This paper has been accepted for publication at the Transactions on Robotics (T-RO) 2025

HDVIO2.0: Wind and Disturbance Estimation with Hybrid Dynamics VIO

Giovanni Cioffi, Leonard Bauersfeld, Davide Scaramuzza

***Abstract*—Visual-inertial odometry (VIO) is widely used for state estimation in autonomous micro aerial vehicles using onboard sensors. Current methods improve VIO by incorporating a model of the translational vehicle dynamics, yet their performance degrades when faced with low-accuracy vehicle models or continuous external disturbances, like wind. Additionally, incorporating rotational dynamics in these models is computationally intractable when they are deployed in online applications, e.g., in a closed-loop control system. We present HDVIO2.0, which models full 6-DoF, translational and rotational, vehicle dynamics and tightly incorporates them into a VIO system with minimal impact on the runtime. HDVIO2.0 builds upon the previous work, HDVIO, and addresses these challenges through a hybrid dynamics model combining a point-mass vehicle model with a learning-based component, with access to control commands and IMU history, to capture complex aerodynamic effects. The key idea behind modeling the rotational dynamics is to represent them with continuous-time functions. HDVIO2.0 leverages the divergence between the actual motion and the predicted motion from the hybrid dynamics model to estimate external forces as well as the robot state. Our system surpasses the performance of state-of-the-art methods in experiments using public and new drone dynamics datasets, as well as real-world flights in winds up to 25 km/h. Unlike existing approaches, we also show that accurate vehicle dynamics predictions are achievable without precise knowledge of the vehicle state.**

***Index Terms*—Visual-Inertial SLAM, Learning Robot Dynamics, Aerial Systems: Perception and Autonomy.**

Supplementary Material

Video: https://youtu.be/wUaEp0YGpDM
Code:https://github.com/uzh-rpg/hdvio2.0

I. Introduction

VISUAL-INERTIAL odometry (VIO) is the standard method for state estimation in consumer and inspection drones. To enhance the performance of VIO systems, several recent approaches have proposed tightly integrating drone dynamics into the VIO pipeline [1, 2, 3, 4, 5]. Incorporating system dynamics into the VIO framework provides additional information, enabling the system to distinguish between motion resulting from actuation and motion caused by external perturbations. This integration improves pose estimation accuracy and allows for the estimation of external forces acting on the drone.

While effective in many scenarios, state-of-the-art methods face significant performance degradation in cases of large model mismatches (e.g., high speeds, systematic noise in actuation inputs) or persistent external disturbances like wind. These issues arise because existing methods rely on simplifying assumptions—such as neglecting aerodynamic drag and assuming zero-mean noise in system dynamics—that fail to hold under such conditions. Directly incorporating high-fidelity dynamics models [6, 7] into a VIO pipeline can be counterproductive because these models require the drone state as input (typically velocity and attitude). This situation can create a compounding effect, where errors in the VIO output propagate through the dynamics model, and, in turn, further impact the VIO.

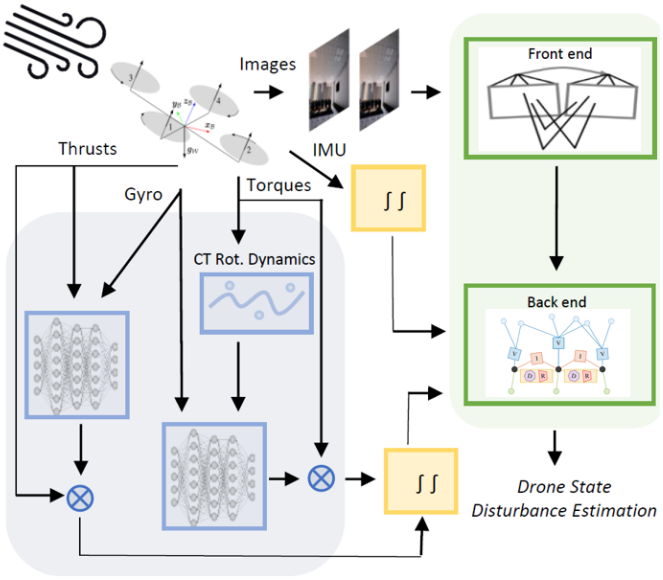

Fig. 1: HDVIO2.0 estimates the robot's state and external disturbances using visual, inertial, and dynamics measurements. Notably, it models the robot's dynamics by combining a simple physical model with learning-based components.

Overcoming these challenges is essential for deploying model-based VIO estimators in applications where aerodynamic effects play a significant role, such as fast flights [8], operations in windy conditions [9], or scenarios with modeling inaccuracies [10]. The state-of-the-art methods VIMO [1] and VID [2] incorporate the translational drone dynamics into an optimization-based VIO framework [11] through a residual term derived from the propagation of a point-mass dynamics model. The residual term is formulated based on the preintegration theory [12], which requires separating the measurements (namely, the control inputs) from the states. While this formulation is straightforward for translational dynamics, extending it to rotational dynamics is not trivial. The simplified dynamics model neglects aerodynamic effects, treating drag as part of the external force estimate. Addi-

The authors are with the Robotics and Perception Group, Department of Informatics, University of Zurich, Switzerland, https://rpg.ifi.uzh.ch.
This work was supported by the European Union's Horizon Europe Research and Innovation Programme under grant agreement No. 101120732 (AUTOASSESS) and the European Research Council (ERC) under grant agreement No. 864042 (AGILEFLIGHT).

tionally, potential systematic offsets in actuation inputs (e.g., miscalibrations such as incorrect rotor lift coefficients) are interpreted as accelerometer biases, introducing errors into the inertial residuals and reducing motion estimation accuracy.

Improving the drone dynamics model within the VIO estimator is key to addressing these limitations.

Contribution

We present HDVIO2.0, the first VIO pipeline incorporating the 6-DoF drone dynamics using a hybrid model that combines a physical model with a learning-based component, see Fig 1. Unlike prior methods [6, 7], our learned dynamics model predicts residual aerodynamic forces without requiring the drone state (i.e., attitude and velocity) as input. Instead, it uses a temporal convolutional network (TCN) [13] that takes control commands and gyroscope measurements as input. Our hybrid model is integrated into an optimization-based VIO framework [14, 15], leveraging preintegration theory [12] to efficiently compute and optimize the dynamics residuals alongside monocular camera and IMU residuals. HDVIO2.0 extends the previous work, HDVIO [16], by integrating the rotational dynamics in the drone model. We represent the drone's angular velocity as a continuous-time function using B-splines, which effectively capture the drone's dynamics [17]. The B-spline is optimized by minimizing the difference between its first derivative and the angular acceleration derived from the drone's rotational model, which combines torque commands and learned residual torques. Angular velocities sampled from the B-spline representation of the drone's rotational dynamics model are preintegrated into a residual term that can be directly included in the VIO. HDVIO2.0 employs two TCNs: one predicts residual thrust using thrust commands and gyroscope measurements (as in HDVIO), while the other predicts residual torque based on torque commands and gyroscope measurements—a novel contribution.

We evaluate HDVIO2.0 against the same VIO system without the proposed hybrid dynamics model, and the state-of-the-art systems, VIMO [1] and VID [2]. On public datasets, Blackbird [18] and VID [19], we show that HDVIO2.0 surpasses state-of-the-art methods. In wind field experiments, we show that HDVIO2.0 accurately predicts the wind force, again outperforming the baselines. On the NeuroBEM dataset [6], our learned dynamics model demonstrates competitive performance with existing aerodynamics models. Notably, it is the first data-driven dynamics model to predict forces without requiring the vehicle state, i.e., linear velocity and attitude, as input. Additionally, to the best of our knowledge, our learning-based model is the first data-driven dynamics model that trains without requiring ground-truth force measurements, relying solely on position, velocity, and orientation supervision signals. This eliminates the need for motion-capture systems, as pose estimates obtained from offline structure from motion-based systems [20, 17] are sufficiently accurate for training.

By providing accurate state and external force estimates, we believe that HDVIO2.0 advances the deployment of autonomous drones in safety-critical applications, such as disaster site surveying and air transport, which currently depend on human pilots.

II. Related Work

The related work on visual-inertial odometry (VIO) with external force estimation can be categorized into *loosely-coupled* and *tightly-coupled* methods. Loosely-coupled approaches [21, 22, 23, 24, 25, 26, 27, 28] estimate external forces independently from motion estimation, while tightly-coupled methods [1, 2, 4] jointly estimate both the robot's motion and external perturbations.

A. Loosely-Coupled Methods

Early loosely-coupled methods [21, 22, 23] rely on deterministic force and torque observers derived from the robot's dynamics model. These approaches assume access to accurate state estimates from separate estimators. Later, probabilistic methods [24, 25, 26, 27] improved accuracy by incorporating sensor noise. These methods utilize the Extended Kalman Filter (EKF) [25, 27, 29] or the Unscented Kalman Filter (UKF) [24, 26]. For example, the work in [3] uses a quadrotor model to enhance an EKF-based VIO estimator [30] for simultaneous system identification and state estimation. This work highlights that decoupling state estimation from dynamics measurements is optimal in the presence of high noise. In [31], a UKF estimates external disturbances like wind and human interactions using outputs from a neural network that processes airflow sensor data and motion capture measurements.

While effective under high signal-to-noise ratio conditions, loosely-coupled methods neglect the correlation between estimated variables and their noise characteristics, resulting in reduced performance when using noisy sensors.

B. Tightly-Coupled Methods

Tightly-coupled methods address this limitation by jointly estimating robot motion and external perturbations. VIMO [1] integrates robot dynamics into an optimization-based VIO framework [11]. It introduces a residual term representing translational motion constraints derived from robot dynamics, including external forces, using IMU preintegration theory [12]. This method preintegrates high-rate thrust inputs into residuals between consecutive camera frames. External forces are modeled as zero-mean Gaussian variables to account for their unknown dynamics. VIMO is the first work that enables the simultaneous estimation of external forces and robot states. Multiple extensions to VIMO exist. The work in [32] extends VIMO with a disturbance observer for constant force estimation. The disturbance observer allows the system to differentiate between the constant external force and the accelerometer bias. VID-Fusion [2] extends VIMO by modifying the external force model, where the mean of the Gaussian distribution is based on the average difference between accelerometer and thrust measurements within the preintegration window. This model of the external force allows VID-Fusion to estimate constant loads attached to the drone. We employ this external force model in HDVIO2.0. The VIMO framework is general and can be employed for any type of robot. In fact, extensions of VIMO for legged robots are proposed in [4, 5].

However, as discussed in Sec. I, VIMO and all its extensions struggle with continuous external forces or model mismatches.

C. Drone Dynamics Modeling

Accurate dynamics modeling is critical for HDVIO2.0. Prior methods assume access to the vehicle state, which is unsuitable for VIO pipelines as it introduces a compounding effect that propagates errors in the dynamics model to the VIO and vice versa. For completeness, a brief review of quadrotor modeling literature is presented. Basic models treat quadrotors as rigid bodies with linear mass and inertia dynamics, exerting force in the body-z direction while neglecting or simplifying (assuming it linear) aerodynamic drag [33, 34, 35, 36]. First principles can be used to refine these basic models, resulting in blade-element momentum (BEM) theory [6, 37, 38, 39]. Pure data-driven models [7] have gained traction due to the complexity of quadrotor aerodynamics and have shown superior performance compared to first-principles-based methods. The state-of-the-art model, NeuroBEM [6], combines a physical model with a learning-based component, outperforming previous methods. This hybrid modeling approach inspired our use of a learned component in HDVIO2.0 to enhance drone dynamics modeling.

HDVIO2.0 extends HDVIO by modeling the quadrotor's rotational dynamics using a continuous-time representation of angular velocities, implemented via B-splines for computational efficiency. Alternatively, Gaussian Processes (GPs) have been used as CT representations in prior work [40, 41, 42] to derive preintegration terms based on IMU measurements. In [40, 41], the 6-DoF sensor pose is modeled using six independent GPs. These GPs are optimized to fit IMU measurements, and preintegration terms for orientation, velocity, and position are obtained either by (i) sampling angular velocities and linear accelerations followed by numerical integration [40], or (ii) applying linear integration operators directly to the GPs [41]. The method proposed in [41] is applied in an event-based odometry system in [42]. While GPs provide accurate results, their high computational cost makes them impractical for real-time use in HDVIO2.0.

III. Methodology

This section outlines our visual-inertial-hybrid drone dynamics odometry algorithm. We begin by defining the notation used throughout the paper and describing the drone dynamics. Next, we formulate the estimation problem. Following, we provide a concise derivation of the dynamics residual term, which is based on the preintegration theory [12]. Finally, we introduce our learning-based module of drone dynamics.

A. Notation

In this paper, scalars are represented using non-bold notation $[s, S]$, vectors are denoted in lowercase bold $\boldsymbol{v}$, and matrices are expressed in uppercase bold $\boldsymbol{M}$. World $\mathcal{W}$, Body $\mathcal{B}$, IMU $\mathcal{I}$, and camera $\mathcal{C}$ frames are defined with an orthonormal basis, such as $\{\boldsymbol{x}^{\mathcal{W}}, \boldsymbol{y}^{\mathcal{W}}, \boldsymbol{z}^{\mathcal{W}}\}$. The $\mathcal{B}$ frame is positioned at the quadrotor's center of mass, and for simplicity, the IMU frame $\mathcal{I}$ is assumed to coincide with $\mathcal{B}$. The notation $(\cdot)^{\mathcal{W}}$ is used to indicate quantities expressed in the world frame, and similar notation is applied to other reference frames. At time t_k, the position, orientation, and linear velocity of $\mathcal{B}$ relative to $\mathcal{W}$ are denoted as $\boldsymbol{p}^{\mathcal{W}}_{\mathcal{B}_k} \in \mathbb{R}^3$, $\boldsymbol{R}^{\mathcal{W}}_{\mathcal{B}_k} \in \mathbb{R}^{3\times3}$ (a member of the rotation group $SO(3)$), and $\boldsymbol{v}^{\mathcal{W}}_{\mathcal{B}_k} \in \mathbb{R}^3$, respectively. The unit quaternion representation of $\boldsymbol{R}^{\mathcal{W}}_{\mathcal{B}_k}$ is given as $\boldsymbol{q}^{\mathcal{W}}_{\mathcal{B}_k}$. The cross product of two vectors is denoted by $\times$. The quaternion product is denoted by $\otimes$. The gravity vector in the world frame is denoted by $\boldsymbol{g}^{\mathcal{W}}$. The accelerometer model is: $\hat{\boldsymbol{a}}^{\mathcal{B}_k} = \boldsymbol{a}^{\mathcal{B}_k} + \boldsymbol{b}_{a_k} + \boldsymbol{n}_a$, where the noise is modeled as additive Gaussian noise $\boldsymbol{n}_a \sim \mathcal{N}(0, \boldsymbol{\sigma}_a^2)$ and the bias as a random walk $\dot{\boldsymbol{b}}_{a_k} = \boldsymbol{n}_{b_a}$, with $\boldsymbol{n}_{b_a} \sim \mathcal{N}(0, \boldsymbol{\sigma}_{b_a}^2)$. The gyroscope model is: $\hat{\boldsymbol{\omega}}_g^{\mathcal{B}_k} = \boldsymbol{\omega}_g^{\mathcal{B}_k} + \boldsymbol{b}_{\omega_{gk}} + \boldsymbol{n}_{\omega_g}$, where the noise is modeled as additive Gaussian noise $\boldsymbol{n}_{\omega_g} \sim \mathcal{N}(0, \boldsymbol{\sigma}_{\omega_g}^2)$ and the bias as a random walk $\dot{\boldsymbol{b}}_{\omega_{gk}} = \boldsymbol{n}_{b_{\omega_g}}$, with $\boldsymbol{n}_{b_{\omega_g}} \sim \mathcal{N}(0, \boldsymbol{\sigma}_{b_{\omega_g}}^2)$. We indicated noisy measurements using the symbol $\hat{\cdot}$.

B. Quadrotor Dynamics

The quadrotor is modeled as a 6 degree-of-freedom rigid body with mass m and a diagonal moment of inertia matrix $\boldsymbol{J} = diag(J_x, J_y, J_z)$. The dynamics governing the position, velocity, and orientation of the quadrotor platform are described by the following equations:

$$
\begin{aligned}
\dot{\boldsymbol{p}}^{\mathcal{W}}_{\mathcal{B}_k} &= \boldsymbol{v}^{\mathcal{W}}_{\mathcal{B}_k} \\
\dot{\boldsymbol{v}}^{\mathcal{W}}_{\mathcal{B}_k} &= \boldsymbol{R}^{\mathcal{W}}_{\mathcal{B}_k}(\boldsymbol{f}^{\mathcal{B}}_{t_k} + \boldsymbol{f}^{\mathcal{B}}_{res_k} + \boldsymbol{f}^{\mathcal{B}}_{e_k}) + \boldsymbol{g}^{\mathcal{W}} \\
\dot{\boldsymbol{q}}^{\mathcal{W}}_{\mathcal{B}_k} &= \frac{1}{2}\boldsymbol{q}^{\mathcal{W}}_{\mathcal{B}_k} \otimes [0, \boldsymbol{\omega}^{\mathcal{B}_k}_k]^\top \\
\dot{\boldsymbol{\omega}}^{\mathcal{B}}_k &= \boldsymbol{J}^{-1}(\boldsymbol{\tau}^{\mathcal{B}}_k + \boldsymbol{\tau}^{\mathcal{B}}_{res_k} - \boldsymbol{\omega}^{\mathcal{B}}_k \times \boldsymbol{J}\boldsymbol{\omega}^{\mathcal{B}}_k),
\end{aligned} \tag{1}
$$

where $\boldsymbol{f}^{\mathcal{B}}_{t_k} = [0, 0, T_k]^\top$ represents the mass-normalized collective thrust, $\boldsymbol{f}^{\mathcal{B}}_{e_k}$ denotes the external force acting on the quadrotor, $\omega^{\mathcal{B}}_k$ is the quadrotor's body rate, and $\boldsymbol{\tau}^{\mathcal{B}}_k$ is the torque produced by the propellers. We define the mass-normalized collective thrust measurement model as: $\hat{\boldsymbol{f}}^{\mathcal{B}}_{t_k} = \boldsymbol{f}^{\mathcal{B}}_{t_k} + \boldsymbol{n}_{f_t}$, where $\boldsymbol{n}_{f_t} \sim \mathcal{N}(0, \boldsymbol{\sigma}_{f_t}^2)$ is a zero-mean gaussian noise to account for uncertainty in the force direction. For conciseness, we will refer to the mass-normalized collective thrust as simply thrust hereafter. Similarly, the torque measurement model is: $\hat{\boldsymbol{\tau}}^{\mathcal{B}}_k = \boldsymbol{\tau}^{\mathcal{B}}_k + \boldsymbol{n}_\tau$, where $\boldsymbol{n}_\tau \sim \mathcal{N}(0, \boldsymbol{\sigma}_\tau^2)$ is a zero-mean gaussian noise. The body rate measurement model is: $\hat{\boldsymbol{\omega}}^{\mathcal{B}}_k = \boldsymbol{\omega}^{\mathcal{B}}_k + \boldsymbol{n}_\omega$, where $\hat{\boldsymbol{\omega}}^{\mathcal{B}}_k$ is sampled from the B-spline, see Sec. III-B1, and $\boldsymbol{n}_\omega$ is a noise value that accounts for uncertainty in the B-spline fitting process. To account for aerodynamic effects and unknown systematic noise in the inputs, residual terms $\boldsymbol{f}^{\mathcal{B}}_{res_k}$ and $\boldsymbol{\tau}^{\mathcal{B}}_{res_k}$ are introduced. The external force $\boldsymbol{f}^{\mathcal{B}}_{e_k}$ is modeled as a random variable following a Gaussian distribution obtained by computing the difference between acceleration and thrust measurements as proposed in [2]. Modeling the external force in this manner enables the estimator to differentiate between the slowly varying accelerometer bias and external forces, which may arise from incidental disturbances or constant loads, such as an external mass attached to the drone.

The dynamics motion constraints, c.f. Section III-D, are derived using the preintegration theory [12]. The preintegration

theory allows for efficient integration of these constraints in the VIO optimization-based backend. To apply the preintegration theory, a measurement model of the form $\boldsymbol{y} = f(\boldsymbol{x})$ is required in which the optimization variables $\boldsymbol{x}$ can be separated from the measurements $\boldsymbol{y}$, and $f(\cdot)$ describes the relationship between $\boldsymbol{x}$ and $\boldsymbol{y}$. The quadrotor rotational dynamics (last two equations of Eq. 1) cannot be formulated in a measurement model of this form in a time interval $\Delta t_n = [t_k, t_{k+n}]$ of arbitrary length n, as required by the preintegration theory. This is because the torque inputs (the measurements) cannot be decoupled from the quadrotor's orientations (the optimization variables), a fact that was initially noted in VIMO [1]. For this reason, the rotational dynamics of the quadrotor are not considered in VIMO, VID, and HDVIO. Instead, these works obtain the evolution of the orientation of the quadrotor from the gyroscope model. As a result, they introduce inconsistency in the estimation process due to the repeated use of gyroscope measurements, namely, in the dynamics and IMU residuals. Furthermore, their dynamics residuals constrain only the linear dynamics (position and linear velocity) of the quadrotor, while leaving the orientation unconstrained.

In HDVIO2.0 we address these limitations by representing the rotational dynamics of the quadrotor using a continuous-time formulation. Specifically, we employ B-splines as the continuous-time function. The study in [17] demonstrated that B-splines are well-suited for representing quadrotor dynamics. Furthermore, the derivatives of B-splines can be computed efficiently [43], facilitating the use of gradient descent-based optimization methods to optimize the placement of the control points.

1) Continuous-time Representation of Rotational Dynamics: We represent the quadrotor body rates $\boldsymbol{\omega}$ using a B-spline. Specifically, we adopt a uniform time representation of the B-spline [43], which allows using a matrix form formulation for sampling. The B-spline order is denoted by N. Sampling a point from the B-spline depends only on a local segment defined by N control points. The control points are placed at the time $t_i = t_0 + i \cdot \Delta t$, $i \in [0, K]$, where t_0 is the time of the first control point, i is the index of the control point, and Δt is the constant time spacing between consecutive control points. For a given time t, the uniform time representation is defined as $u(t) = s(t) - i$, where $s(t) = \frac{t - t_0}{\Delta t}$ represents the index of the B-spline segment between control points i and $i+1$. The control point i is the leftmost control point affecting the sampling at the time t. For simplicity of the notation, we use u instead of $u(t)$, in some of the equation presented in this section. The quadrotor body rate at the uniform time u is expressed as: $\boldsymbol{\omega}(u) = [\boldsymbol{\omega}_i, \cdots, \boldsymbol{\omega}_{i+N-1}]\boldsymbol{M}^N\boldsymbol{u}$. The matrix $\boldsymbol{M}^N$ is the blending matrix, which is constant and precomputed offline once the B-spline order N is known. The vector $\boldsymbol{u}$ contains the base coefficients, where the j-th entry is equal to u^j. The time derivative are computed as: $\boldsymbol{\omega}^d(u) = [\boldsymbol{\omega}_i, \cdots, \boldsymbol{\omega}_{i+N-1}]a^d\boldsymbol{M}^N\boldsymbol{u}$, where $a^d = \frac{1}{\Delta t^d}$ and d is the derivative order. Specifically, we sample the quadrotor angular accelerations with $d = 1$. We optimize the B-spline control points to fit the quadrotor rotational dynamics model,

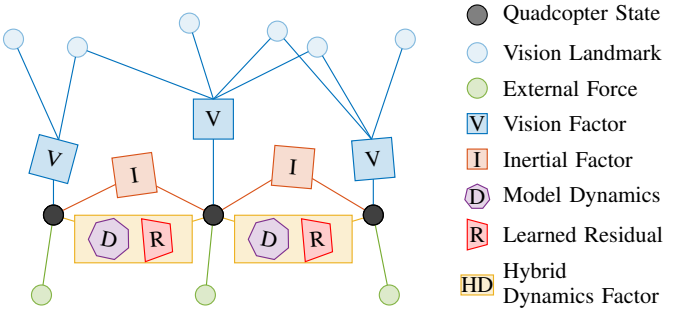

Fig. 2: Factor graph representation of HDVIO2.0 with visual, inertial, and 6-DoF hybrid dynamics factors.

c.f. Eq 1, using the measurement model at time t_k:

$$\hat{\boldsymbol{\tau}}_k^{\mathcal{B}} + \boldsymbol{\tau}_{res_k}^{\mathcal{B}} - \boldsymbol{n}_\tau = \boldsymbol{J}\dot{\boldsymbol{\omega}}^{\mathcal{B}}(u(t_k)) + \boldsymbol{\omega}^{\mathcal{B}}(u(t_k)) \times \boldsymbol{J}\boldsymbol{\omega}^{\mathcal{B}}(u(t_k)). \tag{2}$$

Alternatively, one could use a B-spline to represent the quadrotor's orientations instead of its angular velocities. However, we choose to represent angular velocities due to the faster convergence observed when optimizing the B-spline control points using the measurement model in Eq. 2. An ablation study on the convergence time of the B-spline fitting problem is included in Sec. A of the appendix.

2) Implementation Details: We use a B-spline of order 5 and Δt = 0.01 s. The length of the B-spline is set to 0.1 s, resulting in a B-spline represented by 10 control points. When a new camera frame arrives, new control points are added to the B-spline at the desired sampling rate, starting from the time of the last control point of the previous B-spline up to the timestamp of the newly arrived camera frame. To maintain a fixed-length B-spline, the oldest control points are removed. The new control points are initialized by interpolating the gyroscope measurements at the desired times. At this point, the B-spline optimization begins. Error terms are computed based on Eq. 2, along with their corresponding Jacobians for each available torque measurement. In our experiments, torque measurements are sampled at either 200 Hz (Sec. IV) or 100 Hz (Sec. V). Next, the Hessian matrix is computed using its first-order approximation, as proposed in the Levenberg-Marquardt (LM) algorithm [44]. The LM optimization is then performed, with a maximum of 10 inner-loop iterations and 100 outer-loop iterations. Residuals, Jacobians, and the Hessian matrix are recomputed at each outer-loop iteration. The algorithm terminates early if the solution (i.e., the update to the control point values) falls below a predefined threshold (1e-6 in all our experiments). Finally, angular velocities are sampled from the optimized B-spline at the control command rate and used to compute the dynamics residuals, as described in Sec. III-D. The programming effort to achieve the integration of the proposed continuous-time-based quadrotor dynamics in a VIO system is a key contribution of our work.

C. Estimation Problem Formulation

We implement our hybrid drone dynamics in a sliding-window optimization-based VIO system. An overview of the proposed optimization-based VIO with hybrid drone dynamics, using a factor graph representation, is shown in Fig. 2. The sliding window contains the most recent L keyframes

and K drone states. We set L=10 and K=5. The optimization variables are defined as: $\mathcal{X} = \{\mathcal{L}, \mathcal{X}_\mathcal{L}, \mathcal{X}_\mathcal{B}\}$, where $\mathcal{L}$ consists of the position of the 3D landmarks visible in the sliding window, $\mathcal{X}_\mathcal{L}$ represents the poses of the keyframes: $\mathcal{X}_\mathcal{L} = [\boldsymbol{\zeta}_1, \cdots, \boldsymbol{\zeta}_l], l \in [1, L]$, and $\mathcal{X}_\mathcal{B}$ the poses of the drone: $\mathcal{X}_\mathcal{B} = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_k], k \in [1, K]$. The pose of the l^{th} keyframe is $\boldsymbol{\zeta}_l = [\mathbf{p}^\mathcal{W}_{\mathcal{B}_l}, \mathbf{q}^\mathcal{W}_{\mathcal{B}_l}]$, and the state of the k^{th} drone is $\boldsymbol{x}_k = [\boldsymbol{p}^\mathcal{W}_{\mathcal{B}_k}, \boldsymbol{q}^\mathcal{W}_{\mathcal{B}_k}, \boldsymbol{v}^\mathcal{W}_{\mathcal{B}_k}, \boldsymbol{b}_{a_k}, \boldsymbol{b}_{g_k}, \boldsymbol{f}^\mathcal{B}_{e_k}]$. Keyframes are selected based on heuristics that consider the magnitude of the estimated motion and the number of visual keypoints being tracked. We adopt the keyframe selection strategy proposed in [14]. Visual keypoints are detected only at keyframes. The visual-inertial-dynamics estimation problem is formulated as a joint nonlinear optimization that solves for the maximum a posteriori estimate of $\mathcal{X}$. The cost function to minimize is:

$$\mathcal{L}^{HDVIO2.0} = \sum_{h=0}^{L+K-1} \sum_{j \in \mathcal{J}_h} \left\| \boldsymbol{e}_v^{j,h} \right\|^2_{\boldsymbol{W}_v^{j,h}} + \sum_{k=0}^{K-1} \left\| \boldsymbol{e}_i^k \right\|^2_{\boldsymbol{W}_i^k} + \sum_{k=0}^{K-1} \left\| \boldsymbol{e}_d^k \right\|^2_{\boldsymbol{W}_d^k} + \left\| \boldsymbol{e}_m \right\|^2. \tag{3}$$

The cost function in Eq. 3 consists of the visual residuals $\boldsymbol{e}_v$, inertial residuals $\boldsymbol{e}_i$, dynamics residuals $\boldsymbol{e}_d$, and marginalization residuals $\mathbf{e}_m$. All residuals are weighted according to their measurement noise. The visual residuals are defined as $\boldsymbol{e}_v^{j,h} = \mathbf{z}^{j,h} - h(\mathbf{l}_j^\mathcal{W})$, representing the re-projection error of the landmark $\mathbf{l}_j^\mathcal{W} \in \mathcal{J}_h$, where $\mathcal{J}_h$ is the set of all the landmarks visible from the frame h. The function $h(\cdot)$ denotes the camera projection model, and $\mathbf{z}^{j,h}$ represents the corresponding 2D image measurement. For further details, we refer the reader to [15]. The inertial residuals $\boldsymbol{e}_i$ are computed using the IMU preintegration algorithm described in [12]. The dynamics residuals are detailed in Sec. III-D. The error term $\boldsymbol{e}_m$ represents prior information obtained from marginalization. The marginalization factor encodes information about quantities that fall outside the current sliding window. We follow the marginalization strategy proposed in [15]. This approach distinguishes between variables to marginalize (included in the derivation of the marginalization residual) and variables to drop. The poses of keyframes and 3D points that are connected to keyframes still included in the optimization window are marginalized. Instead, the variables to drop are those not connected to keyframes, such as 3D points visible only from frames outside the sliding window that are not selected as keyframes. Dropping these variables, rather than marginalizing them, preserves the sparsity of the Jacobian in Eq. 3. Our implementation of the sliding-window optimization is based on [15].

We integrate this VIO backend with the visual frontend introduced in [14]. This decision is motivated by the robustness demonstrated by [14], attributed to its semi-direct approach to visual feature tracking and its low computational requirements. This aspect makes it particularly well-suited for VIO applications onboard flying vehicles. The code for this VIO pipeline is publicly available as open-source[1]. We incorporate the proposed 6-DoF hybrid-dynamics model, the previous 3-DoF hybrid-dynamics model [16], as well as the baselines VIMO [1] and VID [2] into this VIO system.

[1]https://github.com/uzh-rpg/rpg_svo_pro_open

D. Dynamics Residuals

Given two consecutive states at times t_k and t_{k+1}, the dynamics motion constraint is:

$$\boldsymbol{e}_d^k = \begin{bmatrix} \boldsymbol{\alpha}^{\mathcal{B}_k}_{\mathcal{B}_{k+1}} - \hat{\boldsymbol{\alpha}}^{\mathcal{B}_k}_{\mathcal{B}_{k+1}} \\ \boldsymbol{\beta}^{\mathcal{B}_k}_{\mathcal{B}_{k+1}} - \hat{\boldsymbol{\beta}}^{\mathcal{B}_k}_{\mathcal{B}_{k+1}} \\ \boldsymbol{\gamma}^{\mathcal{B}_k}_{\mathcal{B}_{k+1}} - \hat{\boldsymbol{\gamma}}^{\mathcal{B}_k}_{\mathcal{B}_{k+1}} \\ \boldsymbol{f}^\mathcal{B}_{e_{k+1}} - \hat{\boldsymbol{f}}^\mathcal{B}_{e_{k+1}} \end{bmatrix}, \boldsymbol{W}_d^k = \begin{bmatrix} {}^{\boldsymbol{P}}\boldsymbol{W}_d^k & \mathbf{0} \\ \mathbf{0} & {}^{\boldsymbol{F}}\boldsymbol{W}_d^k \end{bmatrix}. \tag{4}$$

The quantities $\boldsymbol{\alpha}^{\mathcal{B}_k}_{\mathcal{B}_{k+1}}$, $\boldsymbol{\beta}^{\mathcal{B}_k}_{\mathcal{B}_{k+1}}$, and $\boldsymbol{\gamma}^{\mathcal{B}_k}_{\mathcal{B}_{k+1}}$ are the position, velocity, and orientation change in the time interval $[t_k, t_{k+1}]$:

$$\begin{aligned} \boldsymbol{\alpha}^{\mathcal{B}_k}_{\mathcal{B}_{k+1}} &= \boldsymbol{R}^{\mathcal{B}_k}_\mathcal{W}(\boldsymbol{p}^\mathcal{W}_{\mathcal{B}_{k+1}} - \boldsymbol{p}^\mathcal{W}_{\mathcal{B}_k} - \boldsymbol{v}^\mathcal{W}_{\mathcal{B}_k}\Delta t_k - \frac{1}{2}\boldsymbol{g}^\mathcal{W}\Delta t_k^2) \\ &\quad - \frac{1}{2}\boldsymbol{f}^\mathcal{B}_{e_k}\Delta t_k^2 \\ \boldsymbol{\beta}^{\mathcal{B}_k}_{\mathcal{B}_{k+1}} &= \boldsymbol{R}^{\mathcal{B}_k}_\mathcal{W}(\boldsymbol{v}^\mathcal{W}_{\mathcal{B}_{k+1}} - \boldsymbol{v}^\mathcal{W}_{\mathcal{B}_k} - \boldsymbol{g}^\mathcal{W}\Delta t_k) - \boldsymbol{f}^\mathcal{B}_{e_k}\Delta t_k \\ \boldsymbol{\gamma}^{\mathcal{B}_k}_{\mathcal{B}_{k+1}} &= \boldsymbol{q}^{\mathcal{B}_k}_{\boldsymbol{W}} \otimes \boldsymbol{q}^{\boldsymbol{W}}_{\mathcal{B}_{k+1}}. \end{aligned} \tag{5}$$

The quantities $\hat{\boldsymbol{\alpha}}^{\mathcal{B}_k}_{\mathcal{B}_{k+1}}$, $\hat{\boldsymbol{\beta}}^{\mathcal{B}_k}_{\mathcal{B}_{k+1}}$, $\hat{\boldsymbol{\gamma}}^{\mathcal{B}_k}_{\mathcal{B}_{k+1}}$ are the preintegrated position, velocity, and orientation. We calculate them in the discrete time using Euler numerical integration over the timestep δt:

$$\begin{aligned} \hat{\boldsymbol{\alpha}}^{\mathcal{B}_k}_{i+1} &= \hat{\boldsymbol{\alpha}}^{\mathcal{B}_k}_i + \hat{\boldsymbol{\beta}}^{\mathcal{B}_k}_i\delta t + \frac{1}{2}\boldsymbol{R}(\hat{\boldsymbol{\gamma}}^{\mathcal{B}_k}_i)(\hat{\boldsymbol{f}}^\mathcal{B}_i + \boldsymbol{f}^\mathcal{B}_{res_k})\delta t^2 \\ \hat{\boldsymbol{\beta}}^{\mathcal{B}_k}_{i+1} &= \hat{\boldsymbol{\beta}}^{\mathcal{B}_k}_i + \boldsymbol{R}(\hat{\boldsymbol{\gamma}}^{\mathcal{B}_k}_i)(\hat{\boldsymbol{f}}^\mathcal{B}_i + \boldsymbol{f}^\mathcal{B}_{res_k})\delta t \\ \hat{\boldsymbol{\gamma}}^{\mathcal{B}_k}_{i+1} &= \hat{\boldsymbol{\gamma}}^{\mathcal{B}_k}_i \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\hat{\boldsymbol{\omega}}^\mathcal{B}_i\delta t \end{bmatrix}, \end{aligned} \tag{6}$$

where the initial conditions are $\hat{\boldsymbol{\alpha}}^{\mathcal{B}_k}_{\mathcal{B}_k} = \hat{\boldsymbol{\beta}}^{\mathcal{B}_k}_{\mathcal{B}_k} = 0$ and $\hat{\boldsymbol{\gamma}}^{\mathcal{B}_k}_{\mathcal{B}_k}$ equal to the identity quaternion. The quantity $\boldsymbol{R}(\hat{\boldsymbol{\gamma}}^{\mathcal{B}_k}_i)$ is the rotation matrix representation of $\hat{\boldsymbol{\gamma}}^{\mathcal{B}_k}_i$. The force residual $\boldsymbol{f}^\mathcal{B}_{res_k}$ is constant in the integration interval. In our experiments, the dynamics measurements are sampled at either 200 Hz or 100 Hz, which corresponds to a timestep δt of 5 ms and 10 ms, respectively. The derivation of the covariance of the preintegrated terms is based on the error-state system [45] that describes the evolution of the error state $\delta \boldsymbol{z} = [\delta\boldsymbol{\alpha}, \delta\boldsymbol{\beta}, \delta\boldsymbol{\theta}]^\top$ and noise $\boldsymbol{n} = [\boldsymbol{n}_{f_t}, \boldsymbol{n}_\omega]^\top$, where $\delta\boldsymbol{\theta}$ is a 3-D perturbation around $\hat{\boldsymbol{\gamma}}^{\mathcal{B}_k}$. The discrete-time evolution of the covariance matrix is obtained by linearizing the error-state system in the timestep δt. The weight assigned to the residual, ${}^{\boldsymbol{P}}\boldsymbol{W}_d^k$, is the inverse of this covariance matrix. The quantity $\hat{\boldsymbol{f}}^\mathcal{B}_{e_{k+1}}$ is the external force preintegrated term. Following the derivation proposed in [2], we compute this term as the mean difference between the acceleration measurements and the thrust measurements in the time window $[t_k, t_{k+1}]$: $\hat{\boldsymbol{f}}^\mathcal{B}_{e_{i+1}} = \hat{\boldsymbol{f}}^\mathcal{B}_{e_i} + \boldsymbol{R}(\boldsymbol{\gamma}^{\mathcal{B}_k}_i)(\hat{\boldsymbol{a}}^{\mathcal{B}_i} - \boldsymbol{b}_{a_k} - \hat{\boldsymbol{f}}^\mathcal{B}_i - \boldsymbol{f}^\mathcal{B}_{res_k})$, with $\hat{\boldsymbol{f}}^\mathcal{B}_{e_i} = 0$. The weight ${}^{\boldsymbol{F}}\boldsymbol{W}_d^k$ is obtained using the same covariance propagation schema as described above with noise $\boldsymbol{n} = [\boldsymbol{n}_a - \boldsymbol{n}_{f_t}, \boldsymbol{n}_{\boldsymbol{b}_a}]^\top$. This preintegrated term depends on the accelerometer bias. To avoid repropagating this term each time the accelerometer bias estimate changes, we adopt the

strategy proposed in [12]. Specifically, the preintegration term is corrected by its first-order approximation with respect to the change in the accelerometer bias.

E. Learning Residual Dynamics

The dynamics residual term described above relies on accurately estimating the forces acting on the vehicle. In previous works, modeling aerodynamic effects—such as drag forces—requires knowledge of the vehicle's linear velocity, which is not directly measured but instead forms part of the state to be estimated. As a result, simply employing a state-of-the-art quadcopter dynamics model is not feasible.

In our approach, we have access to rotor speeds and IMU measurements, as these quantities are directly measured. Our goal is to estimate residual forces $\boldsymbol{f}_{res}^{\mathcal{B}}$ and torques $\boldsymbol{\tau}_{res}^{\mathcal{B}}$, which account for aerodynamic effects and model mismatches, including systematic noise, between the commanded or measured thrust and torque and the actual force acting on the robot in the absence of external disturbances. To estimate the residual forces, we propose two temporal convolutional networks (TCN). TCNs have been shown to be as effective as recurrent networks in modeling temporal sequences [46], while requiring less computation. The first TCN is used to predict the residual thrust, and the second TCN is used to predict the residual torque. We found empirically that using two TCNs produces more accurate predictions than using a single TCN to predict both the residual thrust and torque. A TCN architecture consists of four temporal convolutional layers with 64 filters each, followed by three temporal convolutional layers with 128 filters each. A final linear layer maps the output to a 3-dimensional vector representing the learned residual thrust or torque. The network predicting the residual thrust takes a buffer of collective thrust and gyroscope measurements as input. The network predicting the residual torque takes a buffer of torques and gyroscope measurements as input. In both cases, the gyroscope measurements are bias-corrected. The inputs are sampled at 100 Hz and fed into the TCNs as input buffers of 100 ms length, resulting in 10 thrust or torque and 10 gyroscope measurements per buffer. We use the Gaussian Error Linear Unit (GELU) activation function. During training, we model the bias as a random Gaussian variable with zero mean and a standard deviation of 1e-3. At deployment, the current bias estimate is used instead. Given a buffer of measurements over the time interval $\Delta t_{i,j} = t_j - t_i$, the TCNs output the residual thrust $\boldsymbol{f}_{res_i}^{\mathcal{B}}$ and torque $\boldsymbol{\tau}_{res_i}^{\mathcal{B}}$. The residual thrust is added to the thrust inputs $\boldsymbol{f}_{t_k}^{\mathcal{B}}$ with $k \in [t_i, t_j]$ to compute the forces $\hat{\boldsymbol{f}}_k^{\mathcal{B}}$, which account for aerodynamics and robot miscalibration into account. Similarly, the residual torque is added to the torque inputs $\boldsymbol{\tau}_k^{\mathcal{B}}$ to compute the torques $\hat{\boldsymbol{\tau}}_k^{\mathcal{B}}$. The corrected torques are used as measurements in the optimization of the B-spline representing the quadrotor body rates, see Sec. III-B1. The quadrotor body rates are sampled from the B-spline. These body rates, as well as the corrected thrusts, are used inside the preintegration framework, see Sec. III-D, to derive relative velocity, position, and orientation measurements. We train the neural network that predicts the residual thrust to minimize the MSE loss:

$$\mathcal{L}_{\boldsymbol{f}}^{HD} = \frac{1}{M}\sum_{m=1}^{M}\left(\left\|\boldsymbol{\alpha}_{\mathcal{B}_i}^{\mathcal{B}_j} - \hat{\boldsymbol{\alpha}}_{\mathcal{B}_i}^{\mathcal{B}_j}\right\|^2 + \left\|\boldsymbol{\beta}_{\mathcal{B}_i}^{\mathcal{B}_j} - \hat{\boldsymbol{\beta}}_{\mathcal{B}_i}^{\mathcal{B}_j}\right\|^2\right). \quad (7)$$

We train the neural network that predicts the residual torque to minimize the MSE loss:

$$\mathcal{L}_{\boldsymbol{\tau}}^{HD} = \frac{1}{M}\sum_{m=1}^{M}\left\|\boldsymbol{\gamma}_{\mathcal{B}_i}^{\mathcal{B}_j} - \hat{\boldsymbol{\gamma}}_{\mathcal{B}_i}^{\mathcal{B}_j}\right\|^2. \quad (8)$$

where $\boldsymbol{\alpha}_{\mathcal{B}_i}^{\mathcal{B}_j}$, $\boldsymbol{\beta}_{\mathcal{B}_i}^{\mathcal{B}_j}$, and $\boldsymbol{\gamma}_{\mathcal{B}_i}^{\mathcal{B}_j}$ are the ground-truth velocity, position, and orientation changes, and M is the batch size. To learn the aerodynamic effects and systematic noise in the input measurements, the training data is collected under conditions where no external forces act on the drone. Furthermore, our training approach does not require ground-truth force data. The training data can be generated using a Structure-from-Motion pipeline [20, 17] if a motion-capture system is not available. The neural networks are trained on a laptop running Ubuntu 20.04 with an Intel Core i9 2.3 GHz CPU and an Nvidia RTX 4000 GPU. Training is performed using the Adam optimizer with an initial learning rate of 1e-4. The inference runs either on the laptop or on an NVIDIA Jetson TX2, which is the computing platform onboard the quadrotor. The TCN inference runs at ≈180 Hz on an NVIDIA Jetson TX2, which exceeds the required 100 Hz state-update rate of our controllers for agile flight.

IV. Experiments on Benchmark Datasets

In our experiments, we compare our method against HDVIO VIMO, VID, and the same VIO system without the proposed hybrid-dynamics model (hereafter referred to as VIO). Following best practices for evaluating VIO algorithms [47], we use the following metrics: translation absolute trajectory error (ATE_T [m]), rotation absolute trajectory error (ATE_R [deg]), and relative translation and rotation errors. These error metrics are computed after aligning the estimated trajectory using the pose-yaw method [47]. For a detailed description of these metrics, we refer the reader to [47]. In addition to trajectory evaluation, we evaluate the accuracy of force estimation by computing the root mean squared error (RMSE) between the ground-truth and predicted forces.

A. NeuroBEM Dataset

Experimental Setup: In this set of experiments, we evaluate the hybrid dynamics model independently of the full VIO pipeline. Specifically, we evaluate the accuracy of the predicted external force, $\boldsymbol{f}^{\mathcal{B}}$, acting on the quadcopter. For this evaluation, we utilize the NeuroBEM dataset [6], which features data from indoor drone flights at speeds of up to 65 km/h. The dataset includes rotor speeds (from which thrust and torque measurements are derived), IMU measurements, and ground-truth force data. We use the provided training sequences to train our learned dynamics model, and the provided testing sequences to evaluate its performance, as well as the performance of the baseline methods. In the testing sequences, 30% of the trajectories are entirely unseen during

TABLE I: Comparison in terms of RMSE of the force, F, and torque, M, estimates on the test set of the NeurBEM dataset. Polyfit, NeuroBEM, and HDVIO2.0 are data-driven methods. Quadratic fit and BEM are first-principles methods. Our method performs remarkably well given it has no information about the velocity or orientation of the vehicle and only falls short of the NeuroBEM method which has access to the ground-truth vehicle state. The values for baseline methods are taken from [6]. In bold are the best values, and in underlined are the second-best values.

Model	Inputs	F_{xy} [N]	F_z [N]	M_{xy} [Nm]	M_z [Nm]	F [N]	M [Nn]
Quadratic Fit	thrust	1.536	1.381	0.104	0.033	1.486	0.087
BEM [6]	full state	0.803	1.265	0.090	0.017	0.982	0.074
PolyFit [7]	full state	0.453	0.832	<u>0.027</u>	0.008	0.606	<u>0.022</u>
NeuroBEM [6]	full state	**0.204**	**0.504**	**0.014**	**0.004**	**0.335**	**0.012**
HDVIO2.0 (Ours)	dynamics+gyro	<u>0.402</u>	<u>0.672</u>	**0.014**	<u>0.006</u>	<u>0.491</u>	**0.012**

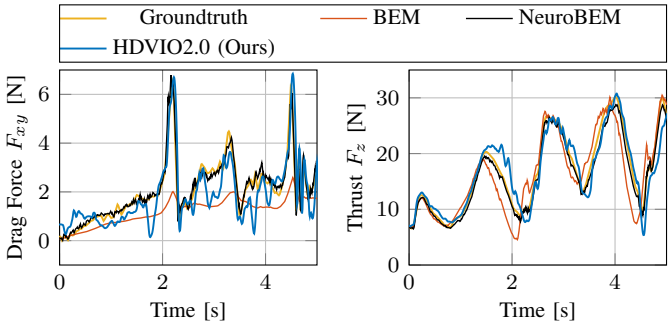

Fig. 3: This figure illustrates the results shown in Tab. I by exemplarily showing the force estimates on a very fast trajectory from the NeuroBEM dataset. Our HDVIO2.0 clearly outperforms the state-of-the-art first-principle model BEM and is able to model aerodynamic effects accurately on short timescales.

training, while the remaining 70% differ in speed and size. We compare the force estimation accuracy of our learned dynamics model to several state-of-the-art baselines:

- Quadratic Fit: The model used in VIMO and VID.
- BEM: A first-principles model that computes forces and torques acting on a propeller by integrating over infinitesimal area elements of the propeller [37, 6].
- PolyFit: A data-driven model that relies on polynomial basis functions to capture drone dynamics [7].
- NeuroBEM: A hybrid model that augments the BEM model with a learning-based component [6].

All baselines, except for the Quadratic Fit model, require the vehicle state as input, including linear and angular velocities. Notably, our method only requires thrust, torque, and IMU measurements.

Evaluation: The results are summarized in Tab. I. The NeuroBEM method, which has access to the full robot state, achieves the best performance. However, our HDVIO2.0 outperforms the BEM model in terms of residual thrust estimates by a factor of three and the PolyFit model by a factor of two. This is further illustrated in Fig. 3, which shows the ground-truth forces alongside the forces estimated by NeuroBEM, BEM, and our HDVIO2.0 during the first five seconds of a high-speed flight. In this flight, the quadrotor accelerates to 15 m/s while following a lemniscate track. The performance of our HDVIO2.0 is remarkable as it achieves this performance without access to ground-truth state information, such as the vehicle's linear or angular velocity, which the baselines require. Moreover, our HDVIO2.0 achieves very

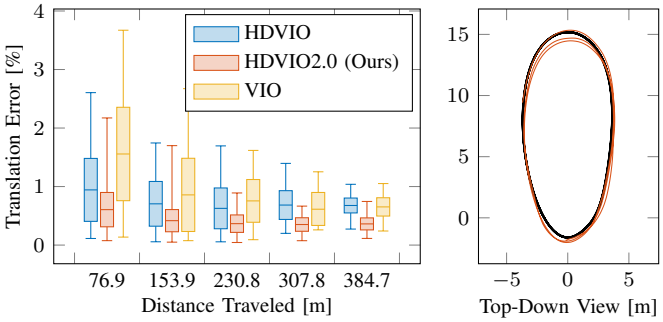

Fig. 4: The performance of VIO, HDVIO and HDVIO2.0 is compared on the *Egg* 8 m/s trajectory. In the right plot, the ground truth is depicted in black and the trajectory estimated by HDVIO2.0 in red.

similar accuracy as NeuroBEM in estimating torques. From this experiment, we conclude that our learning-based component effectively captures the aerodynamic forces acting on the quadrotor, demonstrating its suitability for integration into a VIO pipeline. Additionally, the ability to estimate aerodynamic forces with such precision using only a history of thrust, torque and gyroscope measurements is an interesting and significant finding in itself.

B. Blackbird Dataset

Experimental Setup: In this set of experiments, we evaluate our system and the baseline methods on the Blackbird dataset [18]. The dataset provides rotor speed measurements, which we use to compute mass-normalized collective thrust measurements, recorded onboard a quadrotor flying within a motion-capture system. Additionally, the dataset includes IMU measurements and, in some sequences, photorealistic images of synthetic scenes. The Blackbird dataset comprises 18 diverse trajectories with speeds ranging from 0.5 m/s to 9.0 m/s. Since the dataset does not feature external disturbances, we focus on evaluating the accuracy of pose estimates. Among the trajectories with available camera images, we select six representative sequences for testing: *Bent Dice*, *Clover*, *Egg*, *Mouse*, *Star*, and *Winter*. The remaining 80 % of the trajectories, corresponding to approximately 2 hrs of flight data, are used for training, and 20 % are reserved for validation of our neural networks. To assess the generalization capability of our system, we also train our network on a reduced training dataset containing only trajectories with speeds up to 2 m/s. This allows us to evaluate the performance of our method on higher-speed trajectories beyond those seen during training.

Evaluation: We present the ATE_T and ATE_R results for the testing sequences in Table II. Our approach consistently outperforms VIO baseline, VIMO, VID, and HDVIO demonstrating the effectiveness of our 6-DoF hybrid drone model. The performance improvement becomes more pronounced at higher speeds, achieving an improvement of the ATE_T of 57 % and 27 % compared to VIO and HDVIO respectively, at the maximum velocity of 8 m/s. This result is explained by the fact that our method includes the learned drag forces as measurements in the dynamics motion constraint. In Fig. 4, we provide the relative translation error alongside a top-down view of the trajectories estimated by HDVIO2.0 and the ground truth. Notably, our system continues to outperform the

TABLE II: Evaluation of the trajectory estimates in the Blackbird dataset. HDVIO2.0* (ours) is trained on a reduced training set, with speeds up to 2 m/s to evaluate generalization performance. In bold are the best values, and in underlined are the second-best values. Our method, either using the full training dataset or the reduced one, outperforms the baselines in all the sequences.

Trajectory Name	$v_{\max}$ [m/s]	Evaluation Metric: ATE_T [m] / ATE_R [deg]					
		VIO	VIMO	VID	HDVIO	HDVIO2.0 (Ours)	HDVIO2.0* (Ours)
Bent Dice	3	0.20 / 1.78	0.31 / 1.53	0.25 / 1.18	0.21 / 1.53	0.18 / **0.84**	**0.16** / 0.90
Clover	5	0.90 / 3.52	0.88 / 3.66	0.83 / 2.48	0.60 / 2.08	0.49 / 1.99	**0.48** / **1.93**
Egg	5	1.07 / 1.54	0.75 / 1.34	0.81 / 1.61	0.59 / 1.21	**0.56** / **1.07**	**0.56** / 1.20
Egg	6	1.40 / 2.35	0.98 / 4.89	1.10 / 2.42	0.83 / 1.62	0.69 / **1.60**	**0.58** / 1.86
Egg	8	1.79 / 4.55	1.57 / 3.69	1.47 / 4.84	1.06 / 2.89	**0.77** / 2.61	1.44 / **2.51**
Mouse	5	1.10 / 4.54	0.76 / 2.14	0.54 / 2.10	0.36 / 1.40	**0.22** / **0.99**	0.34 / 1.01
Star	1	0.17 / 0.78	0.18 / 1.05	0.18 / 0.54	0.16 / 0.58	**0.09** / 0.74	0.10 / **0.49**
Star	3	0.62 / 3.50	0.43 / 1.38	0.50 / 2.93	0.38 / 1.40	**0.19** / **0.93**	0.27 / 0.96
Winter	4	0.97 / 2.92	0.69 / 2.46	0.66 / 2.05	0.57 / 1.54	**0.12** / **0.78**	0.51 / 2.02

TABLE III: Ablation study of the effect of learning residuals (LR) on the trajectory estimates for the Blackbird dataset. In bold are the best values.

Traj.	$v_{\max}$ [m/s]	ATE_T [m]		ATE_R [deg]	
		w/ LR	w/o LR	w/ LR	w/o LR
Clover	5	**0.49**	0.71	**1.99**	2.24
Egg	6	**0.69**	0.92	**1.60**	1.84
Mouse	5	**0.22**	0.43	**0.99**	1.86

TABLE IV: RMSE of the external force estimates. The external force originates either from a pulling rope (*sequence 17* of the VID dataset) or from an external load (*sequence 16* of the VID dataset). HDVIO2.0 drastically improves the force estimation along the z axes. In bold are the best values.

Method	Pulling rope		External Load
	F_z [N]	F [N]	F_z [N]
VIMO	1.73	1.08	0.81
VID	1.96	1.12	0.45
HDVIO	0.55	0.65	**0.34**
HDVIO2.0 (Ours)	**0.39**	**0.59**	**0.34**

baselines across almost all sequences, even when the networks are trained on the reduced dataset (containing trajectories with speeds up to 2 m/s), as shown in the last column of Table II. This result highlights that HDVIO2.0 is capable of generalizing to velocities up to 4x higher than those present in the training data. The fact that the accuracy of HDVIO2.0 and HDVIO2.0* is higher than HDVIO in most of the sequences highlights the benefit of incorporating the rotational dynamics in the dynamics residuals. In Table III, we show an ablation study to evaluate the effect of learning residual thrusts and torques. The results on the faster sequences of the Blackbird dataset show that including learned residuals in the drone dynamics improves the trajectory estimates.

C. VID Dataset

Experimental Setup: In this set of experiments, we evaluate the ability of our system to estimate external forces acting on the quadrotor and test the performance of our learned component in scenarios where ground-truth data from a motion capture system is unavailable. For this purpose, we use the VID dataset [2], which includes visual, inertial, and actuation inputs, as well as ground-truth force measurements. The dataset contains data recorded onboard a quadrotor flying both indoors, in an office room equipped with a motion-capture system, and outdoors, in a parking area. We use the provided rotor speed measurements to compute the thrust and torque values. The indoor sequences, which include portions with ground-truth force data, are used to evaluate our system's capability to estimate external forces. The outdoor sequences are used to validate our learned module when ground-truth training data for position, velocity, and orientation is obtained from an offline visual-inertial SLAM system [17] rather than a motion-capture system. Since the outdoor sequences do not include ground-truth force measurements, we focus on the estimation of the drone poses. Since the quadrotor mass

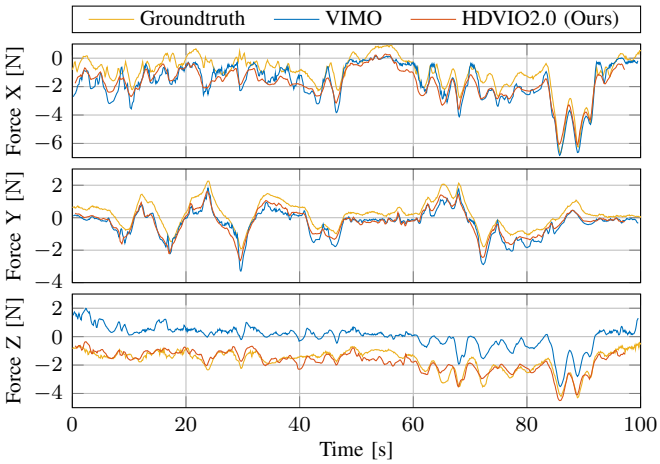

Fig. 5: Comparison of the external force estimate in the *sequence 17* of the VID dataset. In this sequence, the drone is attached to an elastic rope. The other end of the rope is attached to a force sensor. HDVIO2.0 drastically improves the force estimation along the z-axis.

differs between the indoor and outdoor sequences, we train different neural networks for the indoor drone configuration and the outdoor configuration. For the indoor configuration, we train our neural networks using sequences without external perturbations and ground truth from the motion capture system. These sequences consist of hover, circle, and figure 8 trajectories, amounting to only 6 min of flight data. Of this data, 80% is used for training and 20% for validation. For the outdoor configuration, the sequences include circle, figure 8, and rectangle trajectories, and the ground truth is obtained by the offline visual-inertial SLAM system. We use one figure 8 and one rectangle trajectory for testing. The remaining data, totaling 11 min of flight time, is split into 80% for training and 20% for validation.

TABLE V: We demonstrate the performance of HDVIO2.0 in a setting where the training data for the learning-based component is obtained from a vision-based SLAM system in the outdoor sequences of the VID dataset. The flown trajectories are at low speeds, below 3 m/s, allowing all the methods to perform well, with HDVIO2.0 showing the highest accuracy. In bold are the best values.

Method	Figure 8		Rectangle	
	ATE_T [m]	ATE_R [deg]	ATE_T [m]	ATE_R [deg]
VIO	1.89	4.02	2.09	2.01
VIMO	1.67	3.70	1.89	1.92
VID	1.84	3.50	1.99	2.49
HDVIO	1.48	3.70	1.72	1.68
HDVIO2.0 (Ours)	**1.41**	**0.71**	**1.69**	**1.61**

Evaluation: We compare the external force estimates of VIMO, VID, HDVIO, and HDVIO2.0 in an indoor sequence where the drone is attached to an elastic rope, with the other end connected to a force sensor, and in a sequence where an unknown external load is attached to the drone, as summarized in Table IV. The force estimates are aligned with the motion-capture reference frame using the *posyaw* alignment method [47]. HDVIO2.0 significantly outperforms the baselines, VIO and VID, while the performance increase compared to HDVIO is smaller. As shown in Fig. 5, our method significantly improves force estimation along the z-axis. This improvement is attributed to our neural network. We believe that the proposed network has learned to compensate for a systematic residual error in the thrust inputs, likely caused by inaccuracies in the thrust coefficients used to compute the collective thrust from rotor speed measurements. In these sequences, the slow vehicle motion and the textured environment simplify the pose estimation problem. Consequently, VIO, VIMO, VID, HDVIO and our method achieve similar pose estimation performance, with a ATE_T of 0.02 m and of 0.10 m, respectively.

The Table V presents the evaluation of pose estimates in the outdoor sequences. Since the trajectories are flown at low speeds below 3 m/s, all three methods demonstrate good performance, with HDVIO2.0 achieving the highest accuracy. Remarkably, including the rotational dynamics in the estimation process improves the rotation error by 80% compared to the previous system HDVIO in the sequence *Figure 8*. This experiment highlights that our learning-based dynamics model can be effectively trained without relying on an external motion-capture system.

V. Flights in Continuous Wind

In these experiments, we demonstrate that HDVIO2.0 can estimate continuous external disturbances, such as continuous wind, outperforming all the state-of-the-art methods. To this end, we fly a quadrotor in a controlled wind field, as illustrated in Fig.6. Details about the quadrotor platform can be found in [48]. The platform is equipped with an onboard Intel RealSense T265[2] camera, which provides camera and IMU measurements. While the camera includes stereo fisheye sensors, we only use images from the left camera. Rotor speed measurements are not available on this quadrotor platform.

[2]https://www.intelrealsense.com/wp-content/uploads/2019/09/Intel_RealSense_Tracking_Camera_Datasheet_Rev004_release.pdf

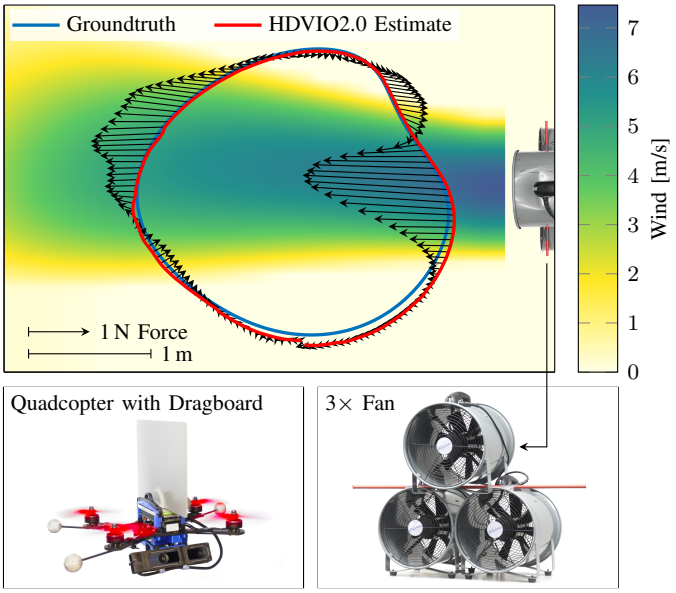

Fig. 6: A quadrotor with a dragboard attached is flown on a circular trajectory through a wind field generated with three industrial fans. Our HDVIO2.0 is used to estimate the position of the drone (shown in red) and the external disturbance force (black arrows) acting on the vehicle. The ground-truth position of the vehicle is shown in blue.

Instead, we use the collective thrust commands output by the MPC controller [48] to control the vehicle. The quadrotor is flown in a motion-capture system that provides pose data at an update rate of 200 Hz. We conduct experiments using two quadrotor configurations: one with the vehicle in its nominal state and another with a 22 cm×16 cm dragboard attached. The dragboard is mounted such that its normal aligns with the drone's body y-axis. This attachment increases drag in the y-direction by more than a factor of two, making the vehicle significantly more sensitive to crosswinds.

1) Wind Generation: To generate a wind field, we placed three axial fans (Ekström 12 inch, as shown in Fig. 6) in an office-like room measuring 8x10 m. The fans were positioned at a height of 1.6 m above the ground and angled slightly inwards to ensure high wind speeds across a virtual tube in front of them. Each fan has an advertised air circulation rate of $1.3\,\mathrm{m}^3/\mathrm{s}$ and produces a measured wind speed of up to 8 m/s at its front grill. To quantitatively evaluate the performance of our method, we require ground-truth data for the external wind forces. In the following, we describe the procedure used to obtain this ground-truth data.

2) Wind Speed Map: In the first step, we measured the wind produced by our experimental setup. Local wind speeds were recorded at 50 points within the wind cone in front of the fans, with a higher sampling density in regions where wind speed changes more rapidly. Measurements were taken using a hand-held anemometer (Basetech BS-10AN), and the position of the anemometer was tracked using the motion-capture system. To generate the ground-truth wind speed map shown in Fig. 6, a smoothing spline was fitted to the recorded data.

3) Lift and Drag Coefficients: We compute the wind force based on the wind speed map. The aerodynamic forces acting on a quadrotor are primarily determined by three components:

TABLE VI: Trajectories estimates in drone flights in a wind field with wind gusts up to 25 km/h. We use (d) to indicate that a drag board was attached to the drone. In bold are the best values.

Method	ATE_T [m] / ATE_R [deg]			
	Circle (d)	Circle	Lemniscate (d)	Lemniscate
VIO	0.07 / 2.02	0.06 / 1.21	0.38 / 2.39	0.27 / 2.44
VIMO	0.10 / 1.80	0.08 / 1.19	0.34 / 2.93	0.32 / 1.93
VID	0.10 / 2.31	0.06 / 1.37	0.53 / 2.39	0.28 / 2.05
HDVIO	0.07 / 2.06	0.06 / 1.17	0.30 / 2.81	0.20 / 1.84
HDVIO2.0 (Ours)	**0.05 / 1.02**	**0.04 / 1.02**	**0.21 / 2.34**	**0.14 / 1.53**

TABLE VII: RMSE of the external force estimates. The external force originates from continuous wind, with gusts up to 25 km/h. We use (d) to indicate that a dragboard was attached to the drone. In bold are the best values.

Method	F [N]			
	Circle (d)	Circle	Lemniscate (d)	Lemniscate
VIMO	0.62	0.26	0.52	0.44
VID	0.56	0.33	0.73	0.51
HDVIO	0.54	0.23	0.44	0.33
HDVIO2.0 (Ours)	**0.51**	**0.23**	**0.39**	**0.31**

the body/fuselage drag f_d^{fus}, the induced drag from the propellers f_d^{ind}, and the lift and drag incurred by the flat-plate drag board attached to the top of the quadrotor, f_l^{brd} and f_d^{brd}. The magnitudes of these forces can be approximated using established aerodynamic models [6, 8, 49]:

$$\begin{aligned} f_d^{\text{fus}} &= 0.5\,\rho\,A^{\text{fus}}\,c_d^{\text{fus}}\,v_{\text{rel}}^2 \\ f_d^{\text{ind}} &= k\,v_{\text{rel}} \\ f_{l|d}^{\text{brd}} &= 0.5\,\rho\,A^{\text{brd}}\,c_{l|d}^{\text{brd}}(\alpha)\,v_{\text{rel}}^2\ , \end{aligned} \tag{9}$$

where ρ is the air density, A is a surface area, v_{rel} the relative air speed, α is the angle of attack of the dragboard, k is the propeller drag coefficient, and $c_{l|d}$ are the lift and drag coefficients of the fuselage and dragboard. The relative air speed is calculated as the norm of the relative velocity, which is the sum of the vehicle's ego motion and the wind velocity.

In this model, the fuselage is approximated as a square prism with an angle-of-attack-independent drag coefficient of $c_d^{\text{fus}} = 2.0$ [50]. For the flat-plate wing, we employ a simplified high-angle-of-attack model widely used in propeller modeling [37, 49], which also aligns well with experimental data for flat-plate wings [51]:

$$c_l^{\text{brd}}(\alpha) = \sin(2\alpha)\ , \qquad c_d^{\text{brd}}(\alpha) = 2\sin^2(\alpha).$$

To validate the predicted forces for the fuselage and drag board described in Eq. (9), the quadrotor was mounted on a load cell[3]. At a wind speed of 7 m/s, the measured lift and drag forces were within 10 % of the calculated values. Additionally, the linear propeller drag coefficient was determined to be $k = 0.145$ Ns/m.

4) Wind Forces: Our method distinguishes between aerodynamic effects (such as body drag and induced drag) and external forces. To obtain the ground truth for the disturbance caused by the wind, we calculate the forces acting on the quadrotor with the fans turned on and subtract the forces calculated when the fans are turned off.

A. Dataset Collection

The training data consists of approximately 10 min of random trajectories flown without wind. We use 80% of this data for training the neural network and the remaining 20% for validation. We exclusively use *random* trajectories, generated by sampling position data with a Gaussian Process. This method ensures diverse training data and helps prevent overfitting to specific trajectories. The test data is collected with the quadrotor flying in a wind field with gusts reaching up to 25 km/h. The test trajectories include a circle and a lemniscate, both with a maximum speed of 2 m/s. Additionally, we recorded a second dataset that features the same training, validation, and test trajectories, but with the quadrotor equipped with a drag board. In this setup, the drag and the external force due to wind gusts are increased, emphasizing the advantage of HDVIO2.0 over the baselines.

1) Evaluation: We present the estimation of the external force due to wind gusts in Fig. 7 and Table VII. Since the wind gusts impact the quadrotor along the y-axis of the world reference frame, we display both the y-component of the estimated force and the force norm. The external forces are estimated in the quadrotor's body frame, then aligned to the world frame (which corresponds to the motion-capture reference frame) using the ground-truth orientations. This alignment allows for a direct comparison between the estimates of our method and those from the baselines. Notably, HDVIO2.0 is able to accurately predict the wind gusts when the quadrotor enters the wind field. This is evident in Fig. 7 from the fact that our method accurately captures the peaks of the wind force.

As noted by the authors [1], the measurement model in VIMO, when dealing with continuous external disturbances, introduces an inconsistency in the estimation of the accelerometer bias, leading to decreased motion estimate accuracy. We show in Fig. 8 the accelerometer bias estimated by the VIO algorithm, VIMO, VID, and our method for a sequence in which the quadrotor, equipped with the drag board, flies a circle trajectory. Although we do not have access to the ground-truth accelerometer bias, we consider the one estimated by the VIO algorithm to be a good approximation of the true value, given that VIO achieves very high performance in this sequence (see Table VI) due to the high number of visual features being tracked. The bias estimate of HDVIO2.0 closely tracks the VIO estimate, while VIMO and VID estimate diverges along the x-axis and y-axis to an incorrect value. We include the position and orientation absolute trajectory errors in Table VI. In all four sequences, the rich-texture environment simplifies the pose estimation problem, resulting in accurate pose estimates for all the algorithms. However, improvements in the rotation estimates, driven by the inclusion of rotational dynamics in the estimation process, are evident.

Table VIII reports the runtime of HDVIO2.0 on two platforms: a laptop running Ubuntu 20.04 with an Intel Core i9 2.3 GHz CPU and an Nvidia RTX 4000 GPU, and an NVIDIA Jetson TX2 (the onboard computer used during flight). We break down the runtime into two components corresponding to

[3]https://www.ati-ia.com/products/ft/ft_models.aspx?id=Mini40

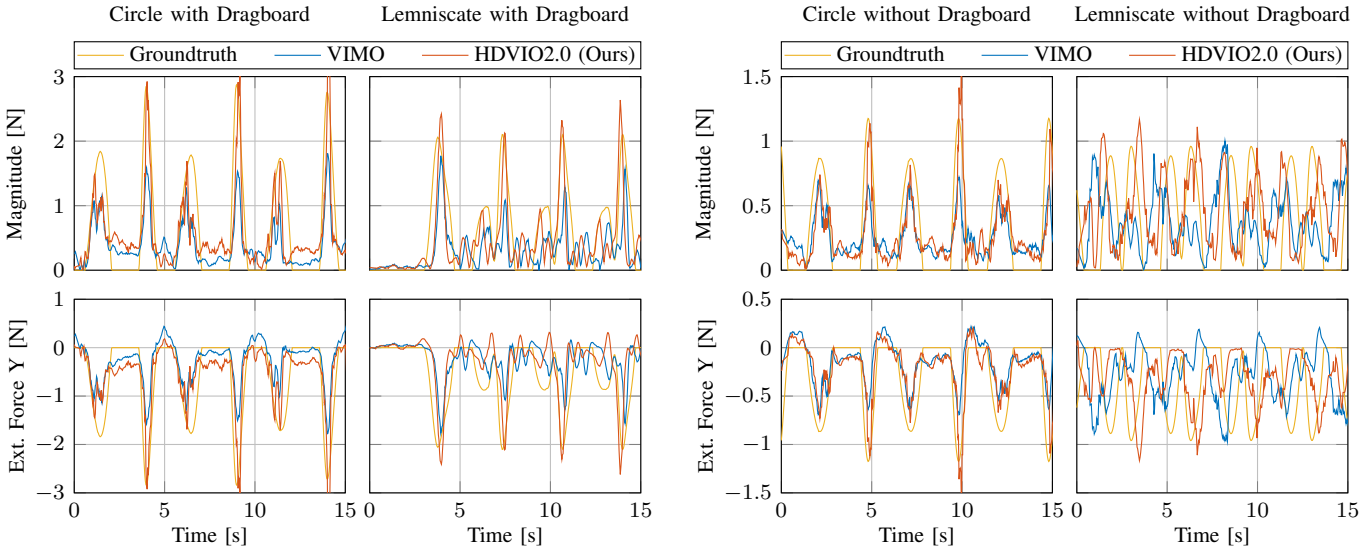

Fig. 7: Wind disturbance estimates. The magnitude and the y-axis component of the wind force estimated by HDVIO2.0 and VIMO. Left: drone equipped with a dragboard. Right: standard drone configuration. In all the plots, it is visible that HDVIO2.0 achieves more accurate force estimates than VIMO.

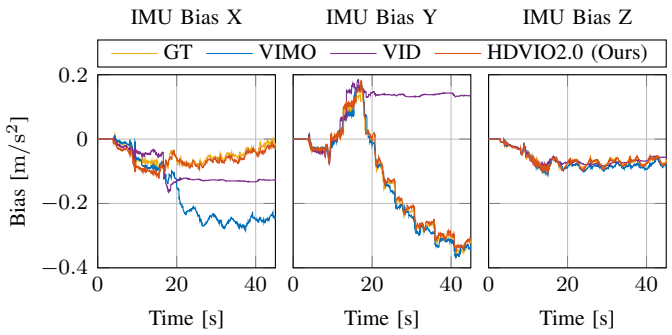

Fig. 8: Accelerometer bias estimates. The ground-truth (GT) bias is obtained from the VIO system. The estimates of our HDVIO2.0 match the ground-truth values, while VIMO and VID estimates diverge along the x-axis and the y-axis.

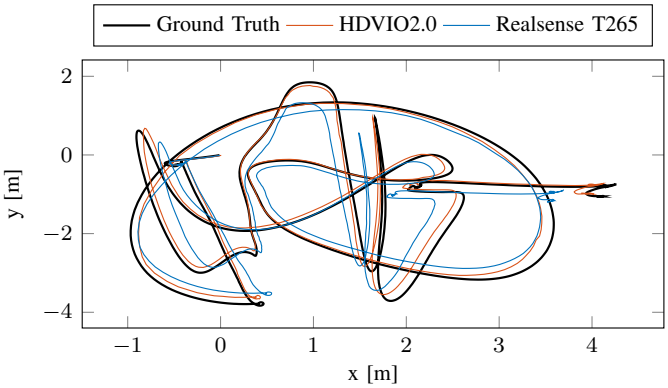

Fig. 9: Closed-loop experiment where the quadrotor state estimated by HDVIO2.0 is provided to the flight controller. The figure shows the top view of the trajectory estimated by HDVIO2.0, the Realsense T265, and the ground truth from a motion capture system. Our method outperforms the commercial stereo-based SLAM from the Realsense T265.

TABLE VIII: Runtime [ms] of HDVIO2.0 . The backend runtime includes the B-spline optimization and network inference.

HDVIO2.0 thread	Runtime [ms] Laptop	Nvidia Jetson TX2
Frontend	5.4	14.9
Backend	21.3	31.8

the frontend and backend threads of HDVIO2.0. The frontend runtime measures the time required for visual feature tracking and detection. The backend runtime is the time from the arrival of a new camera frame to the estimation of its pose, including the setup and solution of the optimization problem defined in Eq. 3. This includes both the B-spline optimization and neural network inference. HDVIO2.0 achieves real-time performance, namely the pose of a camera frame is estimated before the next frame is available (given a 30 FPS camera as in all our experiments) on both the laptop and the NVIDIA Jetson TX2.

VI. Closed-loop Control Flights

In this section, we demonstrate the performance of HDVIO2.0 running onboard a quadrotor within a closed-loop control system. We use the same drone described in Section V and adopt the controller from [48].

HDVIO2.0 provides state estimates at the camera frame rate (30 Hz in our experiments), which is insufficient for high-frequency feedback control that requires estimates at a frequency higher than 100 Hz. To bridge this gap, we integrate HDVIO2.0pose estimates into an extended Kalman filter (EKF). The EKF state includes the quadrotor's full 6-DoF pose, velocity, and IMU biases. It is propagated using the IMU measurements and updated with the pose estimates from HDVIO2.0. As a result, the EKF provides state estimates at 200 Hz, matching the IMU rate. These high-frequency estimates are then supplied to the controller.

Figure 9 compares the trajectory estimated by HDVIO2.0, the onboard Intel RealSense T265, and ground truth. Notably, HDVIO2.0 outperforms the commercial SLAM system running onboard the T265. Indeed, HDVIO2.0 achieves an ATE_T of 0.18 m, while the Intel RealSense T265, which employs a stereo camera configuration, achieves 0.29 m.

A clip of this experiment is included in the supplementary

video.

VII. Discussion

A key component of HDVIO2.0 is the learning-based drone dynamics model, which estimates residual aerodynamic forces using only thrust, torque, and gyroscope measurements, sensors commonly available on most commercial drone platforms. This learning-based module outperforms state-of-the-art first-principles quadrotor models and achieves performance comparable to NeuroBEM, despite not having access to the ground-truth quadrotor state as these methods do.

Another important advantage is that it does not necessarily require ground-truth forces for supervised training. Instead, the training minimizes the error between predicted and observed changes in position, velocity, and orientation, which can be obtained from SLAM [17] or Structure from Motion [20], avoiding the need for motion-capture systems. Table V shows successful training using only SLAM-derived supervision. In this way, the data collection is simplified to the extent that even a human pilot can control the drone, as expert pilots typically issue collective thrust commands alongside desired body rates [52]. This simple training data collection mitigates a limitation of our drone dynamics model: while a single dynamics model is tailored to a specific drone, recording new data to train a model for a different drone is straightforward.

HDVIO2.0 drone dynamics model demonstrates strong generalization capabilities to velocities and trajectories unseen in the training data.

Generalization to unseen velocities is shown in Table II where HDVIO2.0* is trained with trajectories that contain speeds only up to 2 m/s, compared to HDVIO2.0 which is trained on speeds up to 9 m/s. When tested on the full range of speeds (up to 8 m/s), HDVIO2.0* achieves comparable results to HDVIO2.0 and still outperforms the baselines.

Generalization across different trajectory types is shown in various experiments. In the NeuroBEM dataset (see Sec. IV-A), 30% of the test trajectories were entirely unseen during training, while the remaining 70% differ in speed and size. Fig. 5 shows the force estimates of HDVIO2.0 on an unseen trajectory from the VID dataset. In Sec. V-A, the network is exclusively trained on random trajectories. These advantages highlight HDVIO2.0 robustness and versatility in real-world applications.

Furthermore, HDVIO2.0 demonstrates robustness to VIO failures. In Sec. IV-B, HDVIO2.0 achieves the largest improvements, with reductions of 57% and 43% in translation and rotation error compared to the VIO on the fastest trajectory, Egg 8 m/s (see Table II and Fig 4). In this scenario, motion blur and fast yaw changes make feature tracking difficult, causing the VIO system to accumulate significant drift.

In this work, as in the baselines, the model is assumed to remain fixed during a flight. Changes in actuation inputs (e.g., hardware degradation) are treated as external forces. A promising direction for future work would be to train the neural network to estimate these model changes as residual forces, overcoming the challenge of generating suitable training data.

The core idea behind HDVIO2.0 learning residual dynamics to derive motion constraints for inclusion in a VIO backend, is not specific to quadrotors and can be applied to other robotic platforms. The feasibility of learning residual dynamics has been demonstrated on diverse robot types, including rigid [53] and soft [54] robotic arms, and legged robots [55]. Since VIO backends are robot-agnostic, incorporating motion constraints based on the learned dynamics is generally applicable. However, specific system design choices, including the selection of the network architecture, may need to be adapted based on the specific dynamics of the target robot. For instance, while TCNs have proven effective for modeling residual dynamics [6], motion priors [56], and control commands [57] in quadrotors, they may not be optimal for other types of robots. Assessing their application for different robotic platforms is an interesting direction for future work.

VIII. Conclusion

In this work, we introduce a novel method for modeling 6-DoF quadrotor dynamics in visual-inertial odometry systems. Our dynamics model integrates a first-principles quadrotor model with a learning-based component that captures unmodeled effects, such as aerodynamic drag. The proposed method addresses the limitations of the state-of-the-art systems, VIMO, VID, and HDVIO, improving the accuracy of motion estimation and external force estimation.

Our learning-based component demonstrates strong generalization capabilities to trajectories and speeds beyond those present in the training dataset. Furthermore, an evaluation of residual force estimation accuracy reveals that our learning-based approach outperforms first-principles models, even those with access to the full state of the quadrotor. Controlled experiments in windy conditions further validate our hybrid dynamics model's ability to accurately predict forces acting on the quadrotor due to continuous wind.

HDVIO2.0 enhances the safety of autonomous drone operations in challenging scenarios, such as high-speed flights and operations in windy environments. With the growing integration of drones into everyday applications, these improvements are relevant.

Appendix

A. Representation of Rotational Dynamics

In HDVIO2.0 we represent the quadrotor's angular velocities with a B-spline. Alternatively, it is possible to use a B-spline to represent the quadrotor's orientations. Our choice to represent angular velocities results from the faster convergence observed in the optimization problem when optimizing the B-spline control points according to the measurement model defined in Eq. 2. To demonstrate this, we conducted an ablation study comparing the convergence time of the B-spline fitting optimization problem when the B-spline represents either the quadrotor's orientations (*RotBSpl*) or its angular velocities (*VelBSpl*). The results of this study are presented in Table IX. We selected a sequence from the NeuroBEM dataset where the drone follows a random trajectory. This trajectory was split into sequential, non-overlapping segments of durations 0.5, 1, and 2 s. For each segment, we solved the B-spline fitting optimization problem using all available torque inputs

as measurements (the NeuroBEM dataset provides control commands at 400 Hz). The control points for *RotBSpl* were initialized via linear interpolation of the quadrotor's orientations provided by the dataset, while those for *VelBSpl* were initialized using linear interpolation of the gyroscope measurements. We evaluated different B-spline orders and control point spacings. The experiments are run on a laptop running Ubuntu 20.04 and equipped with an Intel Core i9 2.3 GHz CPU. As shown in Table IX, the B-spline representing angular velocities consistently achieved faster convergence than the one representing orientations, regardless of B-spline order, control point spacing, or segment length.

TABLE IX: Convergence time in [ms] of the optimization problem that optimizes the B-spline control points according to the quadrotor's rotational dynamics model (see Eq. 2). We compare two B-spline formulations. *RotBSpl*: the control points represent the quadrotor orientation in *SO(3)*. *VelBSpl*: the control points represent the quadrotor angular velocity. *VelBSpl* is the B-spline representation used in HDVIO2.0 because of its fast convergence time. We report the average convergence time computed over all the trajectory segments. In **bold** the smallest convergence time. – indicates an infeasible configuration caused by a trajectory segment that is too short relative to the chosen B-spline order and control point spacing.

B-spline Order	**Ctrl. Points Spacing** [ms]	**Trajectory Segment Length** [s]					
		0.5 [s]		1.0 [s]		2.0 [s]	
		RotBSpl	*VelBSpl*	*RotBSpl*	*VelBSpl*	*RotBSpl*	*VelBSpl*
5	100	–	–	500.8	**12.4**	1498.2	**40.3**
	50	272.3	**7.6**	707.5	**21.4**	1720.9	**34.9**
	20	431.6	**7.9**	914.0	**17.4**	1881.3	**29.3**
	10	536.3	**3.6**	1082.0	**9.3**	2243.5	**16.2**
	5	702.6	**8.3**	1384.8	**19.6**	2795.7	**28.8**
6	100	–	–	509.3	**6.2**	1676.4	**16.9**
	50	278.8	**3.4**	805.2	**12.1**	1878.4	**20.9**
	20	494.4	**11.3**	1028.4	**16.3**	2153.0	**37.1**
	10	643.6	**9.6**	1362.6	**16.2**	2785.5	**50.4**
	5	874.6	**14.4**	1724.7	**27.9**	3603.8	**69.7**
7	100	–	–	518.6	**4.5**	1897.7	**18.4**
	50	275.8	**2.8**	907.5	**10.1**	2131.2	**19.4**
	20	557.8	**6.4**	1215.3	**12.0**	2571.2	**24.3**
	10	691.2	**8.8**	1543.1	**18.0**	3186.0	**36.1**
	5	909.6	**16.2**	1915.4	**33.5**	4086.0	**66.5**

B. Computational efficiency of the B-spline Optimization

To demonstrate the computational efficiency of our B-spline implementation, we conducted an ablation study on the convergence time of the B-spline optimization problem. The results, presented in Table X, are based on a sequence from the NeuroBEM dataset, where the drone follows a random trajectory, split into sequential, non-overlapping segments of 0.1, 0.2, 0.5, and 1.0 s. We evaluated different B-spline orders, control point spacings, and torque measurement rates, all within the practical operating range of HDVIO2.0. Control points were initialized via linear interpolation of gyroscope measurements. The experiments are run on a laptop running Ubuntu 20.04 and equipped with an Intel Core i9 2.3 GHz CPU. The results confirm that our B-spline optimization is highly efficient.

TABLE X: Convergence time in [ms] of the optimization problem that optimizes the B-spline control points according to the quadrotor's rotational dynamics model (see Eq. 2). We selected a sequence from the NeuroBEM dataset where the drone follows a random trajectory. This trajectory is split into sequential, non-overlapping segments. We evaluated different B-spline orders, control point spacings, and torque measurement rates, all within the practical operating range of HDVIO2.0. We report the average convergence time computed over all the trajectory segments. – indicates an infeasible configuration caused by a trajectory segment that is too short relative to the chosen B-spline order and control point spacing.

B-spline Order	**Ctrl. Points Spacing** [ms]	**Input Measurements Frequency** [Hz]							
		100				200			
		Trajectory Segment Length [s]							
		0.1	0.2	0.5	1.0	0.1	0.2	0.5	1.0
5	50	–	–	0.6	1.7	–	–	0.9	2.5
	20	–	0.5	1.5	3.2	–	0.3	1.0	2.0
	10	1.0	1.9	5.0	10.1	0.4	1.0	2.1	4.3
6	50	–	–	0.8	2.7	–	–	1.2	3.2
	20	–	0.6	2.3	5.0	–	0.7	2.7	5.3
	10	0.8	1.9	6.8	11.0	0.6	1.7	4.3	9.3
7	50	–	–	1.0	3.1	–	–	1.3	4.2
	20	–	0.7	3.3	7.0	–	0.8	3.2	6.7
	10	0.6	1.6	4.7	9.5	0.8	2.1	5.7	12.0

REFERENCES

[1] Barza Nisar, Philipp Foehn, Davide Falanga, and Davide Scaramuzza. VIMO: Simultaneous visual inertial model-based odometry and force estimation. *Robotics: Science and Systems (RSS)*, 2019.

[2] Ziming Ding, Tiankai Yang, Kunyi Zhang, Chao Xu, and Fei Gao. Vid-fusion: Robust visual-inertial-dynamics odometry for accurate external force estimation. In *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2021.

[3] Chuchu Chen, Yulin Yang, Patrick Geneva, Woosik Lee, and Guoquan Huang. Visual-inertial-aided online mav system identification. In *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2022.

[4] Mederic Fourmy, Thomas Flayols, Pierre-Alexandre Léziart, Nicolas Mansard, and Joan Solà. Contact forces preintegration for estimation in legged robotics using factor graphs. In *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2021.

[5] Jeonguk Kang, Hyunbin Kim, and Kyung-Soo Kim. View: Visual-inertial external wrench estimator for legged robot. *IEEE Robot. Autom. Lett.*, 2023.

[6] Leonard Bauersfeld, Elia Kaufmann, Philipp Foehn, Sihao Sun, and Davide Scaramuzza. NeuroBEM: Hybrid aerodynamic quadrotor model. *Robotics: Science and Systems (RSS)*, 2021.

[7] Sihao Sun, Coen C de Visser, and Qiping Chu. Quadrotor gray-box model identification from high-speed flight data. *Journal of Aircraft*, 2019.

[8] Leonard Bauersfeld and Davide Scaramuzza. Range, endurance, and optimal speed estimates for multicopters. *IEEE Robot. Autom. Lett.*, 2022.

[9] Michael O'Connell, Guanya Shi, Xichen Shi, Kamyar Azizzadenesheli, Anima Anandkumar, Yisong Yue, and Soon-Jo Chung. Neural-fly enables rapid learning for

agile flight in strong winds. *Science Robotics*, 2022.

[10] Elia Kaufmann, Leonard Bauersfeld, Antonio Loquercio, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Champion-level drone racing using deep reinforcement learning. *Nature*, 2023.

[11] Tong Qin, Peiliang Li, and Shaojie Shen. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Trans. Robot.*, 2018.

[12] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza. On-manifold preintegration for real-time visual-inertial odometry. *IEEE Trans. Robot.*, 2016.

[13] 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. Accessed: 2023-19-05.

[14] Christian Forster, Zichao Zhang, Michael Gassner, Manuel Werlberger, and Davide Scaramuzza. SVO: Semidirect visual odometry for monocular and multicamera systems. *IEEE Trans. Robot.*, 2017.

[15] Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart, and Paul Furgale. Keyframe-based visual–inertial odometry using nonlinear optimization. *Int. J. Robot. Research*, 2015.

[16] Giovanni Cioffi, Leonard Bauersfeld, and Davide Scaramuzza. Hdvio: Improving localization and disturbance estimation with hybrid dynamics vio. *Robotics: Science and Systems (RSS)*, 2023.

[17] Giovanni Cioffi, Titus Cieslewski, and Davide Scaramuzza. Continuous-time vs. discrete-time vision-based slam: A comparative study. *IEEE Robot. Autom. Lett.*, 2022.

[18] Amado Antonini, Winter Guerra, Varun Murali, Thomas Sayre-McCord, and Sertac Karaman. The blackbird uav dataset. *Int. J. Robot. Research*, 2020.

[19] Kunyi Zhang, Tiankai Yang, Ziming Ding, Sheng Yang, Teng Ma, Mingyang Li, Chao Xu, and Fei Gao. The visual-inertial-dynamical multirotor dataset. In *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2022.

[20] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2016.

[21] Teodor Tomić and Sami Haddadin. A unified framework for external wrench estimation, interaction control and collision reflexes for flying robots. In *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2014.

[22] Burak Yüksel, Cristian Secchi, Heinrich H Bülthoff, and Antonio Franchi. A nonlinear force observer for quadrotors and application to physical interactive tasks. In *2014 IEEE/ASME Int. Conf. on Advanced Intel. Mechatronics*, 2014.

[23] Fabio Ruggiero, Jonathan Cacace, Hamid Sadeghian, and Vincenzo Lippiello. Impedance control of vtol uavs with a momentum-based external generalized forces estimator. In *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2014.

[24] Christopher D McKinnon and Angela P Schoellig. Unscented external force and torque estimation for quadrotors. In *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2016.

[25] Federico Augugliaro and Raffaello D'Andrea. Admittance control for physical human-quadrocopter interaction. In *IEEE Eur. Control Conf. (ECC)*, 2013.

[26] Andrea Tagliabue, Mina Kamel, Sebastian Verling, Roland Siegwart, and Juan Nieto. Collaborative transportation using mavs via passive force control. In *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2017.

[27] Dinuka Abeywardena, Zhan Wang, Gamini Dissanayake, Steven L Waslander, and Sarath Kodagoda. Model-aided state estimation for quadrotor micro air vehicles amidst wind disturbances. In *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2014.

[28] Lucas Wälti and Alcherio Martinoli. Lumped drag model identification and real-time external force detection for rotary-wing micro aerial vehicles. In *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2024.

[29] Yuhan Yin, Qingkai Yang, and Hao Fang. Error-state kalman filter based external wrench estimation for mavs under a cascaded architecture. In *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2023.

[30] Anastasios I Mourikis and Stergios I Roumeliotis. A multi-state constraint kalman filter for vision-aided inertial navigation. In *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2007.

[31] Andrea Tagliabue, Aleix Paris, Suhan Kim, Regan Kubicek, Sarah Bergbreiter, and Jonathan P How. Touch the wind: Simultaneous airflow, drag and interaction sensing on a multirotor. In *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2020.

[32] Amir Moeini, Alan F Lynch, and Qing Zhao. Visual-inertial-actuator odometry for multirotor uavs with rotor drag and external disturbance. *Int. Jo. of Dynamics and Control*, 2024.

[33] Fadri Furrer, Michael Burri, Markus Achtelik, and Roland Siegwart. Rotors—a modular gazebo mav simulator framework. In *Robot operating system (ROS)*, 2016.

[34] Yunlong Song, Selim Naji, Elia Kaufmann, Antonio Loquercio, and Davide Scaramuzza. Flightmare: A flexible quadrotor simulator. In *Conf. on Robot. Learning (CoRL)*, 2020.

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

[36] 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*, 2012.

[37] Rajan Gill and Raffaello D'Andrea. Propeller thrust and drag in forward flight. In *2017 IEEE Conf. on Control Tech. and Applications (CCTA)*, 2017.

[38] 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, 2007.

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

[40] Cedric Le Gentil and Teresa Vidal-Calleja. Continuous integration over so (3) for imu preintegration. *Robotics: Science and Systems (RSS)*, 2021.

[41] Cedric Le Gentil and Teresa Vidal-Calleja. Continuous latent state preintegration for inertial-aided systems. *Int. J. Robot. Research*, 2023.

[42] Xudong Li, Zhixiang Wang, Zihao Liu, Yizhai Zhang, Fan Zhang, Xiuming Yao, and Panfeng Huang. Asynchronous event-inertial odometry using a unified gaussian process regression framework. In *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2024.

[43] Christiane Sommer, Vladyslav Usenko, David Schubert, Nikolaus Demmel, and Daniel Cremers. Efficient derivative computation for cumulative b-splines on lie groups. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11148–11156, 2020.

[44] William H Press, William T Vetterling, Saul A Teukolsky, and Brian P Flannery. *Numerical recipes*. Cambridge University Press, London, England, 1988.

[45] Joan Sola. Quaternion kinematics for the error-state kalman filter. *arXiv preprint arXiv:1711.02508*, 2017. Accessed 2025-05-22.

[46] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018. Accessed 2023-03-02.

[47] Zichao Zhang and Davide Scaramuzza. A tutorial on quantitative trajectory evaluation for visual(-inertial) odometry. In *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2018.

[48] Philipp Foehn, Elia Kaufmann, Angel Romero, Robert Penicka, Sihao Sun, Leonard Bauersfeld, Thomas Laengle, Giovanni Cioffi, Yunlong Song, Antonio Loquercio, and Davide Scaramuzza. Agilicious: Open-source and open-hardware agile quadrotor for vision-based flight. *Science Robotics*, 2022.

[49] Guillaume Ducard and Minh-Duc Hua. Modeling of an unmanned hybrid aerial vehicle. In *2014 IEEE Conf. on Control Applications (CCA)*, 2014.

[50] Nils Paul van Hinsberg. Aerodynamics of smooth and rough square-section prisms at incidence in very high reynolds-number cross-flows. *Experiments in Fluids*, 2021.

[51] Robert E. Sheldahl and Paul C. Klimas. Aerodynamic characteristics of seven symmetrical airfoil sections through 180-degree angle of attack for use in aerodynamic analysis of vertical axis wind turbines. In *Sandia National Labs., Albuquerque, NM (USA)*, 1981.

[52] Christian Pfeiffer, Simon Wengeler, Antonio Loquercio, and Davide Scaramuzza. Visual attention prediction improves performance of autonomous drone racing agents. *Plos one*, 2022.

[53] Andy Zeng, Shuran Song, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. Tossingbot: Learning to throw arbitrary objects with residual physics. *IEEE Trans. Robot.*, 2020.

[54] Junpeng Gao, Mike Y Michelis, Andrew Spielberg, and Robert K Katzschmann. Sim-to-real of soft robots with learned residual physics. *IEEE Robot. Autom. Lett.*, 2024.

[55] Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 2019.

[56] Giovanni Cioffi, Leonard Bauersfeld, Elia Kaufmann, and Davide Scaramuzza. Learned inertial odometry for autonomous drone racing. *IEEE Robot. Autom. Lett.*, 2023.

[57] Jiaxu Xing, Leonard Bauersfeld, Yunlong Song, Chunwei Xing, and Davide Scaramuzza. Contrastive learning for enhancing robust scene transfer in vision-based agile flight. In *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2024.

Giovanni Cioffi holds an M.Sc. in Mechanical Engineering from ETH Zürich, Switzerland, which he obtained in 2019. He is currently pursuing a Ph.D. at the University of Zürich under the supervision of Prof. Davide Scaramuzza. His research centers on the intersection of computer vision and robotics, exploring topics such as visual(-inertial) odometry and SLAM. His contributions were recognized by multiple awards in top-tier robotic conferences and journals, such as the IROS 2023 Best Paper Award and the RA-L 2021 Best Paper Award.

Leonard Bauersfeld received his M.Sc. degree in robotics, system and control from ETH Zurich, Switzerland in 2020. He is currently a PhD student in the Robotics and Perception Group at the University of Zurich, led by Prof. Davide Scaramuzza. His research interests are autonomous vision-based quadrotor flight and quadrotor simulations. He works on novel approaches, combining first-principles methods with modern data-driven models to advance agile quadrotor flight.

Davide Scaramuzza is a Professor of Robotics and Perception at the University of Zurich. He did his Ph.D. at ETH Zurich, a postdoc at the University of Pennsylvania, and was a visiting professor at Stanford University. His research focuses on autonomous, agile microdrone navigation using standard and event-based cameras. He pioneered autonomous, vision-based navigation of drones, which inspired the navigation algorithm of the NASA Mars helicopter and many drone companies. He contributed significantly to visual-inertial state estimation, vision-based agile navigation of microdrones, and low-latency, robust perception with event cameras, which were transferred to many products, from drones to automobiles, cameras, AR/VR headsets, and mobile devices. In 2022, his team demonstrated that an AI-controlled, vision-based drone could outperform the world champions of drone racing, a result that was published in Nature. He is a consultant for the United Nations on disaster response, AI for good, and disarmament. He has won many awards, including an IEEE Technical Field Award, the IEEE Robotics and Automation Society Early Career Award, a European Research Council Consolidator Grant, a Google Research Award, two NASA TechBrief Awards, and many paper awards. In 2015, he co-founded Zurich-Eye, today Meta Zurich, which developed the world-leading virtual-reality headset Meta Quest. In 2020, he co-founded SUIND, which builds autonomous drones for precision agriculture. Many aspects of his research have been featured in the media, such as The New York Times, The Economist, and Forbes.