NeuroBEM: Hybrid Aerodynamic Quadrotor Model

Leonard Bauersfeld*, Elia Kaufmann*, Philipp Foehn, Sihao Sun, Davide Scaramuzza
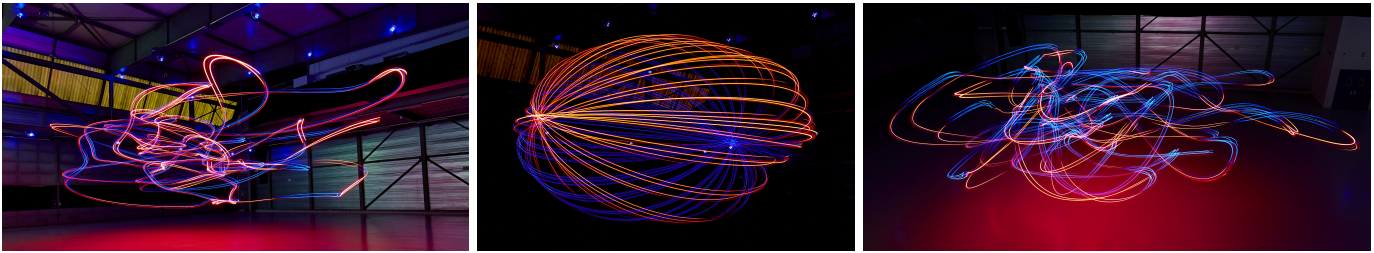

Fig. 1: Long-exposure images depicting quadrotor trajectory tracking at speeds up to 65 km/h in a large-scale motion-capture system. The captured data is used to fit a hybrid quadrotor model combining blade-element-momentum (BEM) theory with a neural network compensating residual dynamics. This hybrid model reproduces the flown trajectories in simulation with a positional RMSE error reduction of over 50% compared to state-of-the-art.

Abstract—**Quadrotors are extremely agile, so much in fact, that classic first-principle-models come to their limits. Aerodynamic effects, while insignificant at low speeds, become the dominant model defect during high speeds or agile maneuvers. Accurate modeling is needed to design robust high-performance control systems and enable flying close to the platform's physical limits. We propose a hybrid approach fusing first principles and learning to model quadrotors and their aerodynamic effects with unprecedented accuracy. First principles fail to capture such aerodynamic effects, rendering traditional approaches inaccurate when used for simulation or controller tuning. Data-driven approaches try to capture aerodynamic effects with blackbox modeling, such as neural networks; however, they struggle to robustly generalize to arbitrary flight conditions. Our hybrid approach unifies *and outperforms* both first-principles blade-element momentum theory and learned residual dynamics. It is evaluated in one of the world's largest motion-capture systems, using autonomous-quadrotor-flight data at speeds up to 65 km/h. The resulting model captures the aerodynamic thrust, torques, and parasitic effects with astonishing accuracy, outperforming existing models with 50% reduced prediction errors, and shows strong generalization capabilities beyond the training set.**

Supplementary Material

A narrated video illustrating our approach is available at https://youtu.be/Nze1wlfmzTQ. Code and dataset can be found at http://rpg.ifi.uzh.ch/NeuroBEM.html.

I. Introduction

In recent years, research on fast navigation of autonomous quadrotors has made tremendous progress, continually pushing the vehicles to more aggressive maneuvers [1–4] (Figure 1). To further advance the field, several competitions have been organized, such as the autonomous drone race series at the recent IROS and NeurIPS conferences [5, 6] and the AlphaPilot challenge [7]. In the near future, estimation and control algorithms will reach the level of maturity necessary to push autonomous quadrotors to the bounds of what is physically possible. This presents the need for quadrotor models that can predict the behaviour of the platform even during highly aggressive maneuvers.

* Equal contribution
The authors are with the Robotics and Perception Group, Dep. of Informatics, University of Zurich, and Dep. of Neuroinformatics, University of Zurich and ETH Zurich, Switzerland (http://rpg.ifi.uzh.ch).
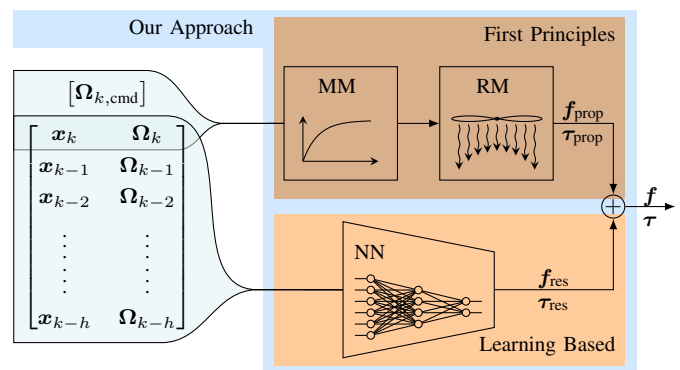

Fig. 2: Overview of the proposed architecture to predict aerodynamic forces and torques. The physical modeling pipeline (upper part) consists of a motor model (MM) and a rotor model (RM)—detailed in Sections III-C and III-D. It takes the current state x_k, current motor speeds Ω_k, and the motor speed command $\Omega_{k,\text{cmd}}$ as an input. Combined with the estimate of the residual forces and torques predicted by the neural network (NN) using the current and past h states, the acting force f and torque τ are calculated.

Accurately modeling quadrotors flying at their physical limits is extremely challenging and requires to capture complex effects due to aerodynamic forces, motor dynamics, and vibrations. Especially aerodynamic forces pose a challenge, as they depend on hidden state variables like airflow, which cannot be easily measured. Furthermore, the individual downwash induced by the rotors interacts with both the frame and the blades depending on the current state of the platform. The repeatability of tracking errors observed in prior work [1, 8, 9] and in this work when performing aggressive maneuvers suggests that the difficulty of learning quadrotor dynamics is not caused by stochasticity in the dynamics, but rather by unobserved state variables such as airflow.

Traditional approaches to quadrotor modeling limit the captured effects to simple linear drag approximations and quadratic thrust curves [10–12]. Such approximations are computationally efficient and describe the platform well in low-speed regimes, but exhibit increasing bias at higher velocities as they neglect the influence of the inflow velocity on the generated thrust. More elaborate models based on blade-element-momentum (BEM) theory manage to accurately model single rotors at high wind velocities, but they do not

account for the aerodynamic interactions between rotors and the frame. Parametric gray-box models [13] aim to overcome these limitations by describing the forces and torques as a linear combination of library functions. While these models can perform well, their performance hinges on the appropriate choice of basis functions, which require human expert knowledge to design. Recent research has investigated computational fluid dynamics [14] to model the aerodynamic effects at play during different flight conditions. While being very accurate, such approaches are computationally expensive and need hours of processing on a compute cluster, rendering them impractical for experiments spanning more than a few seconds.

Accurately predicting forces acting on the quadrotor at high speeds requires to implicitly estimate the airflow around the vehicle. Although this state variable cannot be directly observed, it can be deduced from a sequence of measurements of other observable state variables. Thus, learning a high-order dynamics model requires a method for regression of a nonlinear function in a high-dimensional input space. Deep neural networks have shown to excel at such high-dimensional regression tasks and have already been applied to dynamic system modeling [8, 9, 15–17]. Despite showing promising performance, such purely-learned models require large amounts of data and require careful regularization to avoid overfitting.

Contribution

This work proposes a quadrotor dynamics model that can accurately capture complex aerodynamic effects by *combining* a state-of-the-art rotor model based on BEM theory with learned residual force and torque terms represented by a deep neural network. The resulting hybrid model benefits from the expressive power of deep neural networks and the generalizability of first-principles modeling. The latter reduces the need for extreme amounts of training data. The model is identified using data collected from a large set of maneuvers performed on a real quadrotor platform. Leveraging one of the biggest optical tracking volumes in the world, the platform's state as well as the motor speeds are recorded during flight. The resulting dataset contains 96 flights with a cumulative time of 1h 15min and 1.8 million data points, covering the entire performance envelope of the platform up to observed speeds of $65 \, \mathrm{km \, h^{-1}}$ ($18 \, \mathrm{m \, s^{-1}}$) and accelerations of $46.8 \, \mathrm{m \, s^{-2}}$.

The proposed model is compared against state-of-the-art modeling approaches on unseen test maneuvers. The comparison is done in terms of both evaluation of predicted aerodynamic forces and torques and closed-loop integration of the model in a simulator, each evaluated against real-world reference data. In both categories, a performance increase by a factor of two is observed.

II. RELATED WORK

Traditionally, a rotor is assumed to produce thrust and axial torque proportional to the square of its angular rate with a constant coefficient [18], which is referred to hereinafter as the simple quadratic model. While these assumptions are valid for rotors on a static thrust stand and for near-hover

flight, they do neither account for the case where the rotors move through air, nor for rotor-to-rotor and rotor-to-body interactions. Nevertheless, due to its simplicity, this model is still used in well-known aerial robotics simulators such as AirSim [11], Flightmare [19], RotorS [10] and others [20]. To improve the accuracy of the thrust model in non-stationary flights over the simple quadratic model, momentum theory has been used in [21–23]. Blade element theory is another approach to model a single rotor more accurately. The forces and torques acting on each infinitesimal portion of the blade are integrated over the whole propeller [24]. This theory has been adopted to model aerodynamic effects on a quadrotor in many studies [25–29]. However, both blade-element theory and momentum theory require the value of the induced velocity which is challenging to estimate. Hence, the blade-element-momentum (BEM) theory is proposed, which combines the above two theories to alleviate the difficulty of calculating the induced velocity. The resulting model can accurately capture aerodynamic forces and torques acting on single rotors in a wide range of operating conditions [30–32].

Even though BEM outperforms simple quadratic models and often achieves accurate predictions, it does not account for any interaction between the flow tubes of different propellers or the frame [13]. Previous work has incorporated interaction effects using either static wind tunnel tests [33–35] where the vehicle is rigidly mounted on a force sensor, or by performing fast maneuvers in instrumented tracking volumes [36]. In [36] a simple quadratic model is combined with residual forces predicted by Gaussian Processes. While this approach offers a lightweight solution to learn residual forces and can be used for control, it does not model residual torques, effectively neglecting moments caused by rotor-to-rotor interactions. In [13], the quadrotor platform is identified using a gray-box model that uses a library of polynomials as basis functions and is able to model both aerodynamic forces and torques. This method relies on the predefined function library and also contains discontinuities in the learned model. Another line of works investigates the modeling of quadrotors using computational fluid dynamics (CFD) [14, 37]. While such simulations achieve results that are highly accurate and manage to capture real-world effects well, they require large amounts of computation time on high-performance compute clusters.

Due to their ability to identify patterns in large amounts of data, deep neural networks represent a promising approach to model aerodynamic effects precisely and computationally efficient. A recent line of works employs deep neural networks to learn quadrotor dynamics model purely from data, for both continuous time formulations [8, 9] as well as discrete-time formulations [15, 16, 38, 39]. While approaches relying entirely on learning-based methods have high representative power and the potential to also learn complex interaction effects, they require large amounts of data to train and careful regularization to avoid overfitting.

The approach presented in this work is inspired by [8, 9], but instead of learning the full dynamics, it combines state-of-the-art BEM modeling based on first principles with a data-
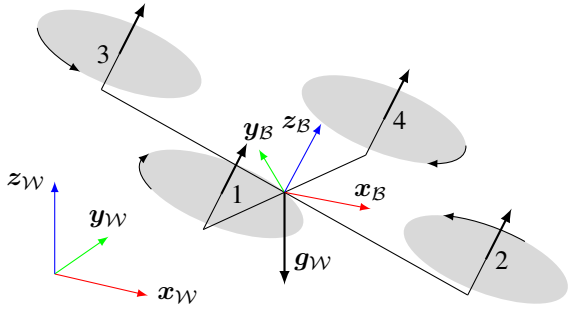
Fig. 3: Diagram of the quadrotor model depicting the world and body frames and illustrating the propeller numbering convention.

driven approach to learn the residual force and torque terms. The resulting model benefits from the strong generalization performance of traditional first-principle modeling and the flexibility of learning-based function approximation.

III. QUADROTOR MODEL

This section explains the hybrid quadrotor model proposed in this work. It starts by introducing the notation and the rigid body dynamics (Figure 3), proceeds to explaining two approaches to single-rotor modeling of increasing complexity and concludes with the learned residual model. The hybrid structure of the model, illustrated in Figure 2, consists of a *rotor model* and a learned correction.

A. Notation

Scalars are denoted in non-bold $[s, S]$, vectors in lowercase bold v, and matrices in uppercase bold M. World $\mathcal{W}$ and Body $\mathcal{B}$ frames are defined with orthonormal basis i.e. $\{x_{\mathcal{W}}, y_{\mathcal{W}}, z_{\mathcal{W}}\}$. The frame $\mathcal{B}$ is located at the center of mass of the quadrotor. A vector from coordinate p_1 to p_2 expressed in the $\mathcal{W}$ frame is written as: $_{\mathcal{W}}v_{12}$. If the vector's origin coincides with the frame it is described in, the frame index is dropped, e.g. the quadrotor position is denoted as $p_{\mathcal{W}\mathcal{B}}$. Furthermore, unit quaternions $q = (q_w, q_x, q_y, q_z)$ with $\|q\| = 1$ are used to represent orientations, such as the attitude state of the quadrotor body $q_{\mathcal{W}\mathcal{B}}$. Finally, full SE3 transformations, such as changing the frame of reference from body to world for a point p_{B1}, can be described by $_{\mathcal{W}}p_{B1} = _{\mathcal{W}}t_{\mathcal{W}\mathcal{B}} + q_{\mathcal{W}\mathcal{B}} \odot p_{B1}$. Note the quaternion-vector product is denoted by $\odot$ representing a rotation of the vector by the quaternion as in $q \odot v = qv\bar{q}$, where $\bar{q}$ is the quaternion's conjugate.

B. Quadrotor Dynamics

The quadrotor is assumed to be a 6 degree-of-freedom rigid body of mass m and diagonal moment of inertia matrix $J = \text{diag}(J_x, J_y, J_z)$. The state space is thus 13-dimensional and its dynamics can be written as:

$$\dot{x} = \begin{bmatrix} \dot{p}_{\mathcal{W}\mathcal{B}} \\ \dot{q}_{\mathcal{W}\mathcal{B}} \\ \dot{v}_{\mathcal{W}\mathcal{B}} \\ \dot{\omega}_{\mathcal{B}} \end{bmatrix} = \begin{bmatrix} v_{\mathcal{W}} \\ q_{\mathcal{W}\mathcal{B}} \cdot \begin{bmatrix} 0 \\ \omega_{\mathcal{B}}/2 \end{bmatrix} \\ \frac{1}{m}\left(q_{\mathcal{W}\mathcal{B}} \odot \left(\underbrace{f_{\text{prop}} + f_{\text{res}}}_{:=f}\right)\right) + g_{\mathcal{W}} \\ J^{-1}\left(\underbrace{\tau_{\text{prop}} + \tau_{\text{res}}}_{:=\tau} - \omega_{\mathcal{B}} \times J\omega_{\mathcal{B}}\right) \end{bmatrix}, \quad (1)$$

where $g_{\mathcal{W}} = [0, 0, -9.81\,\text{m/s}^2]^{\mathsf{T}}$ denotes earth's gravity, f_{prop} is the collective force produced by the propellers including any parasitic effects the rotor model can simulate (e.g. induced drag), and f_{res} denotes residual forces that are not explained by the rotor model used. Similarly, τ_{prop} and τ_{res} are the cumulative torques acting on the platform due to the propellers and residual torques that are not explained by the rotor model.

$$f_{\text{prop}} = \sum_i f_i \quad (2)$$

$$\tau_{\text{prop}} = \sum_i \tau_i + r_{\text{P},i} \times f_i, \quad (3)$$

where $r_{\text{P},i}$ is the location of propeller i expressed in the body frame and f_i, τ_i are the forces and torques generated by the i-th propeller. The rotor models aim at predicting accurate estimates of the single-rotor forces and torques f_i, τ_i, as explained in the following sections. The force and torque effects of the fuselage, body, and rotor interaction are not explicitly modeled, but should be captured by the residual dynamics f_{res} and τ_{res}, predicted by a neural network.

C. Rotor Model: Quadratic

The simplest model for single propeller is a quadratic fit which assumes the thrust and torque produced by a single propeller to be proportional to the square of its rotational rate (propeller speed) Ω.

$$f_i(\Omega) = \begin{bmatrix} 0 \\ 0 \\ c_{\text{l,q}} \cdot \Omega^2 \end{bmatrix} \quad \tau_i(\Omega) = \begin{bmatrix} 0 \\ 0 \\ c_{\text{d,q}} \cdot \Omega^2 \end{bmatrix} \quad (4)$$

The coefficients $c_{\text{l,q}}$ and $c_{\text{d,q}}$ are typically identified using a static propeller test stand. This simplified model is a good approximation for near-hover flight at near-zero velocity without ceiling or ground effects [29] and explains static thrust-test stand measurements very well. However, it ignores that egomotion impacts the lift generated by the propeller. The induced drag, which depends on the propeller speed and body-relative air velocity, is neglected as well, albeit being the dominant source of drag for quadrotors. This is sometimes mitigated by combining the model with a linear drag term such as in [10, 12].

D. Rotor Model: BEM

Compared to the quadratic model, Blade-Element-Momentum-Theory (BEM) accounts for the effects of varying relative air speed on the rotor thrust. It assumes interaction effects between individual rotors to be negligible and describes each rotor separately. The approach presented here is based on classical propeller modeling for helicopters [24]. The modeling of the propeller lift and drag coefficients is based on [31, 40].

First, basic momentum theory is introduced and used to relate the thrust force to a given velocity difference in a flow-tube across the rotor. Then the aerodynamic blade-element model is presented. While momentum theory uses the momentum conservation to relate induced airspeed and generated thrust, a blade-element model sums the contributions of infinitesimal

blade elements to the total thrust force and drag torque. Finally, the full algorithm combining momentum theory and the blade-element model is presented.

For now, the induced velocity v_i is considered to be known as momentum theory and the blade element model jointly yield this information. Together with the known ego-motion of the quadcopter, this fully determines the wind field around the individual propellers.

Momentum Theory. The most simple theory for analyzing rotors is momentum theory as it allows to calculate the thrust of a propeller based on a momentum balance across the rotor. This balance is done inside a flow-tube with radius R that fully contains the propeller. Assuming a known and constant induced velocity v_i across the diameter of the flow tube, the thrust T of a rotor is given by [24]:

$$T = 2v_i \rho A \sqrt{v_\text{hor}^2 + (v_\text{ver} - v_i)^2} \, , \tag{5}$$

where v_hor and v_ver denote the horizontal and vertical velocity component of the flow tube. Note that momentum theory alone does not provide any means for calculating the induced velocity, it merely relates the induced velocity and the thrust based on a momentum balance and does not make any assumption on the physical process that actually accelerates the air.

Blade Element Theory. The main purpose of a blade element model is to estimate the acting forces and torques accurately. A propeller consists of b identical blades (typically $b = 2$ or $b = 3$) attached to the rotor hub, each acting as a wing producing lift and drag forces. The propeller coordinate frame $\mathcal{P}$ shown in Figure 4 is defined such that the $z_\mathcal{P}$-axis points down and the $x_\mathcal{P}$-axis opposes the horizontal component v_hor of the incoming wind.

The finite stiffness of the blade and its hub-mount allow it to bend and deform, transmitting forces and introducing torques around the hub. It also causes the rotor-disk plane to be tilted with respect to the propeller coordinate system. This deformation can be split into a symmetric *coning* component a_0 due to the overall lift produced by the propeller (not shown in Figure 4) and an asymmetric *flapping* component that depends on the azimuth angle Ψ of the propeller. Figure 4 illustrates this: in forward flight one side of the propeller experiences a higher relative airspeed (advancing blade) compared to the opposite side (retracting blade). Thus, there is a lift imbalance between the sides of the propeller which in turn causes the elastic propeller to bend upwards on the advancing side. This is called lateral flapping with the associated flapping angle b_1. Due to the inertia of the blade, the lateral flapping also induces a longitudinal flapping angle a_1.

Figure 5 shows the velocities, angles and forces of a blade element. The chord of the airfoil is rotated by an angle θ relative to the xy-plane. Together with the inflow angle φ, this results in a total angle of attack $\alpha = \theta + \varphi$. Each blade element produces a lift force $\mathrm{d}L$ perpendicular to the incoming airstream and a drag force $\mathrm{d}D$ in the direction of the incoming airflow. The thrust force $\mathrm{d}T$ and horizontal force $\mathrm{d}H$ are aligned with the propeller coordinate frame.
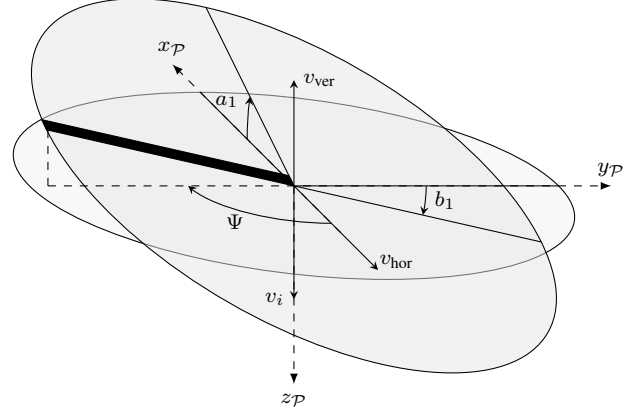

Fig. 4: Lateral flapping b_1 and longitudinal flapping a_1 occur due to lift imbalance. The azimuth angle Ψ of the blade is measured 'from the tail' in the direction of rotation. The horizontal velocity v_hor and vertical velocity v_ver are defined opposite to the propeller frame, i.e. if the propeller moves along $z_\mathcal{P}$ the relative velocity will also have a positive vertical component. Note that coning is not shown to improve the clarity of the schematic.
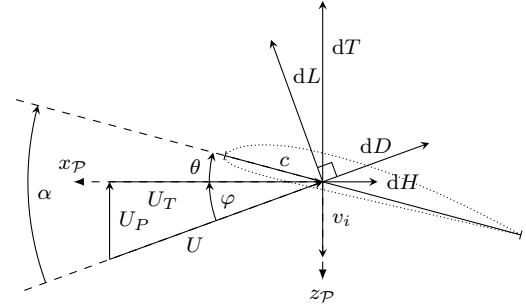

Fig. 5: A blade element located at radius r and azimuth angle Ψ.

The tangential velocity U_T and parallel-to-motor velocity U_P are related to the angular velocity Ω of the propeller as

$$U_T(r, \Psi) = \Omega r + v_\text{hor} \sin \Psi \tag{6}$$

$$\begin{aligned} U_P(r, \Psi) = & \, v_\text{ver} - v_i \\ & - r\Omega(a_1 \sin \Psi + b_1 \cos \Psi) \\ & + v_\text{ver}(a_0 - a_1 \cos \Psi - b_1 \sin \Psi) \cos \Psi \, . \end{aligned} \tag{7}$$

The local angle of attack α can be calculated as

$$\varphi(r, \Psi) = \arctan(U_P(r, \Psi)/U_T(r, \Psi)) \tag{8}$$

$$\alpha(r, \Psi) = \theta_0 + \frac{r}{R}\theta_1 + \varphi(r, \Psi) \, , \tag{9}$$

where θ_0 is the pitch angle of the blade and θ_1 the blade twist. The differential lift force $\mathrm{d}L$ and the differential drag $\mathrm{d}D$ can be expressed as functions of radius r and azimuth angle Ψ:

$$\mathrm{d}L(r, \Psi) = c(r)c_l(\alpha(r, \Psi))(U_T(r, \Psi)^2 + U_P(r, \Psi)^2) \tag{10}$$

$$\mathrm{d}D(r, \Psi) = c(r)c_d(\alpha(r, \Psi))(U_T(r, \Psi)^2 + U_P(r, \Psi)^2), \tag{11}$$

where $c(r)$ is the chord length and $c_l(\alpha)$, $c_d(\alpha)$ are the angle-of-attack dependent coefficients of lift and drag respectively. The coefficients are modeled as proposed in [31, 40] as

$$c_d(\alpha) = c_{d,0} \sin^2 \alpha \qquad c_l(\alpha) = c_{l,0} \sin \alpha \cos \alpha \, , \tag{12}$$

where $c_{d,0}$ and $c_{l,0}$ are experimentally determined by measuring the lift and drag torque on a thrust test stand.
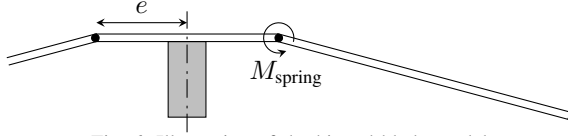
Fig. 6: Illustration of the hinged blade model.

The overall thrust T, horizontal force H and drag torque Q are obtained through integration.

$$T = \frac{b\rho}{4\pi} \int_0^R \int_0^{2\pi} dL \cos\phi + dD \sin\phi \; \mathrm{d}\Psi \; \mathrm{d}r \qquad (13)$$

$$H = \frac{b\rho}{4\pi} \int_0^R \int_0^{2\pi} (-dL \sin\phi + dD \cos\phi) \sin\Psi \; \mathrm{d}\Psi \; \mathrm{d}r \qquad (14)$$

$$Q = \frac{b\rho}{4\pi} \int_0^R \int_0^{2\pi} (-dL \sin\phi + dD \cos\phi)\, r \; \mathrm{d}\Psi \; \mathrm{d}r \qquad (15)$$

Blade Elasticity. Standard helicopters have their rotor blades connected to the rotor hub through a hinge pin, optionally with an offset. This is not true for small rotor sizes typically found on multicopters, since the blade is fixed and elastic which breaks the assumptions made in standard helicopter literature. Therefore, the model is slightly adapted to capture the characteristics of small propellers: a blade is rigid and connected to the rotor hub with a hinge *and a torsional spring* at an offset e [21] as shown in Figure 6. The coning angle a_0 as well as the flapping angles a_1 and b_1 can be calculated by equating the moments acting on the rotor hub. At the hinge position, the following moment equilibrium occurs:

$$0 = M_{\mathrm{w}} + M_{\mathrm{gyro}} + M_{\mathrm{inertial}} + M_{\mathrm{cf}} + M_{\mathrm{aero}} + M_{\mathrm{spring}} \;, \qquad (16)$$

where the moment M_{w} is caused by the weight of the blade, the moment M_{gyro} is due to gyroscopic effects the blade experiences when a non-zero rollrate or pitchrate are present, M_{inertial} comes from the inertia of the blade and its angular acceleration during the flapping motion, M_{cf} is caused by centrifugal forces when the blade flaps, the moment M_{aero} is a result of the lift generated by the blade, and lastly M_{spring} is the restoring moment produced by the hinge spring. For brevity, the derivation of the coning and flapping angles are omitted here. They closely follow [24] (pp. 463). Due to the torsional spring at the hinge, the spring moment needs to be considered additionally:

$$M_{\mathrm{spring}} = k_\beta \left(a_0 + a_1 \cos\Psi + b_1 \sin\Psi \right) \;, \qquad (17)$$

where k_β is the given spring stiffness. From (16), the coning and flapping angles are calculated. The resulting expression is omitted here for readability.

Complete BEM-Model Algorithm. Throughout above explanations, the induced velocity v_i was treated as a known quantity. However, when simulating the vehicle the induced velocity is unknown and needs to be calculated. This can be done by combining the results from momentum theory and blade-element theory: (5) and (13) can both be used to calculate the thrust of the propeller. Due to the relatively stiff blade, the flapping and coning angle are small (typically less than 1°) and can thus be neglected as a first approximation to calculate the induced velocity.

Due to the nature of the physical modeling process, the results from the momentum theory are only valid if the vehicle does not fly in its own downwash. This occurs when the vehicle descends with a certain speed, e.g. the propeller is in vortex-ring state [21] if and only if

$$0 < \frac{v_{\mathcal{P},z}}{v_i} < 2 \;. \qquad (18)$$

If the results from momentum-theory are not applicable, the induced velocity can not be calculated. In [21] a solution is presented which relies on an empirical fit to calculate the induced velocity in such flight conditions. The proposed fit consists of a quartic polynomial to approximate v_i as follows:

$$\tilde{v}_i = v_{h,i} \left(1 + 1.125(v_{\mathcal{P},z}/v_{h,i}) - 1.372(v_{\mathcal{P},z}/v_{h,i})^2 \right. \qquad (19)$$
$$\left. + 1.718(v_{\mathcal{P},z}/v_{h,i})^3 - 0.655(v_{\mathcal{P},z}/v_{h,i})^4 \right) \;,$$

where $v_{h,i}$ is the induced velocity if the vehicle would fly horizontally in the given flight state, i.e. set $v_{\mathcal{P},z} = 0$. To ensure a smooth transition back to the physical modeling, the final induced velocity in vortex ring state is given as $v_i = \max(\tilde{v}_i, v_{h,i})$.

The algorithm thus consists of the following steps:

1) Assume $a_0 = 0, a_1 = 0, b_1 = 0$.
2) Find v_i such that (5) and (13) are simultaneously satisfied, i.e. the thrust calculated by momentum theory and blade element theory are identical. If inside vortex-ring state, use the approximation presented above.
3) Calculate the coning and flapping angle a_0, a_1 and b_1 with the previously computed induced velocity.
4) Using the previously calculated induced velocity and blade flapping angles, (13) – (15) can be evaluated again.
5) The total force of the propeller and torque around the center of the propeller are given by:

$$\boldsymbol{f}_{\mathcal{P}} = \begin{bmatrix} -(H + \sin a_1 T) \\ \pm \sin b_1 T \\ -T \cos a_0 \end{bmatrix} \qquad \boldsymbol{\tau}_{\mathcal{P}} = \begin{bmatrix} \pm k_\beta b_1 \\ k_\beta a_1 \\ \mp Q \end{bmatrix} \;,$$

where the upper sign of $\pm$ and $\mp$ corresponds to propeller rotating clockwise and the lower sign needs to be used for a counter-clockwise spinning propeller.

E. Learned Residual Dynamics

Both rotor models presented in Sections III-C and III-D do not account for aerodynamic forces and torques caused by the quadrotor body or interaction effects between the propellers. In this work, these residual dynamics are approximated by a deep neural network. Modeling such effects accurately requires to implicitly estimate the airflow around the vehicle. Considering the airflow as hidden state of the system, it can be estimated by measuring a history of observable state variables. In this work, the angular and linear velocities, as well as motor speeds are used as input features for the neural network. A history length of $h = 20$, with temporary equally-spaced samples with a $\delta t = 2.5\,\mathrm{ms}$ is used, effectively giving information of the platform evolution over the past $50\,\mathrm{ms}$.

TABLE I: Comparison of different network architectures with respect to RMSE of force and torque prediction on a held-out test set.

Architecture	Force RMSE [N]	Torque RMSE [Nm]	# Param
TCN small	0.365	6.525 $\times 10^{-3}$	12k
TCN medium	0.352	5.274 $\times 10^{-3}$	25k
TCN large	0.355	4.674 $\times 10^{-3}$	72k
MLP	0.356	5.172 $\times 10^{-3}$	30k

The network architecture is empirically validated by minimizing the prediction error on an unseen test set. The candidate architectures consist of temporal-convolutional (TCN) encoders [41] and fully-connected (MLP) encoders, which both are combined with two fully-connected heads, one for the residual force prediction and one for the residual torque prediction. Each architecture uses leaky-ReLU activations and a linear output layer. Training is performed in a supervised fashion using the Adam optimizer by minimizing the RMSE loss on forces and torques between predictions and labels. Table I shows the main results of these ablation experiments. Due to its favourable performance versus inference time trade off, the medium-sized temporal-convolutional encoder (TCN-medium) was selected for all subsequent experiments.

IV. Experimental Setup

A. Data Collection

To train the model and to verify its accuracy, real world measurement data is needed. It is recorded in a flying arena equipped with a motion tracking system with a usable volume of $25\,\text{m} \times 25\,\text{m} \times 8\,\text{m}$. The Vicon[1] motion tracking system allows to record accurate position and attitude measurements at $400\,\text{Hz}$. Additionally, onboard IMU measurements and motor speeds are recorded at $1\,\text{kHz}$ by the low-level flight controller. This onboard data and the pose measurements need to be synchronized and fused in post processing. For this purpose interpolating cubic splines are fitted to the datapoints, which allows fusing the asynchronous measurements from both data sources, and recovers the full dynamic state. Furthermore, to estimate the unobserved linear velocity and angular acceleration, differentiation of the fitted splines provides less noisy estimates than direct differentiation of the discrete, noisy measurements. For the means of time synchronization, offset and clock skew are estimated through the correlation quality of the axis-wise angular rate measurement from the IMU with the spline. Gyroscope measurements are used because they provide better noise characteristics than the accelerometer data. The clock skew was typically observed to be $2.4\,\%$. The motor data is smoothed with a finite-impulse-response fourth-order Butterworth low-pass filter with a cutoff frequency corresponding to the time-constant of the motors, identified from the step response of the motors. This ensures that noise is suppressed without attenuating high-frequency motor signals more than $3\,\text{dB}$.

The resulting dataset contains 1.8 million data points recorded from 96 flights covering 1h:15min of flight time. The dataset is split into 70% training, 20% validation, and 10% test set. Each subsets contains trajectories that cover the full range of speeds and accelerations observed in the full data set.

B. Quadrotor Platform

The real-world flights are performed with a custom-made quadrotor platform. It features an Armattan Chameleon $6\,\text{inch}$ main frame, equipped with Hobbywing XRotor 2306 motors and $5\,\text{inch}$, three-bladed propellers. The platform has a total weight of $752\,\text{g}$ and can produce a maximum static thrust of approximately $33\,\text{N}$, which results in a static thrust-to-weight ratio of 4.5. The weight and power of this platform is comparable to the ones used by professional pilots in drone racing competitions. The platform's main computational unit is an NVIDIA Jetson TX2 accompanied by a ConnectTech Quasar carrier board. In all real world flights, control commands in the form of collective thrust and bodyrates are computed on a laptop computer and sent via a Laird module to the Jetson TX2. The Jetson then forwards these commands to a commercial flight controller running BetaFlight[2], which produces single-rotor commands that are fed to a 4-in-1 electronic speed controller.

C. Control System

Control commands are produced by a control pipeline consisting of two levels: (i) a high-level non-linear quadratic MPC controller generating bodyrate and collective thrust commands at $100\,\text{Hz}$, and (ii) a low-level Betaflight controller tracking the desired bodyrate setpoint at $1\,\text{kHz}$. To ensure repeatability, *BetaFlight* features targeted at human piloted drones (feed-forward terms) are disabled, and the controller is reduced to a PID with equal parameters for simulation and real-world flight. BetaFlight is run as a software module within the simulation, resembling the real control system. Both controllers are kept equal in the simulation with respect to the real-world experiments, to guarantee equal performance in both scenarios.

D. Simulator Extension

To compare different models, their resulting simulation accuracy is evaluated with respect to the real-world trajectory. For this purpose, the closed-loop system is simulated forward in time. While the rigid-body dynamics are given by (1) and the aerodynamic force and torque are provided by the model in question, there are two further components needed for an accurate simulation: integration and motor dynamics.

Integration. The integration is performed by a symplectic Euler scheme with a timestep of $1\,\text{ms}$ using the rigid body dynamics (1), the modeled linear and angular accelerations of the tested model, and the motor dynamics explained in the following section. The advantage of the symplectic Euler scheme is its energy conservation property, which is invalidated with other integration schemes, such as the standard Euler methods or the Runge-Kutta family of integrators.

Motor Dynamics. Since the aerodynamic model is based on the angular speed of the propeller, the motors are modeled as a first-order system according to

$$\frac{\delta}{\delta t}\Omega = \frac{1}{\tau_\Omega}(\Omega_{cmd} - \Omega) \tag{20}$$

where Ω_{cmd} is the commanded propeller speed and τ_Ω is the motor time constant. For the quadrotor platform used throughout the experiments, the time constant was identified to be $\tau_\Omega = 33\,\text{ms}$.

V. EXPERIMENTS AND RESULTS

The evaluation procedure is designed to address the following questions: (i) When does the classical approach to quadrotor modeling based on quadratic thrust and torque curves start to break down? (ii) How do the forces and torques predicted by a model based on BEM compare with respect to a simple quadratic model? (iii) What is the contribution of a learned residual dynamics component? The reader is encouraged to watch the attached video to understand the highly dynamic nature of the experiments.

A. Experimental Setup

Throughout the experiments multiple models are fitted using the training and validation data, and evaluated using the test data. The predictive performance of these models is compared in two different settings, covered in the subsequent sections: in Section V-B, the RMSE of predicted forces and torques is compared on unseen flight data; in Section V-C, different dynamics models are evaluated in conjunction with a known controller to determine the mismatch between simulation and the real world in a closed-loop scenario. Both types of comparisons are performed on unseen test trajectories that cover the entire performance envelope of the platform. Each trajectory has been performed on the real platform in an instrumented tracking volume as explained in Section IV.

B. Comparison of Predictive Performance

In a first set of experiments, the presented models are compared in terms of predicted forces and torques on unseen trajectories. These trajectories cover the entire performance envelope of the quadrotor, ranging from slow near-hover trajectories with speeds below $5\,\text{m}\,\text{s}^{-1}$ to aggressive trajectories at the limit of the platform's capabilities, exceeding speeds of $18\,\text{m}\,\text{s}^{-1}$ and accelerations up to $46.8\,\text{m}\,\text{s}^{-2}$. The models compared in this experiment consist of the quadratic model (*Fit*), the BEM model (*BEM*) and a naive model predicting all zeros (*None*). Each of these models is augmented with a learned residual correction using a neural network, marked with *+NN*. While the *None* model represents a naive baseline to better understand the magnitude of prediction errors, *None+NN* illustrates the performance of a purely learned model. Finally, the approach presented in [13] is compared, denoted as *PolyFit* as it uses automatically selected polynomial basis functions to fit the model.

To evaluate generalization performance, each approach is trained on two datasets: the entire training set as explained

TABLE II: Comparison of model performance in terms of RMSE on an unseen test set. Approaches marked with an asterisk are trained on a reduced training set to compare generalization performance.

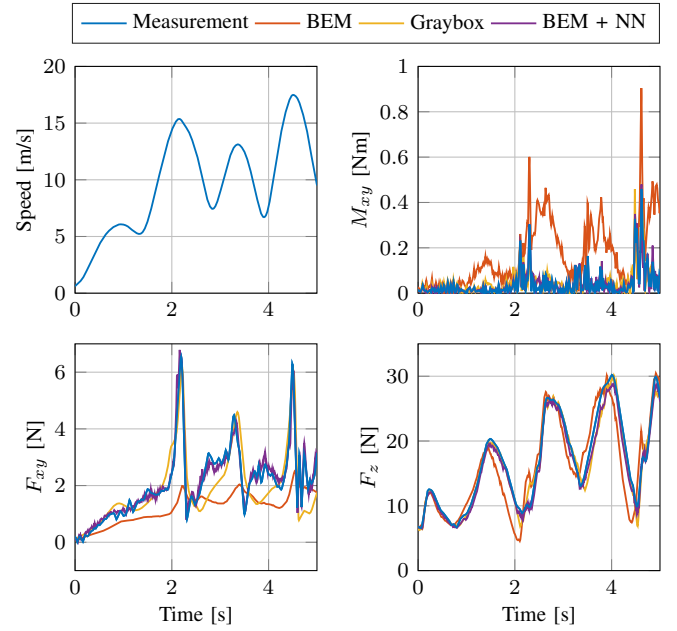| Model	F_{xy} [N]	F_z [N]	M_{xy} [Nm]	M_z [Nm]	F [N]	M [Nm]
None	1.549	13.618	0.036	0.006	7.964	0.029
Fit	1.536	1.381	0.104	0.033	1.486	0.087
BEM	0.803	1.265	0.090	0.017	0.982	0.074
PolyFit [13]	0.453	0.832	0.027	0.008	0.606	0.022
None+NN	0.236	0.681	0.017	**0.002**	0.438	0.014
Fit+NN	0.232	0.722	0.017	0.004	0.458	0.014
BEM+NN (ours)	**0.204**	**0.504**	**0.014**	0.004	**0.335**	**0.012**
PolyFit* [13]	1.450	6.637	2.815	0.164	4.011	2.301
None+NN*	0.470	1.959	**0.007**	**0.002**	1.194	**0.006**
Fit+NN*	0.501	1.225	0.024	0.013	0.817	0.021
BEM+NN* (ours)	**0.344**	**0.816**	0.025	0.008	**0.549**	0.021

Fig. 7: The plot illustrates the results presented in Table II. The plots show a highly aggressive maneuver (from the test dataset) where only models with neural net augmentation predict the forces and torques well.

in Section IV and a reduced dataset that only covers linear speeds up to $5\,\text{m}\,\text{s}^{-1}$. This reduced dataset is used to identify new parameters for each approach, which are marked with an asterisk.

Table II summarizes the results of this experiment, while Figure 7 illustrates performance on a highly aggressive maneuver. The proposed hybrid model based on BEM and learned residual dynamics consistently outperforms all other models on the predicted forces. Note that the trajectories performed in this work are designed to minimize yaw rate, and as a result only cover extremely small yaw torques M_z (the largest yaw torque in the dataset is $0.072\,\text{N}\,\text{m}$), as discussed in Section VI. It is evident that thrust and torque along the body $z_\mathcal{B}$-axis are more challenging to predict accurately. The reason for this is two-fold: First, all linear acceleration actuation lies in the $z_\mathcal{B}$ direction, contributing actuation noise predominantly along this axis. Second, all trajectories are designed to minimize yaw torque, since this is the least actuated torque direction, and

TABLE III: Comparison of closed-loop simulation performance on an unseen test set of different trajectories. Results show the positional RMSE between trajectories flown in simulation with different dynamics models and the same set of trajectories flown on the real platform. Models marked with an asterisk (*) were trained only on slow data up to $5\,\mathrm{m\,s^{-1}}$.

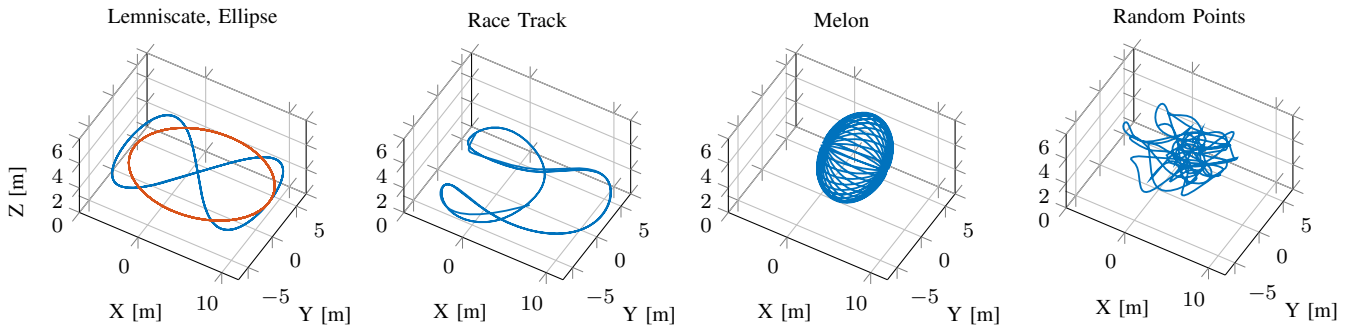|	v_{mean} [m/s]	v_{max} [m/s]	Fit	BEM	PolyFit [13]	None+NN	Fit+NN	BEM+NN (ours)	PolyFit*	None+NN*	Fit+NN*	BEM+NN* (ours)	RotorS [10]
Lemniscate	1.67	3.51	0.061	0.059	**0.043**	0.046	0.049	0.059	**0.046**	0.049	0.048	0.053	0.114
Random Points	2.38	8.25	0.167	0.138	0.130	crash	**0.126**	0.141	crash	0.130	**0.123**	0.134	0.239
Lemniscate	3.21	7.04	0.183	0.146	**0.102**	0.103	0.112	0.109	crash	**0.100**	0.112	0.112	0.148
Melon	3.57	7.63	0.229	0.163	**0.117**	0.126	0.126	0.133	crash	**0.127**	0.134	0.137	0.192
Slanted Circle	6.92	10.75	0.381	0.232	**0.166**	0.172	0.168	0.167	crash	0.210	0.193	**0.185**	0.257
Linear Oscillation	7.25	16.95	0.506	0.438	0.172	0.270	0.234	**0.171**	crash	0.258	0.263	**0.216**	0.562
Race Track	7.64	13.14	0.414	0.286	0.233	0.283	0.223	**0.214**	crash	0.257	0.262	**0.240**	0.424
Melon	7.74	13.55	0.431	0.221	0.239	0.179	0.164	**0.155**	crash	0.263	0.207	**0.185**	0.398
Slanted Circle	8.57	13.32	0.531	0.217	0.255	0.206	0.197	**0.192**	crash	0.340	0.268	**0.180**	0.397
Race Track	9.94	17.81	0.617	0.408	0.820	0.447	0.320	**0.301**	crash	0.446	0.420	**0.370**	0.708
Lemniscate	12.01	19.83	0.762	0.549	0.316	0.782	0.352	**0.286**	crash	0.469	0.423	**0.371**	1.158
Ellipse	15.02	19.20	0.855	0.369	crash	0.347	0.402	**0.285**	crash	0.605	0.481	**0.290**	0.653

Fig. 8: Trajectories used for testing. Each trajectory is flown multiple times with varying speeds.

significantly limits the acceleration envelope, and therefore the attainable agility. This explains the good performance of even very naive baselines with respect to this metric.

When comparing on the reduced dataset (Table II, bottom), the performance of the proposed approach gracefully degrades, still outperforming the baselines in terms of predicted forces. Note that the *PolyFit* baseline completely breaks down in this setting, indicating poor generalization to unseen data. The purely learning-based baseline outperforms all other approaches in the predicted torques, but also fails to generalize the force predictions to the new data.

C. Closed-Loop Comparison

To demonstrate the benefits of an accurate force and torque model, a second set of experiments presents a comparison of closed-loop simulation performance. Using the simulation setup explained in Section IV-D, a set of unseen trajectories (Figure 8) is flown in simulation and the resulting flight path is compared with the data obtained from executing the same set of trajectories on the real platform. As for the previous set of experiments, also this comparison is performed for models identified on the full training set, as well as a reduced training set to compare generalization performance. Additionally to the baselines already used in the previous experiments, this experiment also compares against RotorS [10]. To do this, the standard model in RotorS is updated with the parameters identified from the real platform (i.e. mass, inertia, dimensions, lift and drag coefficients).

Table III illustrates the results of the closed-loop experiment. For each trajectory, the accumulated positional error between the simulated flight and the data observed in the real world is reported. As can be seen, all models achieve similar performance for trajectories close to hover. The *Fit* model exhibits increasing bias with higher speeds, with the error exceeding the worst performance of the proposed approach already at average speeds below $7\,\mathrm{m\,s^{-1}}$. In contrast, the *BEM* model is able to maintain competitive performance up to the fastest trajectories. At low speeds, the *PolyFit* baseline performs very well, but exhibits increasing bias for higher speeds, even resulting in a crash on the fastest trajectory. The RotorS baseline performs inferior on all trajectories, achieving results comparable to the *Fit* baseline. The proposed approach combining BEM with a learned residual term (*BEM+NN*) achieves competitive performance on the slow trajectories and outperforms all baselines on the faster trajectories. Compared to *Fit+NN*, *BEM+NN* achieves consistently better performance for fast maneuvers.

When trained on the reduced dataset, all models show decreased performance. However, while approaches such as *PolyFit* completely break down, the proposed approach experiences only a minor performance reduction around 20%, outperforming the baselines on all faster trajectories. This result highlights the ability of the proposed approach to generalize beyond the data it was trained on (Figure 9).
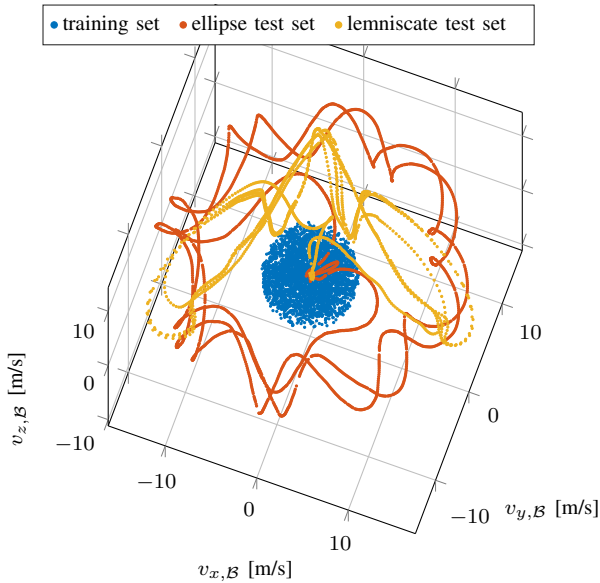
Fig. 9: Visualization of the reduced training set, the ellipse test set and the fastest lemniscate test set in the body-frame velocity space. Although the test set mostly covers regions of the state space that are not part of the training set, the trained BEM+NN⋆ model still provides good accuracy in simulating the trajectory. This demonstrates its remarkable generalization capability.

VI. Discussion

The results obtained in this work show that the proposed hybrid dynamics model, combining first-principles based on blade-element-momentum theory with a learning-based residual term, outperforms state-of-the-art modeling for quadrotors with a 50% decreased aerodynamic force and torque prediction error. Furthermore, evaluation in controlled experiments on a large real-world dataset shows that such a complementary modeling approach outperforms each of its compositional submodules.

In fact, not only does the performance of the proposed hybrid model structure improve with a more capable rotor model, but the learned residual dynamics also increase in accuracy if a broader envelope of effects can already be captured using first principles. Specifically, the learned residual prediction achieves up to 30% better performance when combined with a rotor model based on blade-element-momentum theory.

While the proposed approach significantly improves upon state-of-the-art in quadrotor modeling for highly aggressive maneuvers by up to 60%, its advantages for slow speed trajectories below $5\,\mathrm{m\,s^{-1}}$ are limited. Our experiments indicate that for such slow trajectories, a traditional parametric approach such as [13] achieves very strong performance at a lower computational cost. While the quadratic and polynomial fits can be evaluated in a mere $1\,\mathrm{\mu s}$, even on a micro processor, the BEM model requires in the order of $100\,\mathrm{\mu s}$ on a modern Intel-architecture CPU, and a forward pass of the network averages also at around $100\,\mathrm{\mu s}$ on a modern NVidia GPU. The reason for the dominant runtime of the BEM model is the necessary implicit solution for the induced velocity equation. Even though our approach is not optimized for runtime, simulations can be run at an arbitrary timescale,

where most applications gladly trade-off real-time evaluation for improved accuracy.

The results of the closed-loop simulation using the proposed model could be further improved by refining the following aspects of the control pipeline: (i) The experimental platform currently relies on the BetaFlight inner-loop low-level controller, which is optimized for human pilots. However, as such, it only takes a throttle command and a body rate command as inputs, and relies on the inner-loop to track the rate command. Furthermore, it performs filtering and interpolation of the control signals to ensure a consistent flight feeling for human pilots, which introduces undesirable control-loop shaping. An MPC outer-loop controller directly outputting single-rotor motor speeds that are tracked by an inner loop motor speed controller would improve the accuracy of our approach further as this would minimize the differences between the simulation and the actual experiments. (ii) Modeling the latency from the motion capture pose filtering, the data transmission to the drone, and the communication to the flight controller in simulation would also reduce the error as it improves the realism of the simulator. The authors expect the results in Table III to be even more favorable for their approach in such an ideal setting. The accuracy of the force and torque predictions shown in Table II would also benefit from a custom low-level controller providing more precise and less noisy motor-speed information.

Compared to a purely learning-based approach such as *None+NN*, the proposed approach performs 25% better for all non-trivial trajectories with average speeds above $4\,\mathrm{m\,s^{-1}}$, and extrapolates well to unseen flight data, as opposed to e.g. the *PolyFit* baseline. The authors expect the performance of learning-only approaches to improve with more data. However, in a real world setting, where high-quality data is sparse, pure learning-based approaches fall short of traditional methods. Additionally, accurate aerodynamic force and torque prediction does not necessarily translate to good closed-loop performance, as demonstrated in our evaluation. Moreover, this study observed cases where a purely-learned residual component introduced a feedback loop on the predicted torques that led to a crash. In such cases, support through first-principles is vital for accurate and robust modeling.

VII. Conclusion

This work proposes a novel method to model quadrotors by combining modeling based on first principles with a learning-based residual term represented by a neural network. The proposed method is able to accurately model quadrotors even throughout aggressive trajectories pushing the platform to its limits. This hybrid model outperforms its compositional modules with up to 50% error reduction, including baseline methods that utilize only first-principles modeling, as well as purely learning-based methods. The method shows strong generalization beyond the training set used to identify the model and predicts accurate forces and torques where other methods break down. Controlled experiments indicate that the fusion of

learned dynamics with first-principles is a powerful combination, where the learned dynamics-residual benefits from high-fidelity models, such as the BEM. Applied to simulations, our approach enables unprecedented accuracy, reducing positional RMSE from $\sim 0.8\,\mathrm{m}$ for state-of-the-art approaches, down to below $0.3\,\mathrm{m}$. This could tremendously speed up development and testing of advanced control and navigation strategies for quadrotors, without the need of the tedious and crash-prone trial-and-error strategy on real systems.

REFERENCES

[1] Gilhyun Ryou, Ezra Tal, and Sertac Karaman. Multi-fidelity black-box optimization for time-optimal quadrotor maneuvers. In *Robotics: Science and Systems (RSS)*, 2020.

[2] 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*, 2(2):404–411, 2017. doi: 10.1109/LRA.2016.2633290.

[3] Elia Kaufmann, Antonio Loquercio, René Ranftl, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Deep drone acrobatics. *RSS: Robotics, Science, and Systems*, 2020.

[4] Philipp Foehn, Angel Romero, and Davide Scaramuzza. Time-optimal planning for quadrotor waypoint flight. *Science Robotics*, 2021.

[5] Hyungpil Moon, Jose Martinez-Carranza, Titus Cieslewski, Matthias Faessler, Davide Falanga, Alessandro Simovic, Davide Scaramuzza, Shuo Li, Michael Ozo, Christophe De Wagter, et al. Challenges and implemented technologies used in autonomous drone racing. *Intelligent Service Robotics*, 12(2):137–148, 2019.

[6] R. Madaan, N. Gyde, S. Vemprala, M. Brown, K. Nagami, T. Taubner, E. Cristofalo, D. Scaramuzza, M. Schwager, and A. Kapoor. Airsim drone racing lab. In *PLMR Post Proceedings of the NeurIPS 2019 Competition Track*, 2020.

[7] Philipp Foehn, Dario Brescianini, Elia Kaufmann, Titus Cieslewski, Mathias Gehrig, Manasi Muglikar, and Davide Scaramuzza. Alphapilot: Autonomous drone racing. *RSS: Robotics, Science, and Systems*, 2020.

[8] Ali Punjani and Pieter Abbeel. Deep learning helicopter dynamics models. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3223–3230. IEEE, 2015.

[9] Somil Bansal, Anayo K Akametalu, Frank J Jiang, Forrest Laine, and Claire J Tomlin. Learning quadrotor dynamics using neural network for flight control. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 4653–4660. IEEE, 2016.

[10] Fadri Furrer, Michael Burri, Markus Achtelik, and Roland Siegwart. Rotors—a modular gazebo mav simulator framework. In *Robot Operating System (ROS)*, pages 595–625. Springer, 2016.

[11] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and service robotics*, pages 621–635. Springer, 2018.

[12] Matthias Faessler, Antonio Franchi, and Davide Scaramuzza. Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories. *IEEE Robot. Autom. Lett.*, 2017.

[13] Sihao Sun, Coen C de Visser, and Qiping Chu. Quadrotor gray-box model identification from high-speed flight data. *Journal of Aircraft*, 56(2):645–661, 2019.

[14] Patricia Ventura Diaz and Steven Yoon. High-fidelity computational aerodynamics of multi-rotor unmanned aerial vehicles. In *2018 AIAA Aerospace Sciences Meeting*, page 1266, 2018.

[15] Nima Mohajerin, Melissa Mozifian, and Steven Waslander. Deep learning a quadrotor dynamic model for multi-step prediction. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2454–2459. IEEE, 2018.

[16] Gavin D Portwood, Peetak P Mitra, Mateus Dias Ribeiro, Tan Minh Nguyen, Balasubramanya T Nadiga, Juan A Saenz, Michael Chertkov, Animesh Garg, Anima Anandkumar, and Andreas Dengel. Turbulence forecasting via neural ode. *2nd Workshop on Machine Learning and the Physical Sciences (NeurIPS 2019)*, 2019.

[17] Radek Grzeszczuk, Demetri Terzopoulos, and Geoffrey Hinton. Neuroanimator: Fast neural network emulation and control of physics-based models. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 9–20, 1998.

[18] Robert Mahony, Vijay Kumar, and Peter Corke. Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor. *IEEE Robotics and Automation magazine*, 19(3):20–32, 2012.

[19] Yunlong Song, Selim Naji, Elia Kaufmann, Antonio Loquercio, and Davide Scaramuzza. Flightmare: A flexible quadrotor simulator. 2020.

[20] Johannes Meyer, Alexander Sendobry, Stefan Kohlbrecher, Uwe Klingauf, and Oskar Von Stryk. Comprehensive simulation of quadrotor uavs using ros and gazebo. In *International conference on simulation, modeling, and programming for autonomous robots*, pages 400–411. Springer, 2012.

[21] Gabriel Hoffmann, Haomiao Huang, Steven Waslander, and Claire Tomlin. Quadrotor helicopter flight dynamics and control: Theory and experiment. In *AIAA guidance, navigation and control conference and exhibit*, page 6461, 2007.

[22] Haomiao Huang, Gabriel M Hoffmann, Steven L Waslander, and Claire J Tomlin. Aerodynamics and control of autonomous quadrotor helicopters in aggressive maneuvering. In *2009 IEEE international conference on robotics and automation*, pages 3277–3282. IEEE, 2009.

[23] Gabriel M Hoffmann, Haomiao Huang, Steven L Waslander, and Claire J Tomlin. Precision flight control for a multi-vehicle quadrotor helicopter testbed. *Control engineering practice*, 19(9):1023–1036, 2011.

[24] Raymond W Prouty. *Helicopter performance, stability, and control.* 1995.

[25] Matko Orsag and Stjepan Bogdan. Influence of forward and descent flight on quadrotor dynamics. *Recent Advances in Aircraft Technology*, pages 141–156, 2012.

[26] Pierre-Jean Bristeau, Philippe Martin, Erwan Salaün, and Nicolas Petit. The role of propeller aerodynamics in the model of a quadrotor uav. In *2009 European control conference (ECC)*, pages 683–688. IEEE, 2009.

[27] Derya Kaya and Ali T Kutay. Aerodynamic modeling and parameter estimation of a quadrotor helicopter. In *AIAA Atmospheric Flight Mechanics Conference*, page 2558, 2014.

[28] Yi-Rui Tang and Yangmin Li. Dynamic modeling for high-performance controller design of a uav quadrotor. In *2015 IEEE International Conference on Information and Automation*, pages 3112–3117. IEEE, 2015.

[29] Caitlin Powers, Daniel Mellinger, Aleksandr Kushleyev, Bruce Kothmann, and Vijay Kumar. Influence of aerodynamics and proximity effects in quadrotor flight. In *Experimental robotics*, pages 289–302. Springer, 2013.

[30] Waqas Khan and Meyer Nahon. Toward an accurate physics-based uav thruster model. *IEEE/ASME Transactions on Mechatronics*, 18(4):1269–1279, 2013.

[31] Rajan Gill and Raffaello D'Andrea. Propeller thrust and drag in forward flight. In *2017 IEEE Conference on Control Technology and Applications (CCTA)*, pages 73–79. IEEE, 2017.

[32] Rajan Gill and Raffaello D'Andrea. Computationally efficient force and moment models for propellers in uav forward flight applications. *Drones*, 3(4):77, 2019.

[33] Carl Russell, Jaewoo Jung, Gina Willink, and Brett Glasner. Wind tunnel and hover performance test results for multicopter uas vehicles. In *AHS 72nd annual forum*, pages 16–19, 2016.

[34] Fabrizio Schiano, Javier Alonso-Mora, Konrad Rudin, Paul Beardsley, Roland Y Siegwart, and Bruno Sicilianok. Towards estimation and correction of wind effects on a quadrotor uav. In *IMAV 2014: International Micro Air Vehicle Conference and Competition 2014*, pages 134–141, 2014.

[35] Engin Baris, Colin P Britcher, and George Altamirano. Wind tunnel testing of static and dynamic aerodynamic characteristics of a quadcopter. In *AIAA Aviation 2019 Forum*, page 2973, 2019.

[36] Guillem Torrente, Elia Kaufmann, Philipp Föhn, and Davide Scaramuzza. Data-driven MPC for quadrotors. *IEEE Robot. Autom. Lett.*, 2021.

[37] Jinglin Luo, Longfei Zhu, and Guirong Yan. Novel quadrotor forward-flight model based on wake interference. *Aiaa Journal*, 53(12):3522–3533, 2015.

[38] Nima Mohajerin and Steven Waslander. Multistep prediction of dynamic systems with recurrent neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2019.

[39] Guanya Shi, Xichen Shi, Michael O'Connell, Rose Yu, Kamyar Aziz-zadenesheli, Animashree Anandkumar, Yisong Yue, and Soon-Jo Chung. Neural lander: Stable drone landing control using learned dynamics. *2019 International Conference on Robotics and Automation (ICRA)*, May 2019. doi: 10.1109/icra.2019.8794351. URL http://dx.doi.org/10.1109/ICRA.2019.8794351.

[40] Guillaume Ducard and Minh-Duc Hua. Modeling of an unmanned hybrid aerial vehicle. In *2014 IEEE Conference on Control Applications (CCA)*, pages 1011–1016. IEEE, 2014.

[41] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.