline learning assumes that the global model will remain accurate over time and thus cannot handle varying operating conditions characteristic of most real-world environments. Online learning extends offline learning by continuously updating the model during operation with newly collected data, thereby relaxing the need for a perfect system prior. However, this learning strategy treats the learned model as a passive recipient of data to be processed, neglecting the system's ability to act and influence the operating environment for effective data gathering. Active (online) learning corrects this by enabling the robot to take actions that aid its learning process more effectively [26], [27]. The action selection is performed to maximize a performance gain, such as the maximum entropy [28]. When the actions are properly selected, the model achieves higher sample efficiency.

In this work, we propose a self-supervised learning frame-

work for actively learning the dynamics of nonlinear robotic systems. Our approach employs a Model Predictive Control (MPC) strategy that harnesses data uncertainty in the learned model to concurrently and efficiently optimize its control performance and learning capability. The specific form of our model, the way we incorporate and combine aleatoric uncertainty, the adaptivity and sample efficiency of our controller in the active learning process as well as the ability to transition this class of approaches from theory to practice and specifically tailored to a resource-constrained, real-time environment such as a small quadrotor are unique aspects of our work compared to existing state-of-the-art solutions. Overall, this work presents the following key contributions

- Combining *offline* learning from past experience and *online* learning from current robot-environment interaction while tackling the challenges related to learning stability and robustness inherent in gradient-based online adaptation. This combination demonstrates exceptional adaptation and generalization on embedded systems across varied, challenging environments, significantly enhancing the closed-loop control of embedded robotic platforms.
- Proposing a novel method based on the Unscented Transform (UT) to estimate the aleatoric (data) uncertainty of the learned model, addressing the challenges associated with orientation parametrization from non-Euclidean space of our systems, and capitalizing on its sample and memory efficiency.
- Designing an MPC, heuristically conditioned by this uncertainty, which actively selects actions to maximize its performance and resilience by efficiently adapting its learned robot model.
- Experimentally validating the proposed methodology to a quadrotor's system, demonstrating successful adaptation of the learned dynamics in multiple challenging real-world experiments. The dynamical model adapts to substantial system configuration modifications (e.g., cable-suspended payload, propeller mixing) and stochastic external effects (e.g., wind disturbance) unseen during training, enabling unprecedented flight control.

II. RELATED WORKS

The proposed learning approach combines offline and active online learning. In the following, we discuss these techniques to accurately model and adapt robot dynamics. Additionally, we introduce different approaches for modeling uncertainty.

Offline Learning. Neural Networks (NNs) have been successfully employed for learning robot dynamics due to their favorable expressive power and flexibility [9]–[14] and the simple self-supervised data collection procedure [29], [30]. Similar to previous works, we seek to learn a NN-based model of the system dynamics purely from robot experience. Training the model offline has several advantages compared to pure online learning, such as training without posing any physical risk to the robot and the capability to learn from multiple demonstrations which allows capturing a global model of the system (i.e., the full dynamical evolution of the system). Most of the NN-based approaches, including those

related to quadrotor systems, have focused on learning the continuous-time dynamics in a self-supervised manner [9], [11], [13]. However, recovering the ground truth translational and rotational accelerations for training the continuous-time NN-based models is extremely complex due to the highly non-linear sensor noise. Hence, building accurate datasets for training NN-based models for continuous-time dynamics is extremely difficult and error-prone. Conversely, in this work, we propose to directly learn the discrete-time dynamics, bypassing the differentiation step required to recover the unobserved accelerations, similarly to [19]. Consequently, the learned dynamical model is trained over less noisy data and can better generalize to unseen data, as shown in the presented experimental results.

Another popular learning scheme for modeling dynamics offline is meta-learning [31]–[36], which explicitly learns a distribution over different "tasks" (e.g., dynamics associated with different operating regimes). Meta-learning approaches have demonstrated impressive sample efficiency on several tasks, including those considered in this work such as payload transportation [37] and wind disturbance adaptation [21]. However, efficiency is largely obtained by integrating a strong inductive bias over the testing task and often requires huge amounts of data for effective training. Contrarily, the proposed approach is completely task-agnostic, learns offline only a coarse dynamics model from a basic operating regime - alleviating the need for complex hand-designed data sets, and achieves high sample efficiency by actively selecting actions to safely exploit the dynamics model during operation.

Online Learning. In presence of variations in the operating conditions, the dynamics change, and thus the global model learned offline is no longer accurate. Online learning is a promising solution for this issue and consists in continuously updating the model during operation as new data is seen [19]. The optimization can be performed over the whole NN model or only the last layers. Generally, trading-off between these two approaches depends on the similarity of the online and offline training data distributions as well as the available computational budget. When modeling system dynamics, the two distributions are hardly completely dissimilar but share common representations mostly in the deepest dimensions. Moreover, continuously refining the whole model would make it less explainable and thus may yield unexpected predictions. Consequently, in this work, we employ the proposed online learning procedure to only adapt the last layer's weights.

The proposed online optimization procedure is inspired by [38] and [19]. [38] couples model and controller learning by devising a forward loss similar to our formulation. The proposed iterative method allows them to learn an effective control policy on a manipulator and a quadruped in a few iterations. However, the approach is validated using basic system configurations in simulation, hence neglecting all the complex effects that make dynamics modeling a challenging task. Furthermore, such an iterative method would not be feasible in a real-world setting, particularly for an aerial vehicle such as the quadrotor, due to the extensive time and risk involved in the iterative trial and error process. The fast, non-linear dynamics of quadrotors demand immediate, robust controls, a requirement iterative

methods struggle to meet. We solve these two problems by employing an MPC and combining offline and online learning. [19] combines offline and online learning for modeling the dynamics of a manipulator and used MPC for planning under the learned dynamics. The general methodology significantly differs from our approach in terms of optimized objectives. Moreover, the authors do not consider the aleatoric uncertainty of the learned dynamics to condition the controller. Hence, the controller may be overconfident during the adaptation and select poor actions.

Gaussian Processes (GPs) are another attractive class of data-driven models for modeling robot dynamics that can be easily adapted online to new data [20], [39]. However, a major drawback of GP regression is computational complexity. Since these models are non-parametric, their complexity increases with the size of the training set. This implies the need to carefully choose a subset of the fitted data that best represents the true dynamics through data reduction techniques [40]–[42]. However, since the dynamics are unknown, selecting these points might be challenging. Moreover, due to their poor scaling properties, GPs need to explicitly model individual dynamical effects (e.g., drag force [43]). Generally, the computational complexity of these non-parametric models can be reduced by learning the dynamics component-wise [44]. However, this results in the failure to capture the hidden dependencies that bound forces and torques.

Online adaptation resembles adaptive control, which seeks to estimate the model errors online and then reactively account for them at the control level [45]–[49]. However, this method trades off generalization with performance, which leads to sub-optimal flight performance [50]. Moreover, in case an MPC is employed as in [51], the optimization problem is solved by respecting defective dynamics constraints and thus potentially degrading the controller's predictive performance and accuracy as well as erroneously evaluating the feasibility of additional constraints. Conversely, in this work, we introduce an online learning procedure for continuously adapting the learned dynamics used by the MPC. Consequently, the controller fully exploits its predictive power to generate actions, resulting in superior control compared to adaptive control, as shown in the presented experimental results.

Active Learning. Active learning studies the closed-loop phenomenon of a learner selecting actions that influence what data are added to its training set [26]. When selecting the actions, the learner may seek to achieve different goals. Recent works in active learning of robot dynamics have focused on selecting actions that minimize the model (epistemic) uncertainty, hence directing learning gradients toward exploring the entire state-control space. After a globally accurate model of the system dynamics is obtained, the controller fully exploits the learned model to perform the actual task [52]. While this procedure is well-suited for robots that operate in quasi-static operating regimes, it is not practical for systems with highly-nonlinear dynamics and fast varying operating conditions such as aerial vehicles. The continuously changing dynamics would never allow the exploration stage to converge. Moreover, this exploration-exploitation strategy may not reasonably achievable by parametric models (e.g., NNs, which can potentially require

an infinite number of points to model the entire state-control space). In practice, parametric models that are continuously adapted online are not required to fully capture an accurate global model of the system to ensure controllability and stability, but should only be precise locally on a subset of the state-control space. Therefore, we propose to actively select actions that directly maximize the control performance by conditioning the learner (MPC) to the data (aleatoric) uncertainty. The conditioned controller continuously exploits the refined dynamical model and chooses "safe" actions, resulting in faster convergence of the online learning optimization as demonstrated by the proposed results.

Uncertainty Estimation Learning the dynamics purely from data poses the challenge to make the NN and its training process robust to the aleatoric uncertainty caused by noisy sensor measurements. Among the key strategies for modeling this uncertainty in NNs are Bayesian inference [53], ensemble methods [54], and test time data augmentation [55], [56]. Test time data augmentation methods quantify the predictive uncertainty based on several predictions resulting from different augmentations of the original input sample. In contrast to Bayesian and ensemble methods that rely on ad-hoc architectures or multiple NNs to estimate the uncertainty, test time data augmentation approaches require only one NN for both training and inference, do not need changes on the designed NNs, and are not sensitive to the training process initialization parameters. This motivates us to propose a memory and sample efficient test time data augmentation approach for estimating the dynamics aleatoric uncertainty and heuristically condition the MPC optimization formulation to adapt to the model uncertainties.

III. METHODOLOGY

A. Learning Discrete-Time System Dynamics

Consider a dynamical system with state vector $\mathbf{x} \in \mathbb{R}^n$ and control input $\mathbf{u} \in \mathbb{R}^m$. The discrete-time system identification task requires finding a function $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ such that

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t)$$

at every time step t. We are interested in solving the system identification problem by approximating f using a feed-forward NN $h(\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ are the NN parameters. The system identification problem then becomes finding $\boldsymbol{\theta}$ that minimizes the prediction error of $h(\boldsymbol{\theta})$ over a set of demonstration data. In this setting, the data consists of state-control trajectories with no additional labels. Hence, finding $\boldsymbol{\theta}$ is a self-supervised learning task because of the absence of an external supervisory signal (i.e., no human labeling of data [30]).

In this work, we provide as input to the NN the translational and rotational velocities, the quaternion representation of the robot orientation, and the control input. We do not provide position information as input since we assume the dynamics are position-independent. We recover the change of position information by using explicit Euler integration [57]. We employ the quaternion representation because it allows a singularity-free mapping from the unit sphere S^3 to the group $SO(3)$. However, when employing NNs to predict the system dynamics,
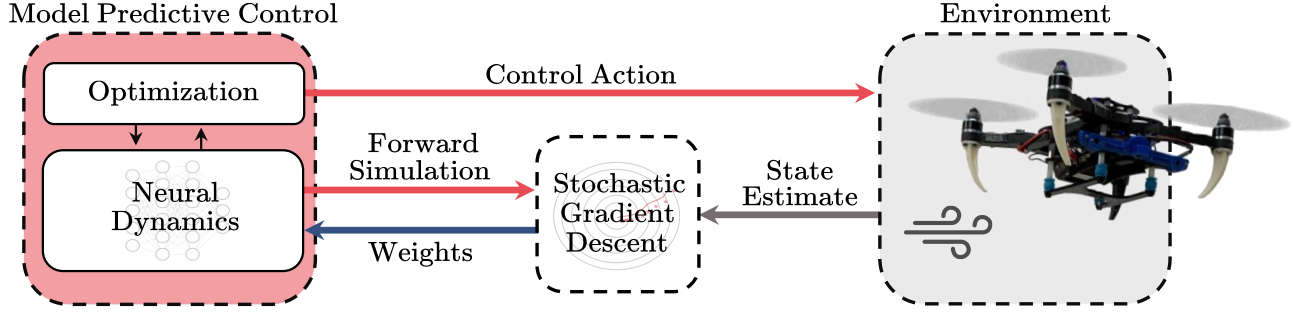
Figure 2: Online learning of the system dynamics. At each control iteration, the neural dynamics are used by the MPC to calculate the next control action to apply to the system and forward simulated to predict the next state. After actuating the predicted action by the controller, the state estimation algorithm provides the actual state reached. Finally, the neural dynamics' weights are updated by using the mismatch between the forward simulated state and the actual state reached by the robot.

the predicted orientation has no guarantee to respect the quaternion algebraic constraints (quaternion's norm can become non-unit, and thus may induce non-orthogonal column vectors in the corresponding rotation matrix). This may cause undesired behaviors at run-time and would negatively affect the control performance. Therefore, following [58], we treat the predicted orientation $\hat{\mathbf{q}}$ as a non-unit quaternion and map it to a rotation matrix

$$\hat{\mathbf{R}} = \frac{\mathbf{Q}}{||\hat{\mathbf{q}}||_2^2},$$

where

$$\mathbf{Q} = \begin{bmatrix} \hat{q}_w^2 + \hat{q}_x^2 - \hat{q}_y^2 - \hat{q}_z^2 & 2\left(\hat{q}_x\hat{q}_y - \hat{q}_w\hat{q}_z\right) & 2\left(\hat{q}_x\hat{q}_z + \hat{q}_w\hat{q}_y\right) \\ 2\left(\hat{q}_x\hat{q}_y + \hat{q}_w\hat{q}_z\right) & \hat{q}_w^2 + \hat{q}_y^2 - \hat{q}_x^2 - \hat{q}_z^2 & 2\left(\hat{q}_y\hat{q}_z - \hat{q}_w\hat{q}_x\right) \\ 2\left(\hat{q}_x\hat{q}_z - \hat{q}_w\hat{q}_y\right) & 2\left(\hat{q}_y\hat{q}_z + \hat{q}_w\hat{q}_x\right) & \hat{q}_w^2 + \hat{q}_z^2 - \hat{q}_x^2 - \hat{q}_y^2 \end{bmatrix}$$

The matrix $\hat{\mathbf{R}}$ can then be converted to any other orientation representation (e.g., a unit quaternion).

Using non-unit quaternions can also be advantageous for MPC optimization. Our past experiences with Learning MPC [59] suggest that certain integration techniques, such as the Runge-Kutta method, don't fully adhere to the SO(3) structure. Consequently, numerical integration errors may cause the quaternion's norm to drift from unity, resulting in non-orthogonal column vectors within the associated rotation matrix.

While one solution could be to impose a unit constraint (a similar approach might be feasible for the structure NN as well), it introduces further constraints to the optimization problem and thus increases computational complexity. Normalizing the quaternion after each integration might be another alternative, but it's unpractical over an MPC horizon due to the inability to ensure intermediate quaternions within the Runge-Kutta calculations accurately represent SO(3) elements.

In this context, the use of non-unit quaternion representation provides a pragmatic compromise between strict mathematical accuracy and computational efficiency, making it particularly suitable for the MPC problem.

B. Online Optimization of Learned Dynamics

In presence of variations in the operating conditions, the dynamics change, and the model learned offline is no longer accurate. Therefore, we derive an online optimization strategy that leverages the model learned offline as prior and continuously refines it to the most recent state-control observations.

We control the robot by employing a model-based controller (MPC as detailed in Section III-D) that at each time step calculates the next control action to apply to the system using a dynamics model. If the model is inaccurate, it will lead to inaccurate predictions internally and degrade the control performance. To assess the model quality independently of the employed controller, we define the mismatch between the forward simulation of the dynamics model and the next state as *forward error*. Formally, we define the forward error as

$$\mathcal{L}(\hat{\mathbf{x}}_t, \hat{\mathbf{x}}_{t-1}, \mathbf{u}_{t-1}) = ||\hat{\mathbf{x}}_t - h(\hat{\mathbf{x}}_{t-1}, \mathbf{u}_{t-1}; \boldsymbol{\theta})||_2^2, \quad (1)$$

where $\hat{\mathbf{x}}_t$ is the state estimate at time t and $\mathbf{u}_{t-1}$ is the action applied at the previous time step. We propose to use the forward error to continuously update the neural dynamics model (Figure 2). The forward error captures all the inaccuracies of the dynamics and is agnostic to the model-based control policy employed. Consequently, optimizing the forward error directly translates into adapting the dynamics to the current operating regime.

We calculate at each control iteration the forward error and use stochastic gradient descent [60] to update the last layer weights of the NN model to minimize it. Formally, let ℓ be the depth of the NN model, then $\boldsymbol{\theta}$ can be decomposed into $\theta^1, \cdots, \theta^\ell$. Consequently, we can formulate the NN such that

$$h(\mathbf{x}, \mathbf{u}; \boldsymbol{\theta}) = \theta^{\ell\top} \cdot \phi(\mathbf{x}, \mathbf{u}; \theta^1, \cdots, \theta^{\ell-1}), \quad (2)$$

where $\phi(\mathbf{x}, \mathbf{u}; \theta^1, \cdots, \theta^{\ell-1})$ represents a feed-forward NN with depth $\ell - 1$ and θ^ℓ is a weight matrix that can be though of as the last linear layer of $h(\mathbf{x}, \mathbf{u}; \boldsymbol{\theta})$. The weight update is therefore defined as

$$\theta_t^\ell = \theta_{t-1}^\ell - \eta \frac{1}{B} \sum_{i=t-B}^{t} \nabla \mathcal{L}(\hat{\mathbf{x}}_i, \hat{\mathbf{x}}_{i-1}, \mathbf{u}_{i-1}), \quad (3)$$

where t is the current control iteration and B defines the batch size. The batch size and learning rate are hyper-parameters that define the NN's ability to adapt and should be carefully chosen. For example, if the environment is significantly affected by external stochastic disturbances or the state estimation is corrupted by noisy spikes, then B should be large to collect more samples while η may be small to take short steps.

Mini-batch optimization offers several key advantages, including stability, generalization performance, and robustness [61]. In particular, by providing mini-batches, the updates to the model parameters become less erratic, and the learning process becomes more stable. Moreover, the stochasticity introduced by mini-batch optimization can help the model to explore different parts of the data distribution and avoid getting stuck in local optima, therefore boosting the generalization performance of the learning process. Finally, mini-batch optimization can be used to mitigate the effects of noisy data by averaging the gradients computed on different mini-batches, which can help to smooth out the noise and produce more accurate updates to the model parameters.

C. Uncertainty Estimation of Neural Dynamics

Learning system dynamics from collected data introduces the challenge of ensuring that the NN and its training process can handle the aleatoric uncertainty caused by noisy sensor measurements. Inspired by Unscented Kalman Filters, we use the UT [62] to estimate this uncertainty in the dynamics.

Denote with $\boldsymbol{\Sigma}$ the corresponding covariance matrix. Given that the previous mean $\overline{\mathbf{x}}_{t-1}$ and covariance $\boldsymbol{\Sigma}_{t-1}$ matrices are known, the UT is accomplished in three sequential steps:

1) Generation of a discrete distribution through sigma points having the same first and second-order moments of the prior data distribution;
2) Propagation of each point in the discrete approximation through the neural dynamics;
3) Computation of the current mean $\overline{\mathbf{x}}_t$ and covariance $\boldsymbol{\Sigma}_t$ from the transformed ensemble.

When computing the UT, state vectors need to be summed and subtracted, which is not directly possible when using quaternions because they do not belong to Euclidean space. Therefore, we need to define a procedure for adding and summing quaternions. Following [63], we employ the exponential map ($\exp_{\mathbf{q}}$) with the quaternion parameterization to commute between a minimal orientation parameterization of rotations, defined as the tangent space at the identity of the Lie group $SO(3)$, and a member of this group

$$\exp_{\mathbf{q}} = e^{q_w} \begin{bmatrix} \cos\|\mathbf{q}_v\| \\ \frac{\mathbf{q}_v}{\|\mathbf{q}_v\|}\sin\|\mathbf{q}_v\| \end{bmatrix}, \qquad (4)$$

where $\mathbf{q}_v = \begin{bmatrix} q_x & q_y & q_z \end{bmatrix}$ is the vector part of the quaternion.

The logarithm map ($\log_{\mathbf{q}}$) performs the inverse operation

$$\begin{aligned} \log_{\mathbf{q}} &= \log\left(\|\mathbf{q}\|\frac{\mathbf{q}}{\|\mathbf{q}\|}\right) \\ &= \log\|\mathbf{q}\| + \log\frac{\mathbf{q}}{\|\mathbf{q}\|} \\ &= \log\|\mathbf{q}\| + \mathbf{u}\theta \\ &= \begin{bmatrix} \log\|\mathbf{q}\| \\ \mathbf{u}\theta \end{bmatrix}, \end{aligned} \qquad (5)$$

where $\mathbf{v} = \mathbf{u}\theta$, with $\theta = \|\mathbf{v}\| \in \mathbb{R}$ and $\mathbf{u}$ unitary.

To simplify the notation, we introduce the two operators $\boxminus, \boxplus$ to perform the operations $-, +$ between the manifold and the tangent space. We define them as

$$\begin{aligned} \mathbf{q}_1 \boxminus \mathbf{q}_2 &:= 2\log_{\mathbf{q}}(\mathbf{q}_2^* \otimes \mathbf{q}_1), \\ \mathbf{q} \boxplus \boldsymbol{\delta} &:= \mathbf{q} \otimes \exp_{\mathbf{q}}(\boldsymbol{\delta}/2), \end{aligned} \qquad (6)$$

where $\boldsymbol{\delta}$ is a vector in the tangent space that represents a rotation around the axis of $\boldsymbol{\delta}$ with angle $|\boldsymbol{\delta}|$, $(\cdot)^*$ is the inverse of the quaternion, and $\otimes$ is the quaternion product

$$\mathbf{p} \otimes \mathbf{q} = \begin{bmatrix} p_w q_w - p_x q_x - p_y q_y - p_z q_z \\ p_w q_x + p_x q_w + p_y q_z - p_z q_y \\ p_w q_y - p_x q_z + p_y q_w + p_z q_x \\ p_w q_z + p_x q_y - p_y q_x + p_z q_w \end{bmatrix}. \qquad (7)$$

Leveraging the map in (6) and given that the dimensionality of $\mathbf{x}$ is n, we can now detail the UT methodology. We use the covariance matrix $\boldsymbol{\Sigma}_{t-1} \in \mathbb{R}^{n \times n}$ and mean $\overline{\mathbf{x}}_{t-1}$ to get a set $\{\mathcal{X}_{t-1}^{(i)}\}$ of $2n + 1$ *sigma points*,

$$\begin{aligned} \mathcal{X}_{t-1}^{(0)} &= \overline{\mathbf{x}}_{t-1}, \\ \mathcal{X}_{t-1}^{(i)} &= \overline{\mathbf{x}}_{t-1} \boxminus \sqrt{n+\lambda}[\sqrt{\boldsymbol{\Sigma}_{t-1}}]_i, \quad i = 1, \dots, n \\ \mathcal{X}_{t-1}^{(i)} &= \overline{\mathbf{x}}_{t-1} \boxplus \sqrt{n+\lambda}[\sqrt{\boldsymbol{\Sigma}_{t-1}}]_i, \quad i = n+1, \dots, 2n \end{aligned} \qquad (8)$$

where $\lambda = \alpha^2(n+\kappa) - n$ depends on hyper-parameters α, κ and determines the sigma points spread, and the square root of the covariance is obtained via Cholesky decomposition.

The state distribution is now represented by a minimal set of carefully chosen sigma points. These points well capture the mean and covariance of the original distribution and can be forward propagated through the neural dynamics to project them ahead of time

$$\mathcal{X}_t^{(i)} = h(\mathcal{X}_{t-1}^{(i)}, \mathbf{u}_{t-1}; \boldsymbol{\theta}), \qquad i = 0, \dots, 2n \qquad (9)$$

Finally, the posterior mean and covariance can be accurately calculated by using a weighted sample mean and covariance of the posterior sigma points

$$\begin{aligned} \hat{\boldsymbol{\mu}}_t &= \mathcal{X}_t^{(0)} \boxplus \sum_{i=0}^{2n} W_i^{(m)}\left(\mathcal{X}_t^{(i)} \boxminus \mathcal{X}_t^{(0)}\right), \\ \hat{\boldsymbol{\Sigma}}_t &= \sum_{i=0}^{2n} W_i^{(c)}\left(\mathcal{X}_t^{(i)} \boxminus \hat{\boldsymbol{\mu}}_t\right)\left(\mathcal{X}_t^{(i)} \boxminus \hat{\boldsymbol{\mu}}_t\right)^\top, \end{aligned} \qquad (10)$$

where W_i is the weight associated with the i-th sigma point defined as

$$
\begin{aligned}
W_0^{(m)} &= \frac{\lambda}{n+\lambda}, \\
W_i^{(m)} &= \frac{1}{2(n+\lambda)}, \\
W_0^{(c)} &= \frac{\lambda}{n+\lambda} + \left(1 - \alpha^2 + \beta\right), \\
W_i^{(c)} &= \frac{1}{2(n+\lambda)},
\end{aligned}
\tag{11}
$$

where β is a hyper-parameter that increases the weight of the mean of the posterior sigma points.

The proposed UT methodology for estimating the dynamics uncertainty is memory and sample efficient since it only stores one model and computes $2n+1$ forward evaluations which can be parallelized as in batch training. Moreover, the propagated sigma points capture the posterior mean and covariance accurately up to the 3rd order (Taylor series expansion) for any non-linearity [64].

D. Uncertainty-Aware Model Predictive Control

In the proposed MPC scheme, we formulate the Optimal Control Problem (OCP) with N multiple shooting steps as

$$
\begin{aligned}
\min_{\substack{\mathbf{x}_0,\ldots,\mathbf{x}_N \\ \mathbf{u}_0,\ldots,\mathbf{u}_{N-1}}} \quad & \frac{1}{2}\sum_{i=0}^{N}\tilde{\mathbf{x}}_i^\top \mathbf{Q_x}\tilde{\mathbf{x}}_i + \frac{1}{2}\sum_{i=0}^{N-1}\tilde{\mathbf{u}}_i^\top \mathbf{Q_u}\tilde{\mathbf{u}}_i \\
\text{s.t.} \quad & \mathbf{x}_{i+1} = h(\mathbf{x}_i, \mathbf{u}_i; \boldsymbol{\theta}), \quad \text{for} \quad i=0,\ldots,N-1 \\
& \mathbf{x}_0 = \hat{\mathbf{x}}_0 \\
& g(\mathbf{x}_i, \mathbf{u}_i) \leq 0
\end{aligned}
\tag{12}
$$

where $\mathbf{Q_x}$, $\mathbf{Q_u}$ are positive semi-definite diagonal weight matrices, while $\tilde{\mathbf{x}}_i = \mathbf{x}_{\text{des},i} - \mathbf{x}_i$ and $\tilde{\mathbf{u}}_i = \mathbf{u}_{\text{des},i} - \mathbf{u}_i$ are the errors between the desired state and input and the actual state and input. Therefore, the cost function calculates the discrepancy between the predicted and reference states over the time horizon, using multiple reference points. The system dynamics are represented by $h(\mathbf{x}_i, \mathbf{u}_i; \boldsymbol{\theta})$ and the initial state is constrained to the current estimate $\hat{\mathbf{x}}_0$. The problem is further constrained by state and input constraints $g(\mathbf{x}_i, \mathbf{u}_i) \leq 0$ which comprise actuator constraints [65]. Additional implementation details are reported in Section IV.

The desired control inputs $\mathbf{u}_{\text{des},i}$ can be obtained from the flat outputs of a differential-flatness planner, as described in [66], [67]. The planner, leveraging the property of differential flatness, designs optimal trajectories in the reduced space of flat outputs. These trajectories are then transformed into the full state and input space due to the unique mapping with respect to the flat outputs, resulting in the desired control inputs.

At each control iteration at time t, the mean $\overline{\mathbf{x}}_{t-1}$ and covariance $\boldsymbol{\Sigma}_{t-1}$ matrices of the state distribution are forward propagated in time as described in Section III-B. Then, the cost matrix $\mathbf{Q_x}$ in (12) is adjusted by the neural dynamics uncertainty and updated as

$$
\mathbf{Q_x} \leftarrow \mathbf{Q_x}\, \text{diag}^{-1}\left(\boldsymbol{\Sigma}_t\right),
\tag{13}
$$

where diag^{-1} is the inverse of the matrix diagonal. This adjustment process is essentially a heuristic that reduces the cost associated with deviations from the desired trajectory when the neural network's predictions exhibit higher uncertainty. Conversely, the cost increases for deviations when the neural model exhibits higher confidence. The outcome is a controller conditioned to favor safer actions that maintain the system closer to the desired state with the least uncertainty.

Once these actions are actuated by the controller, and a state estimate is received from the localization algorithm, the dynamics are subsequently refined to better fit the current operating regime. This refining process creates a continual optimization loop: the controller optimizes actions based on the current dynamic model while the dynamics are refined based on the previous optimal action predicted. This loop facilitates faster convergence and superior sample efficiency in the model training, as evidenced by our experimental results in Section V-C.

Notably, while the controller and the dynamics are both influenced by the aleatoric uncertainty, they do not directly factor in this uncertainty. Instead, it is implicitly accounted for through the update of the state weight matrix in eq. (13).

IV. EXPERIMENTAL SETUP

System. We focus on quadrotor systems for conducting experiments. We choose quadrotors because they are under-actuated mechanical systems with highly-nonlinear dynamics and fast varying flight conditions, thus they pose unique challenges for the system identification task.

We learn the dynamical system across various quadrotor configurations. Our default system configuration (Figure 3a) is a quadrotor weighing 250 g, equipped with a Qualcomm® Snapdragon™ board and four brushless motors [8]. In addition to the default configuration, we extend the system with a cable-suspended payload weighing 75 g and with a cable length of 0.8 m (Figure 3b). This alteration increases the overall system mass by 30%. Furthermore, we vary the system configuration by employing 4 different propellers, each with unique blade compositions (Figure 3c).

Dynamical Model. Following the notation introduced in Table I, we describe the quadrotor's dynamical system by the state vector $\mathbf{x} = \begin{bmatrix} \mathbf{p}^\top & \mathbf{v}^\top & \mathbf{q}^\top & \boldsymbol{\omega}^\top \end{bmatrix} \in \mathbb{R}^{13}$ and control input $\mathbf{u} = \begin{bmatrix} \Omega_0 & \Omega_1 & \Omega_2 & \Omega_3 \end{bmatrix} \in \mathbb{R}^4$. Therefore, the system identification task translates into finding a function $f : \mathbb{R}^{13} \times \mathbb{R}^4 \to \mathbb{R}^{13}$. We approximate f using a feed-forward NN that consists of 3 dense hidden layers of sizes 64, 32, 32. Each hidden layer is stacked with an ELU activation function [68]. We choose ELU over the more commonly used ReLU because it is continuous, differentiable, and slowly becomes smooth. In particular, the two activation functions are defined as follows

$$
\text{ReLU}(x) = \max(0, x),
$$
$$
\text{ELU}(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha(\exp(x) - 1) & \text{if } x \leq 0 \end{cases}.
$$

where x is the input to the layer, W, b are the weight, bias that define the layer's parameters, and α is a hyper-parameter. ReLU and ELU are very similar except for negative inputs.

(a) Default   (b) Payload   (c) Mixed Propellers
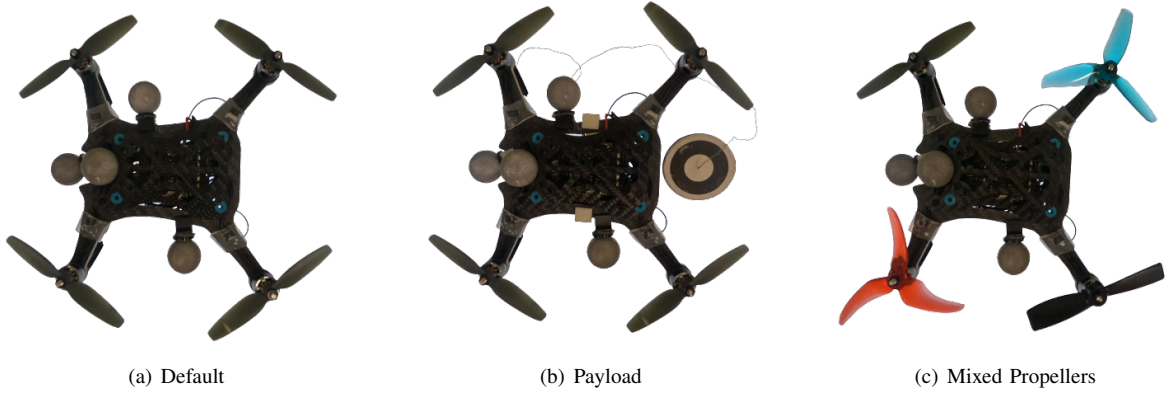
Figure 3: Quadrotor system configurations used in this work. Payload (b) extends Default (a) with a cable-suspended payload that introduces a static model mismatch due to the mass increase and stochastic effects due to the unknown payload swinging motion. Mixed Propellers (c) critically changes Default (a) by using 4 different propellers with various blade compositions.

Table I
NOTATION

$\mathcal{I}, \mathcal{B}$	inertial, body frame
$\mathbf{p} \in \mathbb{R}^3$	position in $\mathcal{I}$
$\mathbf{v} \in \mathbb{R}^3$	linear velocity in $\mathcal{I}$
$\mathbf{q} \in \mathbb{R}^4$	orientation with respect to $\mathcal{I}$
$\boldsymbol{\omega} \in \mathbb{R}^3$	angular velocity in $\mathcal{B}$
$\Omega_i \in \mathbb{R}$	force produced by the i-th propeller

Table II
ABLATION BASELINES

Name	Uncertainty Awareness	Online Learning
Static	✗	✗
Static+UA	✓	✗
Static+OL	✗	✓
Adaptive	✓	✓

Table III
CONTROL BASELINES

Name	Data-Driven Dynamics	Adaptive Control
Nominal	✗	✗
Nominal+L1	✗	✓
Static (Ours)	✓	✗
Adaptive (Ours)	✓	✓

While ELU becomes smooth slowly, ReLU sharply smoothes. Hence, ELU is better suited for MPC optimization and makes the OCP in (12) computationally easier to solve.

The NN input consists of linear velocity $\mathbf{v}$, angular velocity $\boldsymbol{\omega}$, orientation $\mathbf{q}$ and control input $\mathbf{u}$, resulting in a 14×1 tensor. The NN output is the discrete-time dynamic state of the quadrotor consisting of velocities and orientation, resulting in a 10×1 tensor.

For training, we consider the simulated and real-world data sets previously presented and publicly released by [9]. Each dataset consists of 68 trajectories with a total of $58' \, 03''$ flight time, where 6 trajectories are held out for testing (Figure 4). Based on the results from our initial experiments and analysis, we select one of the two data sets for the subsequent training and testing phases. Further details about the chosen data set for specific experiments can be found in the corresponding subsection. Additionally, a more comprehensive overview of the employed data set is provided in AppendixVII-A and AppendixVII-B.

We run the training process for 20K epochs using Adam optimizer, batch of 1024 samples, and learning rate 0.001. For the online learning phase, we set the learning rate to 0.0002 and the batch size to 20. The training process is performed using PyTorch, while the update of the weights during the online optimization is conducted in a custom implementation of the NN-based dynamics in C++.

Control. We control the quadrotor by running the MPC formulated in Section III-D on a laptop computer with an Intel Core i7-9750H CPU. We send the desired body rates and collective thrust computed by the MPC via Wi-Fi to a PID controller running onboard the quadrotor.

We formulate the MPC OCP in eq. (12) with $N = 20$ shooting steps, covering the evolution of the system over 1 s. The optimization is solved using sequential quadratic programming and a real-time iteration scheme [69] with its implementation in the `acados` package [70]. The Quadratic Programming (QP) subproblems are obtained using the Gauss-Newton Hessian approximation and regularized with a Levenberg-Marquardt regularization term to improve the controller robustness. The QPs are solved using the high-performance interior-point method in `HPIPM` [71] with full condensing and the basic linear algebra library for embedded optimization `BLASFEO` [72].

The OCP solver has been prototyped using the `acados` Python interface and the problem functions are generated using `CasADi` [73]. However, since CasADi builds a static computational graph and postpones the processing of the data, it is not compatible with PyTorch which directly performs the computations using the data. Hence, the learned dynamics using PyTorch can not be directly embedded in the formulated MPC. We solve this issue by implementing our customized version of NN-based dynamics directly in CasADi symbolics.

The OCP in eq. (12) is weighted by the definite diagonal weight matrices

$$\mathbf{Q_p} = \mathrm{diag}(200, 200, 300) \, ,$$
$$\mathbf{Q_v} = \mathrm{diag}(10, 10, 10) \, ,$$
$$\mathbf{Q_q} = \mathrm{diag}(150, 150, 200, 1) \, ,$$
$$\mathbf{Q_{\boldsymbol{\omega}}} = \mathrm{diag}(10, 10, 10) \, ,$$
$$\mathbf{Q_u} = \mathrm{diag}(5, 5, 5, 5) \, ,$$

Table IV
PREDICTIVE PERFORMANCE ON UNSEEN TESTING TRAJECTORIES

| Trajectory	$\dot{\mathbf{v}}_{\max}$ $[\mathrm{m\,s^{-2}}]$	$\dot{\boldsymbol{\omega}}_{\max}$ $[\mathrm{rad\,s^{-2}}]$	v RMSE $[\mathrm{m\,s^{-1}}]$		$\boldsymbol{\omega}$ RMSE $[\mathrm{rad\,s^{-1}}]$	
			Continuous-Time [9]	Discrete-Time (Ours)	Continuous-Time [9]	Discrete-Time (Ours)
Ellipse	5.89	7.34	0.097 ± 0.01	$\mathbf{0.078 \pm 0.01}$	0.194 ± 0.03	$\mathbf{0.089 \pm 0.02}$
WarpedEllipse	9.99	12.76	0.122 ± 0.01	$\mathbf{0.109 \pm 0.01}$	0.215 ± 0.07	$\mathbf{0.137 \pm 0.03}$
Lemniscate	13.14	31.98	0.129 ± 0.04	$\mathbf{0.111 \pm 0.05}$	0.287 ± 0.14	$\mathbf{0.140 \pm 0.09}$
ExtendedLemniscate	7.76	27.95	0.099 ± 0.04	$\mathbf{0.064 \pm 0.03}$	0.170 ± 0.13	$\mathbf{0.102 \pm 0.03}$
Parabola	5.91	6.57	$\mathbf{0.032 \pm 0.01}$	0.065 ± 0.01	0.102 ± 0.04	$\mathbf{0.099 \pm 0.02}$
TransposedParabola	17.86	54.90	0.141 ± 0.03	$\mathbf{0.107 \pm 0.04}$	0.398 ± 0.22	$\mathbf{0.147 \pm 0.04}$
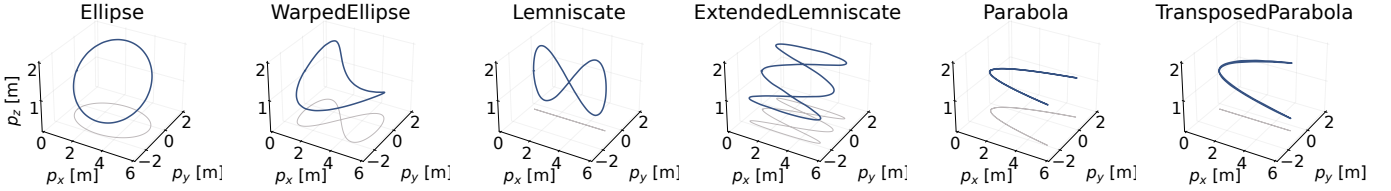
Figure 4: Testing trajectories considered in this work.

where diag denotes a diagonal matrix and is formulated to respect the state and input constraints

$$-10 \le \mathbf{v}_i \le 10 \qquad \forall i \in [0,2] \; [\mathrm{m\,s^{-1}}],$$
$$-3.14 \le \boldsymbol{\omega}_i \le 3.14 \qquad \forall i \in [0,2] \; [\mathrm{rad\,s^{-1}}],$$
$$0.01 \le \boldsymbol{\Omega}_i \le 1.124 \qquad \forall i \in [0,3] \; [\mathrm{N}].$$

The UT methodology depends on the choice of the hyper-parameters α, κ, and β. which are selected based on the characteristics of the state and measurement noise and the nonlinearity of the system. In this work, we set $\alpha = 0.001$, $\kappa = 1.0$, and $\beta = 2.0$ assuming a Gaussian prior distribution.

Localization. We conduct the experiments in an environment equipped with Vicon motion capture system. This choice ensures that the experimental results precisely evaluate the dynamics and controller's accuracy. Introducing onboard sensors presents challenges like sensor noise and state estimation drifts, which would significantly affect tracking performance and, in turn, the demonstration of our contributions.

Metrics. We compare our approach against multiple baselines for dynamics learning and quadrotor control based on the Root Mean Squared Error (RMSE) and the Cumulative RMSE (CRMSE). The error metrics between the true state $\mathbf{x}$ and the predicted state $\hat{\mathbf{x}}$ are formulated as

$$\mathrm{RMSE}(\mathbf{x},\hat{\mathbf{x}}) = \sqrt{\frac{1}{N}\sum_{i=0}^{N-1}(\mathbf{x}_i - \hat{\mathbf{x}}_i)^{\top}(\mathbf{x}_i - \hat{\mathbf{x}}_i)},$$

$$\mathrm{CRMSE}(\mathbf{x}_t,\hat{\mathbf{x}}_t) = \sum_{i=0}^{t}\mathrm{RMSE}(\mathbf{x}_i,\hat{\mathbf{x}}_i),$$

where N is the number of components of the state and t is the number of control iterations from the start of the experiment.

The CRMSE allows us to easily analyze the trend of the error function over a certain period, hence it is a well-suited metric for understanding the role of methods that make a continuous optimization over time.

V. RESULTS

We design our evaluation procedure to address the following questions. (i) How does the predictive performance of our proposed discrete-time model compare with a continuous-time baseline [9] when tested on real-world data? (ii) What advantages does the active learning of dynamics present when controlling the quadrotor? (iii) Can the proposed approach adapt to challenging flight regimes and how does it compare against classical [8] and adaptive control baselines [74]? (iv) How can we statistically demonstrate the convergence of the proposed framework? We encourage the reader to consult the supplementary material at the end of this paper and video for additional results.

A. Discrete vs. Continuous-Time Dynamics

We compare the predictive performance of the proposed dynamics model and a continuous-time baseline. Both models are trained and tested on the real-world dataset (Figure 4) The continuous-time baseline corresponds to the PI-MLP model proposed in [9], which was demonstrated to generate more precise predictions compared to multiple baselines for the quadrotor's system identification task, from traditional approaches to learning-based methods. We feed the continuous-time baseline with the same input information as the proposed discrete-time approach. The baseline outputs the continuous-time dynamics of the quadrotor's system, i.e. linear acceleration $\dot{\mathbf{v}}$ and angular acceleration $\dot{\boldsymbol{\omega}}$. Then, we discretize the continuous-time solutions by employing the Runge-Kutta 4th-order numerical integration.

Table IV summarizes the results of this experiment. Both models demonstrate excellent performance over all the proposed maneuvers, capturing all the complex non-linear effects, and performing accurate predictions. However, despite being trained on the same dataset and sharing a quasi-identical architecture, the proposed model consistently and significantly outperforms the continuous-time baseline, improving the prediction error by up to 50%. The key reason for this dominant performance
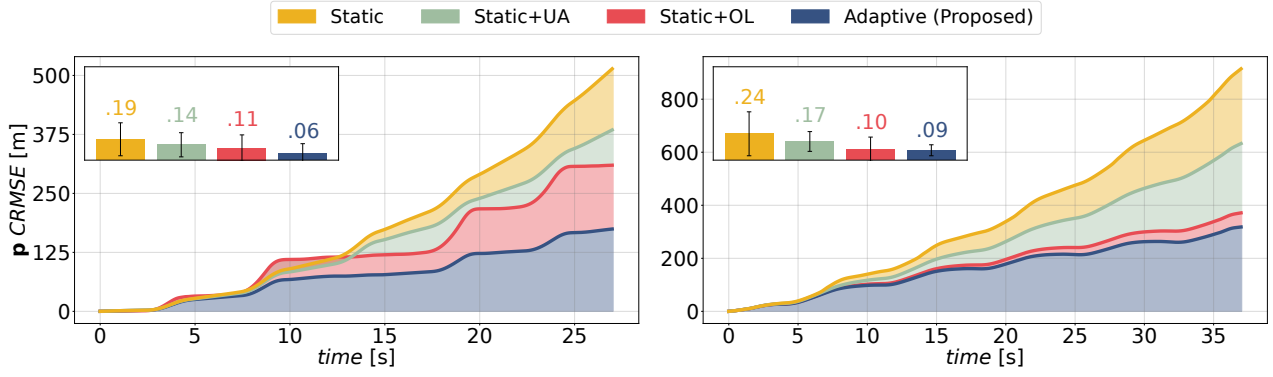
Figure 5: Benefits of combining online learning with uncertainty estimation when continuously tracking WarpedEllipse (left) and Lemniscate (right) trajectories in the real world. The RMSE is accumulated over multiple loops to validate the increased stability introduced by the uncertainty estimation and online optimization. (Inset) Average RMSE over the entire flight performance.

Table V
CLOSED-LOOP TRACKING PERFORMANCE IN SIMULATION

Trajectory	Static $\mathbf{p}$ RMSE [m]	Static+UA $\mathbf{p}$ RMSE [m]
Ellipse	0.34 ± 0.10	$\mathbf{0.29 \pm 0.08}$
WarpedEllipse	0.23 ± 0.10	$\mathbf{0.21 \pm 0.12}$
Lemniscate	0.14 ± 0.09	$\mathbf{0.12 \pm 0.06}$
ExtendedLemniscate	0.09 ± 0.06	$\mathbf{0.09 \pm 0.05}$
Parabola	0.28 ± 0.12	$\mathbf{0.20 \pm 0.14}$
TransposedParabola	0.17 ± 0.06	$\mathbf{0.14 \pm 0.06}$

is the dynamics' level of abstraction that the model learns. Specifically, learning continuous-time dynamics requires explicitly extracting linear and angular accelerations from flight data. However, this usually simply translates into computing the first-order derivative of velocities that are corrupted by noisy measurements. Hence, the computed accelerations are affected by noisy spikes that limit the learning capabilities of the neural model. Low-pass filters can be applied to minimize the noise in the collected data. However, there is no guarantee that the filters' outputs are the true dynamics and that the discarded information is mere noise. Moreover, using filters introduces time delays that may render online learning not effective. Conversely, learning discrete-time dynamics does not require any velocity differentiation and therefore the NN is trained to regress less noisy labels.

B. Leveraging Uncertainty for Control and Dynamics Learning

We demonstrate the benefits of conditioning the MPC objective function on aleatoric uncertainty for controlling the quadrotor and boosting the online learning convergence. We utilize the MPC, as outlined in Section III-D, to control the quadrotor in order to continuously track multiple testing trajectories in both simulated and real-world settings. We compare the tracking performance when selectively omitting the proposed Uncertainty-Awareness heuristic (*UA*) and Online Learning strategy (*OL*), resulting in four different control methods as reported in Table II. For these experiments, the

dynamics model is trained on the real-world dataset, even for the flights in simulation.

Table V provides the mean and standard deviation of the positional RMSE while tracking multiple testing trajectories in simulation. The results reveal that incorporating the estimated uncertainty into the MPC optimization function improves the stability of the flight controller. As a result, the uncertainty-aware MPC (*Static+UA*) achieves better accuracy and precision in tracking performance compared to the conventional MPC (*Static*), reducing the positional RMSE by up to 30%. Moreover, error plots show better tracking performance over time for the uncertainty-informed MPC compared to the classical MPC.

In order to provide a fair comparison between uncertainty-aware and uncertainty-unaware MPCs, we use a simulation environment. However, this environment lacks aerodynamic forces, motor dynamics, communication delays, and other non-linear effects present in real-world scenarios. This results in the aleatoric uncertainty of the dynamical model remaining moderate, hence tracking performance is comparable between the uncertainty-aware and uncertainty-unaware MPC during the early stages of flight. As the flight continues, cumulative tracking errors compound, degrading overall performance. In these later stages, the uncertainty-awareness procedure proves critical for maintaining performance.

The results of similar experiments conducted in the real world are shown in Figure 5. These findings underscore the importance of our proposed online learning procedure for continuously improving the dynamics model over time, facilitating high-performance control on aggressive trajectories not previously seen. When the online learning procedure is enabled (*Static+OL*, *Adaptive*), the quadrotor's tracking performance is improved by up to 60%. Furthermore, the results indicate that conditioning the MPC objective function using the estimated uncertainty enables faster model learning convergence and sample efficiency, hence better tracking performance over time compared to an uncertainty-unaware controller. When the online learning procedure is enabled and the MPC is conditioned (*Adaptive*) - the proposed active learning method, the tracking performance of the quadrotor is dramatically improved by up to 70%. The primary reason for this is that

when the MPC is uncertainty-informed, it opts for safer actions (i.e., prioritizes tracking certain state components), favoring the exploitation of the dynamics model over exploration. This, in turn, significantly boosts the convergence and sample efficiency of the online model training.

C. Adaptation to Challenging Operating Conditions

We evaluate our proposed approach (*Adaptive*) against multiple baselines for quadrotor control in several challenging real-world flight operating conditions, which involve extensive modifications to the system configuration or external disturbances. Specifically, we consider an MPC that leverages a nominal model of the quadrotor as in [8] (*Nominal*) and the nominal MPC augmented by an L1 adaptive controller as in [74] (*Nominal+L1*).

Nominal baseline models the system's dynamics through non-linear differential equations and applies the MPC defined in Section III-D to control the quadrotor. However, it falls short in representing complex non-linear phenomena such as friction, deformation, aerodynamic impacts, and vibrations. To overcome this limitation, we consider a second baseline: an adaptive controller. This controller augments the MPC's predictions in real time by modeling the complex non-linear phenomena that the nominal model fails to capture. We specifically focus on the L1 adaptive controller recently proposed in [74] as it has been effectively utilized for quadrotor control and its code is publicly available.

We keep the parameters the same as the original implementation of the L1 adaptive controller. For more details on the comparison standards, please may refer to the corresponding papers. Table III provides an overview of the different control baselines used in this set of experiments.

We use the controllers mentioned above to control our quadrotor in the following flight operating conditions

- *Payload transportation*: learning the dynamics purely from data poses the challenge to make the NN generalize to domain shifts between training and testing distributions. We study the capability of the proposed approach to adapt to significant changes in the vehicle's mass by attaching to the quadrotor's body a cable-suspended payload with mass 75 g and cable length 0.8 m (Figure 1a). The quadrotor is required to take off and track a circular trajectory of radius 1 m. Note that the payload not only increases the mass of the quadrotor by 30%, but it also introduces stochastic disturbances due to the payload swing motions;
- *Mixing propellers*: propellers provide lift for the quadrotor by spinning and creating airflow. Mixing different propellers may cause the system to lose stability and fail to track even simple near-hovering trajectories [75]. Therefore, we validate the adaptability of the proposed approach to significant domain shifts by altering the quadrotor's propellers. Specifically, we significantly change the system configuration by using 4 different propellers with various blade compositions (Figure 1b). The quadrotor is required to take off and hover at 0.5 m height;
- *Wind disturbance*: accurate tracking under challenging wind conditions is difficult due to the stochastic nature of

wind and the domain shift caused by the new environment dynamics never seen during training. We analyze the adaptability of the proposed approach when an industrial fan blows towards the quadrotor with air speeds up to $3\,\mathrm{m\,s^{-1}}$ (see Figure 1c). The quadrotor is required to take off and hover at 0.5 m height.

Figure 6 illustrates the results of these experiments. The results demonstrate that the learned dynamics can be successfully incorporated into an uncertainty-aware MPC and combined with an online learning strategy to continuously and actively adapt to multiple challenging flight operating conditions.

When extending the system with a cable-suspended payload, non-adaptive controllers (*Nominal*, *Static*) struggle to accurately track the desired trajectory, which may lead to fatal navigation failures. Contrarily, augmenting the MPC with an L1 adaptive controller allows us to continuously refine the control action to match the unmodelled disturbances and improve the tracking error by 33%. However, this augmentation does not improve the dynamics model leveraged by the MPC, hence the optimization problem is still solved by respecting defective dynamics constraints. Conversely, our approach fully leverages online learning to adapt the NN-based dynamics model to the present flight regime, and thus the MPC predictive power is fully exploited during the controller optimization, resulting in a tracking error reduced by 75% with respect to the *Nominal+L1* controller and 92% compared to the *Static* baseline.

When mixing different propellers, the quadrotor undergoes severe oscillations and may fail to track even simple near-hovering trajectories, as demonstrated by *Nominal* and *Nominal+L1* flight performance. The *Nominal* controller heavily relies on an old system model and fails to adapt to the new system dynamics. Consequently, after a few seconds, the quadrotor crashes. L1 adaptive control augments the MPC predicted actions to stabilize the system, however, it is still tightly bound to the old physical dynamics and also leads to a fatal crash. Contrarily, when leveraging NN-based models, the MPC controls the quadrotor to hover close to the desired position. In particular, the *Adaptive* controller successfully stabilizes the system, hence validating the proposed approach. Interestingly, the learning transient during the adaptation of the NN-based dynamics does not introduce additional oscillations but leads to a stable tracking of the desired position.

Accurate tracking under challenging wind conditions is difficult due to the highly stochastic nature of wind that imposes continuous vibrations on the quadrotor, as reflected in the experimental results. When leveraging the physics-based model, the MPC tracks the desired position approximately well. Contrarily, the NN-based dynamics model learned offline struggles to generalize well to the new real-world dynamics and the flight performance of the *Static* baseline deteriorates compared to the *Nominal* controller. On the other hand, when employing the proposed *Adaptive* controller, refining the dynamics model enables the MPC to accurately track the reference state after a minute of online learning, outperforming *Nominal* even when augmented with an L1 adaptive controller.
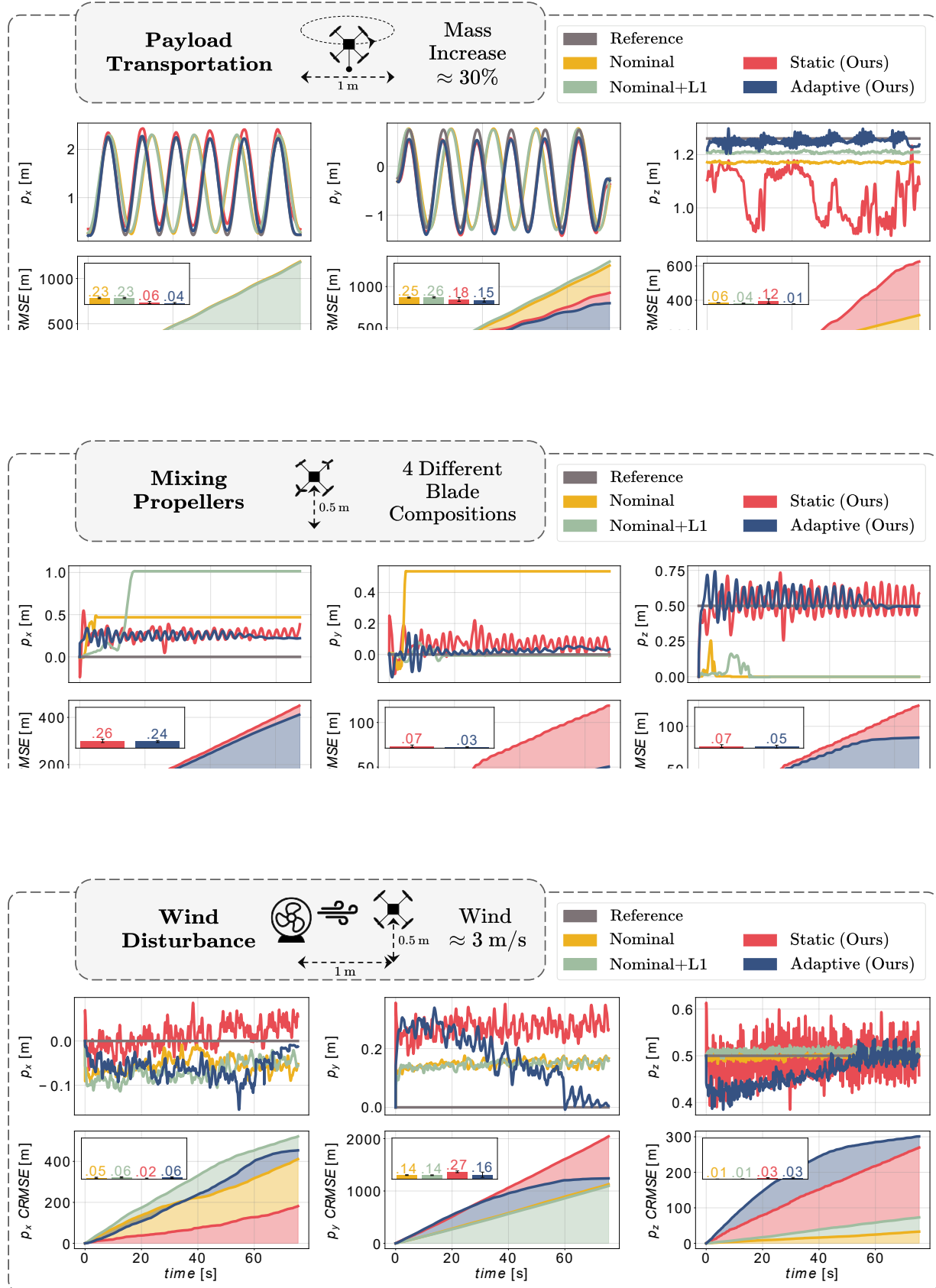
Figure 6: Role of dynamics adaptation in multiple flight operating conditions. (Inset) Average RMSE.
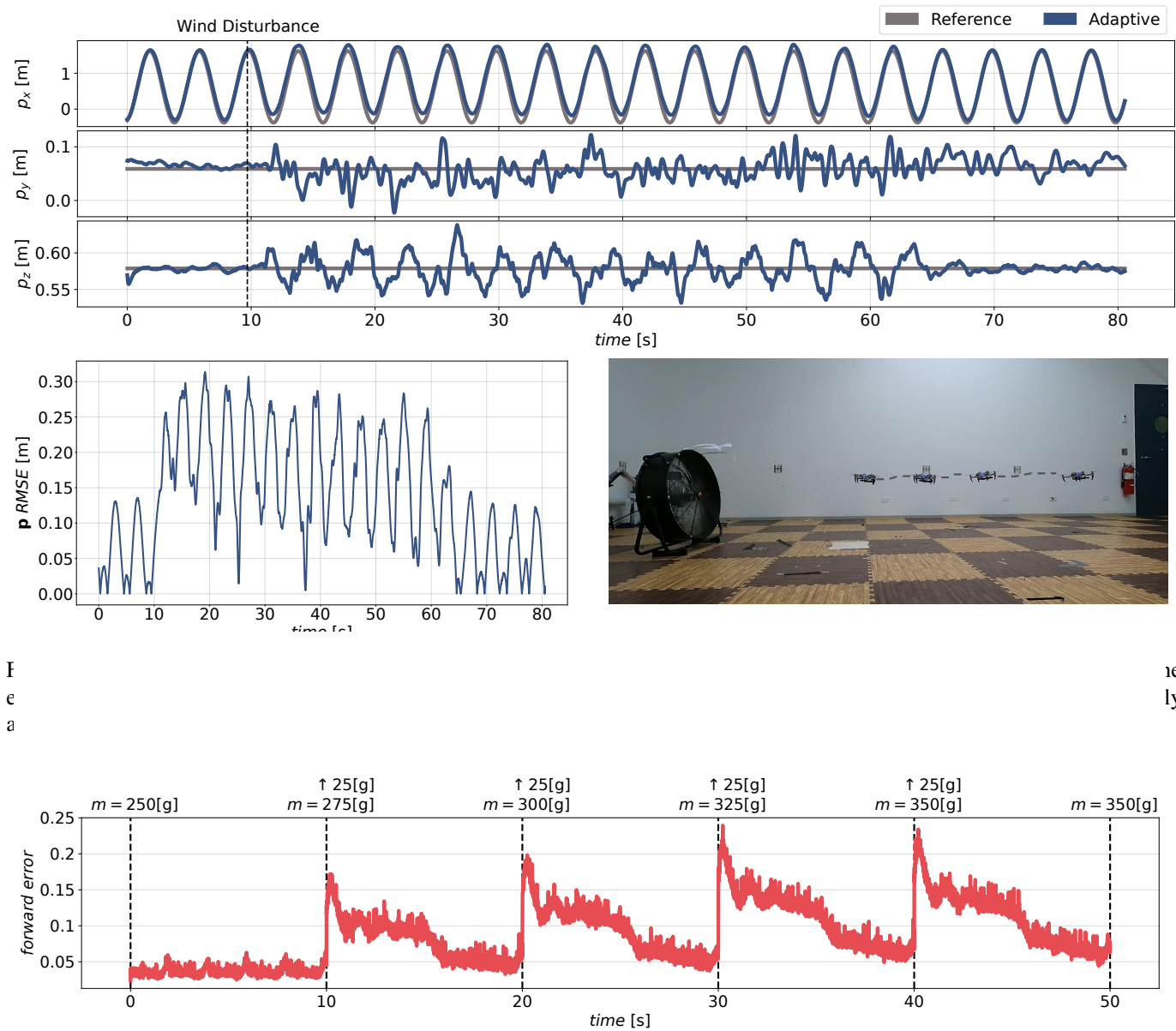
Figure 8: Analysis of the forward error convergence when the system's mass m is increased by 25 g every 10 s. Each time the mass is altered, the forward error increases, as the internal simulated predictions made by the MPC differ substantially from the actual state estimations. However, the online optimization consistently converges by minimizing the error and capturing the system configuration modification. This showcases the resiliency of the proposed learning approach.

D. Convergence Analysis

Theoretical guarantees of convergence are ideal in the development of learning algorithms as they ensure the reliability and predictability of the algorithm's performance under certain conditions. However, real-world scenarios, especially those involving quadrotors, present unique challenges, such as non-linearity, noise, disturbances, non-stationarity, and partial observability of the environment. These complexities often preclude the establishment of solid theoretical guarantees of convergence. Hence, in this work, we propose an empirical approach to study the convergence properties of our approach. We argue that the performance of an algorithm in a complex and unpredictable real-world scenario is the ultimate validation of its effectiveness.

Previous experiments have shown empirically the effectiveness of our approach in improving the model's accuracy over time, in both simulated and real-world environments. Building on this, we further explore the convergence capabilities of our approach in this analysis, focusing on its ability to maintain its effectiveness in the face of multiple unexpected consecutive disturbances, providing consistent tracking performance before and after it adapts to new dynamics. Particularly, to test the algorithm's resiliency in the real world, we introduce an unmodeled wind disturbance along the quadrotor's trajectory as it follows a linear path over the x axis, observing the tracking performance starting from the initial introduction of the disturbance. We use an uncertainty-aware MPC and a dynamics model trained on real-world data to control the

Table VI
COMPUTATION TIME ANALYSIS [ms]

Model Architecture	Intel Core i7-9750H			NVIDIA Jetson Xavier NX		
	Model Prediction	Model Update	MPC Prediction	Model Prediction	Model Update	MPC Prediction
$[64, 64, 32]$	0.004	0.001	284	0.079	0.005	167
$[128, 64, 32]$	0.007	0.001	257	0.088	0.005	103
$[128, 128, 64]$	0.011	0.001	185	0.097	0.009	59
$[256, 128, 128, 64]$	0.017	0.001	75	0.121	0.009	11

Table VII
TRACKING PERF AFTER 1 LOOP [m]

Batch Size	Learning Rate			
	$2e-1$	$2e-3$	$2e-5$	$2e-7$
1	crash	crash	0.44	0.38
10	1.22	0.19	0.10	0.34
20	0.95	0.25	0.11	0.31
40	0.52	0.31	0.18	0.33

Table VIII
TRACKING PERF AFTER 25 LOOPS [m]

Batch Size	Learning Rate			
	$2e-1$	$2e-3$	$2e-5$	$2e-7$
1	crash	crash	0.36	0.39
10	crash	0.17	0.06	0.31
20	0.92	0.21	0.03	0.22
40	0.68	0.08	0.09	0.36

Table IX
TRACKING PERF AFTER 50 LOOPS [m]

Batch Size	Learning Rate			
	$2e-1$	$2e-3$	$2e-5$	$2e-7$
1	crash	crash	0.27	0.38
10	crash	0.09	0.07	0.31
20	1.01	0.16	0.04	0.21
40	0.95	0.12	0.03	0.37

quadrotor. Figure 7 shows the results of this experiment.

We continue our empirical convergence analysis by studying the ability of the learning algorithm to continuously apply its learned knowledge to new, unseen conditions consistently. Specifically, we continually control our quadrotor to track a circular trajectory, increasing the system's weight by 25 g every 10 s, and monitoring the forward error convergence over time. We conduct this experiment in a simulation environment, using the uncertainty-aware MPC and a dynamics model trained on simulated data to control the quadrotor. In doing so, the dynamics of the model closely match those of the simulation environment, and any disturbances introduced in the simulator become the sole source of dynamic uncertainty for the model to adapt to. Figure 8 presents the results of this experiment.

Whenever a new disturbance appears, impacting the dynamics, the model is adjusted by optimizing the forward error. Upon convergence, the model aligns with the new dynamics, allowing the controller to accurately track the desired trajectory. This is particularly visible in the resiliency analysis, where the precise tracking performance preceding the wind disturbance is restored after the convergence of the online optimization. Specifically, after convergence, the overshooting and undershooting behaviors dictated by the unmodeled wind gusts cease to affect the tracking performance.

These results underscore the resilience and generalization capabilities of our proposed approach when faced with new and unseen dynamics. These are essential attributes for a successful online learning algorithm, as they ensure its reliable performance across a variety of conditions and scenarios. These results offer empirical evidence of the dynamic model's capacity for continuous improvement as well as also empirically validate the convergence of our proposed framework.

E. Computational Complexity Analysis

Our framework has been meticulously engineered with the objective of operating optimally on embedded platforms. Nonetheless, throughout the course of our study, we confronted

a significant computational challenge: the implementation of the MPC on the Qualcomm® Snapdragon™ board. Despite simplifying the task to use only the *Nominal* model, the hardware was incapable of conducting the required computations in real time However, this does not discount the proficiency of the Qualcomm® Snapdragon™ board entirely as it is more than capable of handling the inference of our NN-based dynamics.

To underscore the versatility and scalability of our proposed method, we expanded our testing environment. Progressing from laptop-grade hardware, we implemented our method on NVIDIA's Jetson Xavier NX - a popular choice for computational tasks in the field of robotics [76]–[78]. The successful execution on this platform evidences that our method holds promise for scalability and adaptability to commonly utilized hardware platforms in robotics research.

To demonstrate this, we have performed a detailed analysis of the computational cost when running the entire proposed framework with different dynamics model choices on a laptop computer with an Intel Core i7-9750H CPU and on an embedded computer with an NVIDIA Jetson Xavier NX. In particular, we have investigated the computation time required by the different dynamics models to infer a prediction and update their weights and the time required by the MPC to perform its optimization and predict an optimal action. The results of this analysis are reported in Table VI. Note that the model and MPC hyper-parameters are maintained equal to the ones presented in Section IV.

The results show that the complexity of the dynamics model significantly impacts the computational time required by the MPC to solve its optimization problem. In particular, on embedded systems, deep models are impractical for real-time inference. This underlines the importance of tailoring the model architecture based on the specific hardware resources available. If the dynamics model does not ensure a minimum inference time, which would permit the MPC to compute actions at an acceptable rate (typically around 100 Hz for micro aerial vehicles), the actuation of each action commanded by the

controller would be delayed. This delay can lead to a significant degradation in flight performance, potentially causing failures and crashes. Therefore, it is critical to ensure that the model complexity is suited to the hardware and control system in order to maintain the necessary real-time control performance.

Furthermore, the results highlight that the time required by the model to update its weights is negligible, even on embedded systems. We believe that this is due to two key reasons. First, the weight update of the dynamics model entails only refining the weights and biases of the output layer. Second, the dynamics model is efficiently implemented in C++ in our custom implementation, ensuring maximum performance.

Finally, we study the relationship between the state dimension and the computational complexity of the proposed framework. The complexity of a NN, such as the one utilized in this study, is determined by several factors, including the input size, the number of layers, the number of neurons per layer, and the type of activation function employed. For the sake of clarity, let's consider a simplistic NN with a single hidden layer. The primary operations of such a model include matrix multiplication and the application of the activation function. Given an input vector of size n, a hidden layer containing h neurons, and an output layer of size o, we can approximate the computational complexity as follows:

- The input to hidden layer computation: This process involves the multiplication of an $n \times 1$ input vector with an $n \times h$ weight matrix, subsequently adding a $1 \times h$ bias vector. Therefore, the computational complexity of this operation is approximately $O(nh)$.
- Applying the activation function: As this function is applied individually to all h neurons within the hidden layer, its computational complexity is approximately $O(h)$.
- The hidden to output layer computation: This process involves the multiplication of a $h \times 1$ hidden layer output vector with an $h \times o$ weight matrix, subsequently adding a $1 \times o$ bias vector. Therefore, the computational complexity of this operation is approximately $O(ho)$.
- Applying the activation function to the output layer: As this function is applied individually to all o neurons within the output layer, its computational complexity is approximately $O(o)$.

Consequently, for a single pass or forward propagation, the overall computational complexity is roughly $O(nh+h+ho+o)$. As depicted, while the input size n does indeed influence computational complexity, it constitutes only a fraction of the total calculation. The number of neurons h in the hidden layer, alongside the number of output neurons o, considerably contributes to the overall computational complexity. This complexity would be further multiplied by the number of samples processed in a batch, and yet again by the number of epochs for training. Furthermore, this complexity does not account for the backward pass used during training for backpropagation, which would approximately double the computational complexity as it involves similar computations to the forward pass. Nevertheless, it's essential to note that while this analysis is applicable to the dynamics model, it does not apply to the computational complexity of the MPC.

As demonstrated in previous work [9], the state dimension significantly impacts the controller's computation, even when optimized using ACADOS or other MPC-specific frameworks for embedded systems. Consequently, this places a limitation on the integration of historical information in the state, which would be highly beneficial to the learning convergence.

F. Choosing Hyper-parameters for Online Optimization

The efficacy of the learned model's online adaptation to unseen dynamics significantly depends on two hyper-parameters: batch size and learning rate. Improper setting of these parameters can hinder the model's learning ability or lead it to adapt to insignificant data noise due to faulty sensors or inaccurate state estimates. Therefore, we conducted a study to understand the dependency of these hyper-parameters during the model's adaptation to unseen dynamics.

In this set of experiments, we continuously control our quadrotor to track the Ellipse testing trajectory and monitor the positional RMSE after 1, 25, and 50 loops. This experiment was conducted in a simulated environment, using the uncertainty-aware MPC and a dynamics model trained on real-world data to control the quadrotor. The goal of the online optimization is to adapt the dynamics model to the simulated dynamics. By varying batch sizes and learning rates, we could analyze their relationship with control performance. Our findings are presented in Tables VII, VIII, and IX.

The results demonstrate a clear correlation between batch size and learning rate. Achieving a balance between these two parameters is crucial. Larger learning rates are more compatible with larger batch sizes. Conversely, when a high learning rate is paired with small batch size, the model may undergo severe changes, which could potentially cause instability and even crashes. In general, larger batch sizes are preferred, especially when adapting to real-world dynamics that are highly non-linear and subject to noisy state estimates. However, it is important to note that larger batch sizes imply slower adaptation due to the increased amount of state estimates required to compute the weight update.

Significantly, our findings show that an exceedingly low learning rate can inhibit the model from adapting appropriately, leading to suboptimal performance. This underscores the importance of consistently updating the dynamics model using online optimization strategies.

VI. DISCUSSION

The experimental results demonstrate that the learned dynamics can be successfully incorporated into an uncertainty-aware MPC framework and combined with an online learning strategy to continuously and actively adapt to static (e.g., changes of mass, mixing of propellers) and stochastic (e.g., payload swings, wind disturbance) model mismatches. At the same time, the results highlight two important aspects of the proposed online optimization strategy.

First, even though the training and testing dynamics distributions are substantially different (Figure 3), the learning transients do not introduce catastrophic oscillations during the online optimization. We consider mini-batch optimization key

for this behavior. By leveraging batches of observed state-control pairs, the gradient points in the direction of smooth transitions between different operating regimes. In general, the experimental results show a convergence of the stability of the system (i.e., the controller's initial overshooting is minimized due to a refined dynamics model). This behavior is well illustrated by the mixing propellers experiment and the relative clip in the supplementary video. During the wind disturbance experiments, the stochastic nature of the wind disturbance makes the oscillating behavior difficult to minimize. However, the quadrotor is still able to accurately track the desired trajectory and significantly improve the tracking performance.

Second, during online optimization, the gradient directions strictly depend on the magnitude of the forward error on each state component. Hence, if the error over a component is much larger than the error over the others, the update of the dynamics model will most likely improve the predictive performance over that component. Sometimes, this may come at the cost of marginally degrading the performance of the other components. However, since this degradation is minimal, this can be considered negligible. For example, during the wind disturbance experiment (Figure 6), the forward error over the y component of the position is much more significant than the error over the other state components. Hence, the online optimization strategy focuses on minimizing the forward error over this component. As a result, despite the stochastic nature of wind that makes the environmental dynamics extremely difficult to model, after about 60 seconds the quadrotor reaches the desired trajectory. Similarly, during the payload transportation experiment, the online optimization successfully minimizes the error over the z component of position due to the mass mismatch. However, in this experiment, the error over the y component of the position is still significantly high. We believe that such an error is related to the stochastic oscillation of the cable-suspended payload. A solution to this problem would be to have extra information in the input to the dynamics model about the payload position with respect to the quadrotor [37], hence making the state fully observable [79]. Consequently, the dynamics model would be able to understand if the cable is slack or tilted and find a correlation to its dynamical evolution. However, as we do not provide this information to the dynamics model to maintain a task-agnostic approach, the optimization strategy struggles to optimize the tracking performance over the $x - y$ plane.

VII. CONCLUSION AND FUTURE WORKS

Active learning of the system dynamics has the potential to terrifically impact the development of multiple robotic systems, enabling fast modeling and high-performance control in multiple operating conditions. Therefore, in this work, we presented a self-supervised framework for actively learning the dynamics of nonlinear robotic systems. Our innovative approach combines MPC with aleatoric uncertainty, uniquely combining offline dynamics learning with active online adaptation. Furthermore, we addressed the challenges related to transitioning this approach from theory to practice on SWaP-constrained robots, presenting a clear formulation that can be implemented and reproduced. Through extensive experimentation, we demonstrated that learned dynamics continuously adapt to multiple challenging flight regimes and operating conditions, enabling unprecedented flight control. These results provide robust empirical evidence of the proposed approach's resiliency which is critical for practical applications.

Our results, as shown in Table IV, indicate that the proposed discrete-time dynamics model consistently surpasses a state-of-the-art continuous-time dynamics baseline [9] in terms of predictive performance. Currently, the training process of our neural dynamics model does not take into account the impact of noisy labels [80]. Future work will delve into novel loss functions suitable for learning with noisy labels, and the influence of dataset size on the dynamics.

The ablation study presented in Figure 5 validates the proposed online learning procedure for enabling high-performance control on unseen aggressive trajectories. Furthermore, it shows that conditioning the MPC with the estimated uncertainty facilitates quicker model learning convergence and improves sample efficiency. Our proposed uncertainty-awareness heuristic utilizes uncertainty derived directly from the data. However, it does not account for the uncertainty originating from the limitations of the selected dynamics model, namely the epistemic uncertainty [55]. Future work will aim to expand the heuristic to encompass epistemic uncertainty, focusing on errors caused by inadequate model structure or lack of knowledge due to unencountered samples.

The ablation study presented in Figure 5 validates our online learning procedure for achieving high-performance control on unseen aggressive trajectories. Furthermore, it shows that by conditioning the MPC with the estimated uncertainty, model learning convergence accelerates and sample efficiency improves. Our proposed heuristic for uncertainty awareness utilizes uncertainty extracted directly from the data but does not consider uncertainty originating from limitations in the chosen dynamics model, namely epistemic uncertainty [55]. Future work will seek to refine the heuristic to encompass epistemic uncertainty, focusing particularly on errors caused by an insufficient model structure or lack of knowledge due to unencountered samples.

Figure 6 demonstrates that the learned dynamics can be effectively incorporated into an uncertainty-aware MPC framework and synergized with an online learning strategy for continuous adaptation to challenging operational environments. However, while the online learning method showed impressive performance, it demands a considerable amount of time to update the NN's weights and simultaneously maintain stable control performance. This is primarily due to the current strategy's reliance on a shallow NN architecture, limiting the method to updating a restricted number of weights and biases. Future research will investigate computationally feasible methods for utilizing deeper NN architectures that permit the updating of a larger number of parameters while keeping numerous layers frozen. Thus, determining the optimal number of layers to update and the sequence of updates will be a significant focus.

Figure 8 validates the resiliency of the proposed approach. At the same time, it shows a small destructive behavior due

to the competing updates from the offline and online updates. As the mass increases, the frozen backbone dynamics learned offline differ more and more from the current simulated ones. Hybrid control is an intriguing area of future research that targets this problem, seeking to strike the optimal balance between offline and online learning phases [81]–[83]. Building on these findings, we plan to enhance our approach with hybrid learning strategies to increase learning efficiency and overall control performance.

In our work, we employed non-unit quaternions as rotation representations. While this approach is highly efficient computationally, it is not ideal for learning discrete-time dynamics due to the non-smooth manifolds it induces - it can be potentially affected by discontinuous jumps, such as quaternions flipping from +q to -q. Future work will explore other rotation representations that are smooth and minimal [84].

Lastly, our extensive experiments have demonstrated that our algorithm can enhance performance over time and adapt to varying operating conditions. This provides robust empirical evidence of its generalizability and resilience, which are crucial for practical applications. Nevertheless, empirical results, while offering convincing evidence of efficacy, do not supplant the necessity for formal analysis. In the future, we aim to bolster our empirical findings with theoretical insights, thereby deepening our understanding of our algorithm's behavior and performance.

APPENDIX

A. Data Collection

The data is collected by controlling the quadrotor in a series of flights both in simulation and in the real world, resulting in two datasets with analogous trajectories. The simulated flights are performed in the Gazebo simulator, while the real-world flights are performed in an indoor environment $10 \times 6 \times 4 \mathrm{~m}^3$ at the Agile Robotics and Perception Lab (ARPL) at the New York University. The environment is equipped with a Vicon motion capture system that allows recording accurate position and attitude measurements at 100 Hz. Additionally, we record the onboard motor speeds. Each dataset consists of 68 trajectories with a total of $58' \, 03''$ flight time. The trajectories range from straight-line accelerations to circular motions, but also parabolic maneuvers and lemniscate trajectories. All the trajectories are performed for any axis combination (i.e., $x-y$, $x-z$, $y-z$) and with different speeds and accelerations. To capture the complex effects induced by aggressive flight, we push the quadrotor to its physical limits reaching speeds of $6 \mathrm{~m\,s}^{-1}$, linear accelerations of $18 \mathrm{~m\,s}^{-2}$, angular accelerations of $54 \mathrm{~rad\,s}^{-2}$, and motor speeds of $16628 \mathrm{~rpm}$. We recover unobserved accelerations from velocity measurements filtered by a UKF. Moreover, we filter the recorded attitudes and motor speeds measurements using a 4th order Butterworth low-pass filter with a cutoff frequency of 5. Finally, we randomly select 62 trajectories for training, while using the remaining 6 for testing (Figure 4).

B. Discrete vs. Continuous-Time Dynamics

We extend the analysis of discrete and continuous-time dynamics introduced in Section V-A. Specifically, we illustrate the different ground truth data that the NN models are trained on in the two different tasks. Figure 9 and Figure 10 illustrate samples of rotational velocities and accelerations that constitute our and [11] data sets. The rotational velocities are extracted from the IMU sensor and are thus affected by very little noise. Therefore, learning discrete-time dynamics is better suited for the model learning process. In the continuous-time case, the unobserved rotational accelerations are calculated as the first-order derivative of the rotational velocities. Hence, the mild noise affecting the velocities is strongly inflated by the differentiation operation. Despite the low-pass filters that can be employed to reduce the noise, directly training on such noisy labels significantly deteriorates the learning process, leading to a less stable training convergence. This is also demonstrated by the experimental results reported in Section V-A.

C. Dynamics Discretization

Robotic systems, such as quadrotors, operate in continuous time, which ideally requires a continuous dynamical model. However, in practice, a continuous model adds excessive complexity to robot control, making discretization of the dynamics necessary for numerical computation [9], [11].

There are several methods that can be used to discretize continuous-time dynamics of quadrotors, including Euler [85], Runge-Kutta [86], and Trapezoidal [87]. However, even the most advanced methods may introduce precision errors and result computationally expensive.

In this work, we conducted experiments to compare the learning performance of a NN-based dynamical model when extrapolating continuous-time or discrete-time dynamics from collected data. It is important to note that even in the case of continuous-time dynamics, the model is still discretized before being used by the controller, which involves an additional discretization step. However, by directly learning discrete-time dynamics, the discretization step is bypassed, and the NN-based dynamics can infer the discrete state at the next time step directly. This results in higher predictive performance and better control, which has been demonstrated in our experiments.

D. Beyond The Sim-to-Real Gap

Simulators are valuable tools for testing quadrotors in a safe and efficient manner. Modern simulators can replicate complex effects like drag forces, communication delays, and random disturbances. However, it is practically impossible to capture the full range of real-world variability. Quadrotor dynamics have highly nonlinear dynamics that include stochastic effects like aerodynamic forces, torques, propeller interactions, vibrations, and structural flexibility. Modeling the full range of these effects accurately is extremely challenging. Moreover, environmental factors such as wind, humidity, and temperature directly impact quadrotor flight dynamics and stability. Finally, sensor measurements from quadrotors can be affected by noise and other sources of variability, which can impact the accuracy of state estimation and control algorithms. While simulators are valuable, they cannot fully replicate all the dynamic and complex factors that can affect the real-world performance
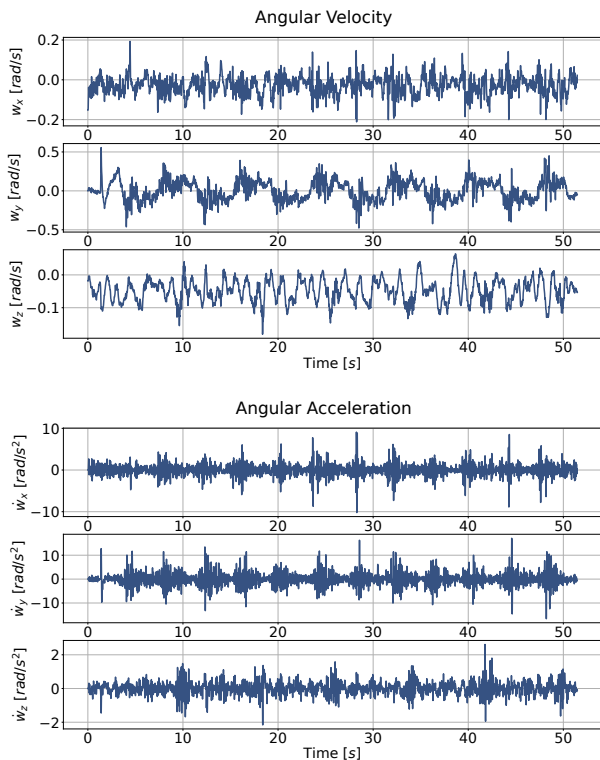
This article has been accepted for publication in IEEE Transactions on Robotics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TRO.2023.3339543

IEEE TRANSACTIONS ON ROBOTICS, 2023   17

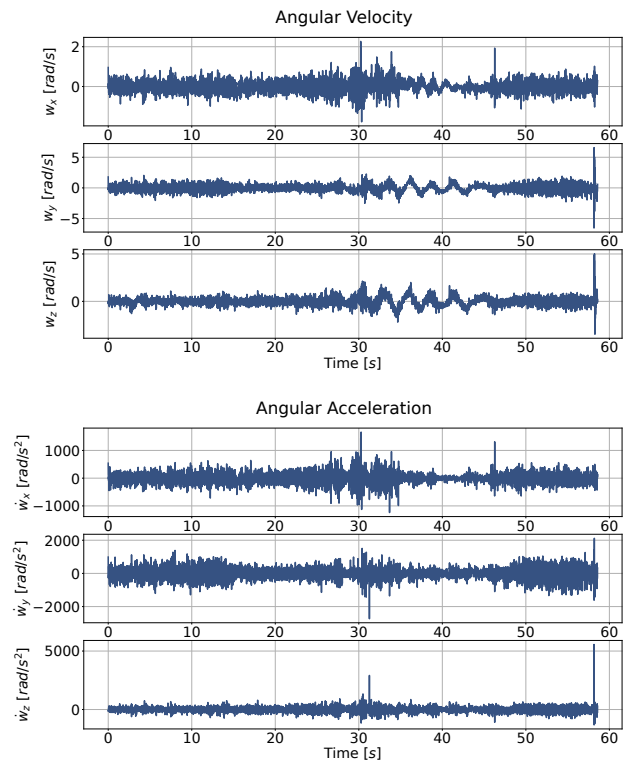Figure 9: Samples of collected rotational velocities and accelerations from [9] data set.

Figure 10: Samples of collected rotational velocities and accelerations from [11] data set.

of quadrotors. Therefore, accurately capturing the real-world dynamics of the quadrotor is essential for modeling their system accurately.

REFERENCES

[1] J. Harrison, A. Sharma, R. Calandra, and M. Pavone, "Control adaptation via meta-learning dynamics," in *NeurIPS Workshop on Meta-Learning*, 2018.

[2] K. Chua, R. Calandra, R. McAllister, and S. Levine, "Deep reinforcement learning in a handful of trials using probabilistic dynamics models," in *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[3] 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 International Conference on Robotics and Automation*, 2017, pp. 1714–1721.

[4] T. M. Moerland, J. Broekens, A. Plaat, C. M. Jonker *et al.*, "Model-based reinforcement learning: A survey," *Foundations and Trends® in Machine Learning*, vol. 16, no. 1, pp. 1–118, 2023.

[5] A. Saviolo and G. Loianno, "Learning quadrotor dynamics for precise, safe, and agile flight control," *Annual Reviews in Control*, 2023.

[6] K. Yoshida and H. Hamano, "Motion dynamics of a rover with slip-based traction model," in *IEEE International Conference on Robotics and Automation*, vol. 3, 2002, pp. 3155–3160.

[7] P.-B. Wieber, R. Tedrake, and S. Kuindersma, *Modeling and Control of Legged Robots*. Springer International Publishing, 2016, pp. 1203–1234.

[8] G. Loianno, C. Brunner, G. McGrath, and V. Kumar, "Estimation, control, and planning for aggressive flight with a small quadrotor with a single camera and imu," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 404–411, 2017.

[9] A. Saviolo, G. Li, and G. Loianno, "Physics-inspired temporal learning of quadrotor dynamics for accurate model predictive trajectory tracking," *IEEE Robotics and Automation Letters*, 2022.

[10] F. Crocetti, J. Mao, A. Saviolo, G. Costante, and G. Loianno, "Gapt: Gaussian process toolkit for online regression with application to learning quadrotor dynamics," *arXiv preprint arXiv:2303.08181*, 2023.

[11] L. Bauersfeld, E. Kaufmann, P. Foehn, S. Sun, and D. Scaramuzza, "NeuroBEM: Hybrid Aerodynamic Quadrotor Model," *Robotics: Science and Systems Foundation*, 2021.

[12] T. Duong and N. Atanasov, "Hamiltonian-based Neural ODE Networks on the SE(3) Manifold For Dynamics Learning and Control," in *Robotics: Science and Systems XVII*, 2021.

[13] A. Punjani and P. Abbeel, "Deep learning helicopter dynamics models," in *IEEE International Conference on Robotics and Automation*, 2015.

[14] S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems," *Proceedings of the national academy of sciences*, vol. 113, no. 15, pp. 3932–3937, 2016.

[15] G. Kulathunga, H. Hamed, and A. Klimchik, "Residual dynamics learning for trajectory tracking for multi-rotor aerial vehicles," *arXiv preprint arXiv:2305.15791*, 2023.

[16] C. Folkestad, S. X. Wei, and J. W. Burdick, "Koopnet: Joint learning of koopman bilinear models and function dictionaries with application to quadrotor trajectory tracking," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 1344–1350.

[17] M. Li, Z. Cai, J. Zhao, J. Wang, and Y. Wang, "Disturbance rejection and high dynamic quadrotor control based on reinforcement learning and supervised learning," *Neural Computing and Applications*, vol. 34, no. 13, pp. 11 141–11 161, 2022.

[18] C. An, S. Jia, J. Zhou, and C. Wang, "Fast model-free learning for controlling a quadrotor uav with designed error trajectory," *IEEE Access*, vol. 10, pp. 79 669–79 680, 2022.

[19] J. Fu, S. Levine, and P. Abbeel, "One-shot learning of manipulation skills with online dynamics adaptation and neural network priors," *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4019–4026, 2016.

[20] L. Wang, E. A. Theodorou, and M. Egerstedt, "Safe learning of quadrotor dynamics using barrier certificates," in *IEEE International Conference on Robotics and Automation*, 2018.

[21] M. O'Connell, G. Shi, X. Shi, K. Azizzadenesheli, A. Anandkumar, Y. Yue, and S.-J. Chung, "Neural-fly enables rapid learning for agile flight in strong winds," *Science Robotics*, vol. 7, no. 66, 2022.

[22] T. Z. Jiahao, K. Y. Chee, and M. A. Hsieh, "Online dynamics learning

for predictive control with an application to aerial robots," in *Conference on Robot Learning*. PMLR, 2023, pp. 2251–2261.

[23] A. Chakrabarty, C. Danielson, S. D. Cairano, and A. Raghunathan, "Active learning for estimating reachable sets for systems with unknown dynamics," *IEEE Transactions on Cybernetics*, vol. 52, no. 4, pp. 2531–2542, 2022.

[24] A. Capone, G. Noske, J. Umlauft, T. Beckers, A. Lederer, and S. Hirche, "Localized active learning of gaussian process state space models," in *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, ser. Proceedings of Machine Learning Research, vol. 120. PMLR, 2020, pp. 490–499.

[25] I. Abraham and T. D. Murphey, "Active learning of dynamics for data-driven control using koopman operators," *IEEE Transactions on Robotics*, vol. 35, no. 5, pp. 1071–1083, 2019.

[26] D. Cohn, Z. Ghahramani, and M. Jordan, "Active learning with statistical models," *Journal of Artificial Intelligence Research*, vol. 4, pp. 129–145, 1996.

[27] P. Ren, Y. Xiao, X. Chang, P.-Y. Huang, Z. Li, B. B. Gupta, X. Chen, and X. Wang, "A survey of deep active learning," *ACM Computing Surveys*, vol. 54, no. 9, pp. 1–40, 2021.

[28] F. Sahli Costabal, Y. Yang, P. Perdikaris, D. E. Hurtado, and E. Kuhl, "Physics-informed neural networks for cardiac activation mapping," *Frontiers in Physics*, vol. 8, 2 2020.

[29] C. Finn and S. Levine, "Deep visual foresight for planning robot motion," *IEEE International Conference on Robotics and Automation*, pp. 2786–2793, 2017.

[30] C. Doersch and A. Zisserman, "Multi-task self-supervised visual learning," in *IEEE International Conference on Computer Vision*, 2017, p. 2051–2060.

[31] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 70. PMLR, 2017, pp. 1126–1135.

[32] J. Harrison, A. Sharma, and M. Pavone, "Meta-learning priors for efficient online bayesian regression," in *Algorithmic Foundations of Robotics XIII*. Springer, 2020.

[33] S. Banerjee, J. Harrison, P. M. Furlong, and M. Pavone, "Adaptive meta-learning for identification of rover-terrain dynamics," *arXiv preprint arXiv:2009.10191*, 2020.

[34] B. Evans, A. Thankaraj, and L. Pinto, "Context is everything: Implicit identification for dynamics adaptation," in *International Conference on Robotics and Automation*, 2022, pp. 2642–2648.

[35] A. Nagabandi, I. Clavera, S. Liu, R. S. Fearing, P. Abbeel, S. Levine, and C. Finn, "Learning to adapt in dynamic, real-world environments through meta-reinforcement learning," in *International Conference on Learning Representations*, 2019.

[36] R. Kaushik, T. Anne, and J.-B. Mouret, "Fast online adaptation in robotics through meta-learning embeddings of simulated priors," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 5269–5276.

[37] S. Belkhale, R. Li, G. Kahn, R. McAllister, R. Calandra, and S. Levine, "Model-based meta-reinforcement learning for flight with suspended payloads," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1471–1478, 2021.

[38] S. Bechtle, B. Hammoud, A. Rai, F. Meier, and L. Righetti, "Leveraging forward model prediction error for learning control," in *IEEE International Conference on Robotics and Automation*, 2021.

[39] L. Hewing, J. Kabzan, and M. N. Zeilinger, "Cautious model predictive control using gaussian process regression," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 6, pp. 2736–2743, 2020.

[40] S. Ambikasaran, D. Foreman-Mackey, L. Greengard, D. W. Hogg, and M. O'Neil, "Fast direct methods for gaussian processes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 2, pp. 252–265, 2016.

[41] J. Quiñonero-Candela and C. E. Rasmussen, "A unifying view of sparse approximate gaussian process regression," *Journal of Machine Learning Research*, vol. 6, no. 65, pp. 1939–1959, 2005.

[42] S. Das, S. Roy, and R. Sambasivan, "Fast gaussian process regression for big data," *Big Data Research*, vol. 14, pp. 12–26, 2018.

[43] G. Torrente, E. Kaufmann, P. Fohn, and D. Scaramuzza, "Data-Driven MPC for Quadrotors," *IEEE Robotics and Automation Letters*, 2021.

[44] S. Sarkka, A. Solin, and J. Hartikainen, "Spatiotemporal learning via infinite-dimensional bayesian filtering and smoothing: A look at gaussian process regression through kalman filtering," *IEEE Signal Processing Magazine*, vol. 30, no. 4, pp. 51–61, 2013.

[45] G. Joshi, J. Virdi, and G. Chowdhary, "Asynchronous deep model reference adaptive control," in *Conference on Robot Learning*, vol. 155, 2021, pp. 984–1000.

[46] J. Mao, J. Yeom, S. Nair, and G. Loianno, "From propeller damage estimation and adaptation to fault tolerant control: Enhancing quadrotor resilience," *arXiv preprint arXiv:2310.13091*, 2023.

[47] M. Labbadi and M. Cherkaoui, "Robust adaptive nonsingular fast terminal sliding-mode tracking control for an uncertain quadrotor uav subjected to disturbances," *ISA Transactions*, vol. 99, 2019.

[48] Y. Zou and B. Zhu, "Adaptive trajectory tracking controller for quadrotor systems subject to parametric uncertainties," *Journal of the Franklin Institute*, vol. 354, no. 15, pp. 6724–6746, 2017.

[49] A. Gahlawat, P. Zhao, A. Patterson, N. Hovakimyan, and E. Theodorou, "L1-gp: L1 adaptive control with bayesian learning," in *Conference on Learning for Dynamics and Control*, vol. 120, 2020, pp. 826–837.

[50] R. Ortega and E. Panteley, "L1–"adaptive" control always converges to a linear pi control and does not perform better than the pi," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 6926–6928, 2014.

[51] J. Pravitra, K. A. Ackerman, C. Cao, N. Hovakimyan, and E. A. Theodorou, "L1-adaptive mppi architecture for robust and agile control of multirotors," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 7661–7666, 2020.

[52] T. Lew, A. Sharma, J. Harrison, A. Bylard, and M. Pavone, "Safe active dynamics learning and control: A sequential exploration–exploitation framework," *IEEE Transactions on Robotics*, pp. 1–20, 2022.

[53] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *International Conference on Machine Learning*, 2016.

[54] M. Valdenegro-Toro, "Deep sub-ensembles for fast uncertainty estimation in image classification," *arXiv preprint arXiv:1910.08168*, 2019.

[55] J. Gawlikowski, C. Tassi, M. Ali, J. Lee, M. Humt, J. Feng, A. Kruspe, R. Triebel, P. Jung, R. Roscher, M. Shahzad, W. Yang, R. Bamler, and X. Zhu, "A survey of uncertainty in deep neural networks," *arXiv preprint arXiv:2107.03342*, 2021.

[56] D. Ebeigbe, T. Berry, M. Norton, A. Whalen, D. Simon, T. Sauer, and S. Schiff, "A generalized unscented transformation for probability distributions," *arXiv preprint arXiv:2104.01958*, 2021.

[57] K. E. Atkinson, *An Introduction to Numerical Analysis*, 2nd ed. John Wiley & Sons, 1989.

[58] C. Rucker, "Integrating rotations using nonunit quaternions," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 2979–2986, 2018.

[59] G. Li, A. Tunchez, and G. Loianno, "Learning model predictive control for quadrotors," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 5872–5878.

[60] H. E. Robbins, "A stochastic approximation method," *Annals of Mathematical Statistics*, vol. 22, pp. 400–407, 2007.

[61] S. L. Smith, E. Elsen, and S. De, "On the generalization benefit of noise in stochastic gradient descent," in *International Conference on Machine Learning*, 2020.

[62] E. Goderer, "A quaternion-based unscented kalman filter for orientation tracking," *International Conference on Information Fusion*, 2003.

[63] V. Wüest, V. Kumar, and G. Loianno, "Online estimation of geometric and inertia parameters for multirotor aerial vehicles," in *IEEE International Conference on Robotics and Automation*, 2019.

[64] E. Wan and R. Merwe, "The unscented kalman filter for nonlinear estimation," in *IEEE Adaptive Systems for Signal Processing, Communications, and Control Symposium*, 2000.

[65] J. Mao, G. Li, S. Nogar, C. Kroninger, and G. Loianno, "Aggressive visual perching with quadrotors on inclined surfaces," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2021, pp. 5242–5248.

[66] S. Sun, A. Romero, P. Foehn, E. Kaufmann, and D. Scaramuzza, "A comparative study of nonlinear mpc and differential-flatness-based control for quadrotor agile flight," *IEEE Transactions on Robotics*, pp. 1–17, 2022.

[67] H. Nguyen, M. Kamel, K. Alexis, and R. Siegwart, "Model predictive control for micro aerial vehicles: A survey," in *European Control Conference*, 2021, pp. 1556–1563.

[68] D. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *arXiv preprint arXiv:1511.07289*, 2016.

[69] M. Diehl, H. G. Bock, J. P. Schlöder, R. Findeisen, Z. Nagy, and F. Allgöwer, "Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations," *Journal of Process Control*, vol. 12, no. 4, pp. 577–585, 2002.

[70] R. Verschueren, G. Frison, D. Kouzoupis, J. Frey, N. van Duijkeren, A. Zanelli, B. Novoselnik, T. Albin, R. Quirynen, and M. Diehl, "acados – a modular open-source framework for fast embedded optimal control," *Mathematical Programming Computation*, 2021.

[71] G. Frison and M. Diehl, "HPIPM: a high-performance quadratic programming framework for model predictive control," *IFAC*, 2020.

[72] G. Frison, D. Kouzoupis, T. Sartor, A. Zanelli, and M. Diehl, "BLASFEO: Basic linear algebra subroutines for embedded optimization," *ACM Transactions on Mathematical Software*, 2018.

[73] J. Andersson, J. Gillis, G. Horn, J. Rawlings, and M. Diehl, "Casadi: a software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, 2018.

[74] Z. Wu, S. Cheng, K. A. Ackerman, A. Gahlawat, A. Lakshmanan, P. Zhao, and N. Hovakimyan, "L1 adaptive augmentation for geometric tracking control of quadrotors," *IEEE International Conference on Robotics and Automation*, 2022.

[75] P.-J. Bristeau, P. Martin, E. Salaün, and N. Petit, "The role of propeller aerodynamics in the model of a quadrotor uav," in *European Control Conference*, 2009, pp. 683–688.

[76] P. Foehn, E. Kaufmann, A. Romero, R. Penicka, S. Sun, L. Bauersfeld, T. Laengle, G. Cioffi, Y. Song, A. Loquercio *et al.*, "Agilicious: Open-source and open-hardware agile quadrotor for vision-based flight," *Science robotics*, vol. 7, no. 67, p. eabl6259, 2022.

[77] A. Saviolo, P. Rao, V. Radhakrishnan, J. Xiao, and G. Loianno, "Unifying foundation models with quadrotor control for visual tracking beyond object categories," *arXiv preprint arXiv:2310.04781*, 2023.

[78] 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*, vol. 29, no. 3, pp. 1203–1218, 2020.

[79] A. Markov and N. Nagorny, "The theory of algorithms," in *The Theory of Algorithms*. Springer Dordrecht, 1954.

[80] H. Song, M. Kim, D. Park, Y. Shin, and J.-G. Lee, "Learning from noisy labels with deep neural networks: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[81] A. Pinosky, I. Abraham, A. Broad, B. Argall, and T. D. Murphey, "Hybrid control for combining model-based and model-free reinforcement learning," *The International Journal of Robotics Research*, p. 027836492210833331, 2022.

[82] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 7559–7566.

[83] Y. Chebotar, M. Kalakrishnan, A. Yahya, A. Li, S. Schaal, and S. Levine, "Path integral guided policy search," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 3381–3388.

[84] V. Peretroukhin, M. Giamou, D. M. Rosen, W. N. Greene, N. Roy, and J. Kelly, "A smooth representation of belief over so (3) for deep rotation learning with uncertainty," *arXiv preprint arXiv:2006.01031*, 2020.

[85] G. Hahn, "A modified euler method for dynamic analyses," *International Journal for Numerical Methods in Engineering*, vol. 32, no. 5, pp. 943–955, 1991.

[86] J. H. Cartwright and O. Piro, "The dynamics of runge–kutta methods," *International Journal of Bifurcation and Chaos*, vol. 2, no. 03, pp. 427–449, 1992.

[87] P. C. Hammer and J. W. Hollingsworth, "Trapezoidal methods of approximating solutions of differential equations," *Mathematical Tables and Other Aids to Computation*, pp. 92–96, 1955.
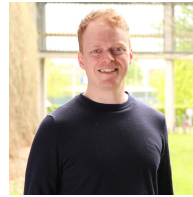
Jonathan Frey studied Mathematics at the Technical University Ilmenau and at University Freiburg, where he obtained his M.Sc. in 2019. He is currently a PhD student at University Freiburg working on numerical algorithms and advantageous problem formulations for fast MPC applications under the supervision of Prof. Dr. Moritz Diehl. He is currently the maintainer of the open-source software framework "acados" which implements fast and embedded solvers for nonlinear optimal control.

Abhishek Rathod is a Robotics Engineer at the National Robotics Engineering Center, working in the areas of control and planning for autonomous aerial systems. He earned his Master's in Robotics from New York University in 2022 and Bachelor's degree in Mechanical Engineering from the University of Idaho in 2020. During and following his Master's degree, he worked as a researcher at the Agile Robotics and Perception Lab, focusing on data-driven model-based control and safety-critical control with control barrier functions.

Moritz Diehl graduated with a Diploma in physics and mathematics at Heidelberg University, Heidelberg, Germany and Cambridge University, Cambridge, U.K. in 1999, and received his Ph.D. degree from Heidelberg University in 2001. From 2006 to 2013, he was a Professor with the Department of Electrical Engineering, KU Leuven, Leuven, Belgium. In 2013, he moved to the University of Freiburg, Freiburg, Germany, where he is currently the Head of the Systems Control and Optimization Laboratory, Department of Microsystems Engineering, and the Department of Mathematics.

Alessandro Saviolo is a Ph.D. candidate at the Agile Robotics and Perception Lab at New York University, focusing on the advancement of AI-driven robotics, particularly in the realm of perpetual, resilient autonomy. He earned his Master's in Computer Engineering from the University of Padova, and his graduate research spanned across National Chao Tung University in Taiwan and the University of Zurich in Switzerland. Following his Master's degree, he worked as a software engineer at Flexsight, a start-up specializing in autonomous perception solutions.

Giuseppe Loianno received the Ph.D. degree in robotics from the University of Naples "Federico II," Naples, Italy, in 2014. He is currently an Assistant Professor with New York University, New York, NY, USA, and the Director of the Agile Robotics and Perception Lab working on autonomous robots. Prior to joining NYU, he was a Postdoctoral Researcher, Research Scientist, and Team Leader with GRASP Lab, University of Pennsylvania, Philadelphia, PA, USA. He has authored or coauthored more than 70 conference papers, journal papers, and book chapters. His research interests include perception, learning, and control for autonomous robots. Dr. Loianno was the recipient of the NSF CAREER Award in 2022, DARPA Young Faculty Award in 2022, IROS Toshio Fukuda Young Professional Award in 2022, Conference Editorial Board Best Associate Editor Award at ICRA 2022, and Best Reviewer Award at ICRA 2016, and he was selected as Rising Star in AI from KAUST in 2023. He is also the Co-Chair of the IEEE RAS Technical Committee on Aerial Robotics and Unmanned Aerial Vehicles. He was the General Chair of the IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR) in 2021 as well as Program Chair in 2019, 2020, and 2022. His work has been featured in a large number of renowned international news and magazines.