2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

AI-IO: An Aerodynamics-Inspired Real-Time Inertial Odometry for Quadrotors

Jiahao Cui*, Feng Yu*, Linzuo Zhang, Yu Hu, and Danping Zou†

***Abstract*— Inertial Odometry (IO) has gained attention in quadrotor applications due to its sole reliance on inertial measurement units (IMUs), attributed to its lightweight design, low cost, and robust performance across diverse environments. However, most existing learning-based inertial odometry systems for quadrotors either use only IMU data or include additional dynamics-related inputs such as thrust, but still lack a principled formulation of the underlying physical model to be learned. This lack of interpretability hampers the model's ability to generalize and often limits its accuracy. In this work, we approach the inertial odometry learning problem from a different perspective. Inspired by the aerodynamics model and IMU measurement model, we identify the key physical quantity—rotor speed measurements required for inertial odometry and design a transformer-based inertial odometry. By incorporating rotor speed measurements, the proposed model improves velocity prediction accuracy by 36.9%. Furthermore, the transformer architecture more effectively exploits temporal dependencies for denoising and aerodynamic modeling, yielding an additional 22.4% accuracy gain over previous results. To support evaluation, we also provide a real-world quadrotor flight dataset capturing IMU measurements and rotor speed for high-speed motion. Finally, combined with an uncertainty-aware extended Kalman filter (EKF), our framework is validated across multiple datasets and real-time systems, demonstrating superior accuracy, generalization, and real-time performance. We share the code and data to promote further research (`https://github.com/SJTU-ViSYS-team/AI-IO`).**

I. INTRODUCTION

Low-cost IMUs are widely preferred for quadrotor state estimation owing to their compact size, low power consumption, and robustness to environmental degradation. However, achieving accurate inertial odometry with low-cost IMUs remains a critical challenge due to their high noise and bias instability.

Traditional inertial odometry, such as dead reckoning, suffers from long-term drift caused by random noise and slowly varying biases in IMUs. Consequently, significant efforts have been made to improve the accuracy of inertial odometry for specific domain applications. These methods typically employ explicit motion priors or learned motion patterns to impose constraints on inertial odometry estimation, such as zero velocity updates (ZUPT) for ground vehicles and pedestrians [1], [2]. However, quadrotors are less constrained in their motion patterns, making it challenging to generalize such methods to quadrotor state estimation. Recent advances in inertial odometry for quadrotors have focused on learning-based methods. These works typically employ neural networks to predict position offsets or velocity estimates, yet they often fail to explicitly clarify the relationship between network inputs and outputs. For example, IMO [3] uses thrust instead of acceleration, and AirIO [4] relies solely on IMU data and attitude. Based on aerodynamic model analysis [5], [6], such input designs are inadequate, leading to incomplete observability in velocity estimation. The absence of a clear input-output mapping constrains prediction accuracy and generalization performance of these networks. Furthermore, most prior studies lack validation on real-world platforms.

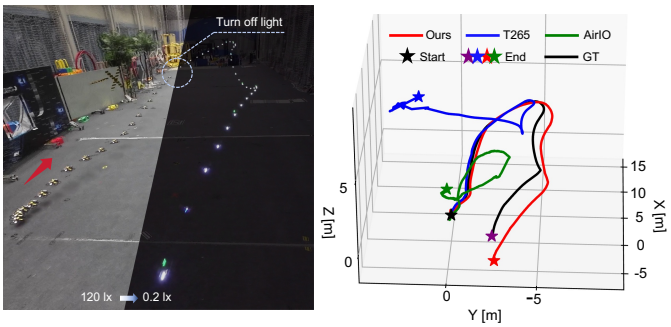

Fig. 1: Comparison of our realtime AI-IO with Intel T265's visual-inertial odometry (VIO) and AirIO. Our method achieves comparable accuracy to VIO under normal lighting and superior performance under dark conditions, demonstrating stronger robustness to visual degradation, while also achieving higher accuracy than state-of-the-art AirIO.

In this work, we propose a novel perspective to explain and guide the design of quadrotor inertial odometry, along with a transformer-based velocity estimator. Our key insight is that augmenting IMU data with rotor speed enables the network to implicitly exploit the underlying quadrotor aerodynamics, thereby improving the accuracy of inertial odometry. Building on this, we introduce a lightweight transformer architecture—the first to be applied in quadrotor inertial odometry—which achieves improved velocity prediction accuracy while maintaining real-time inference efficiency. Extensive evaluations on both self-collected datasets and real-world flight experiments demonstrate that the proposed framework consistently outperforms existing approaches in terms of accuracy and generalization capability. In summary, our contributions are as follows:

- We present an interpretable perspective that derives

*These authors contributed equally to this work.

†Corresponding author. All authors are with the Shanghai Key Laboratory of Navigation and Location-Based Services, Shanghai Jiao Tong University. This work was supported by National Key R&D Program of China (2022YFB3903802) and National Science Foundation of China (62073214).
Emails:{csfalpha, yu-feng, zhanglinzuo, henryhuyu, dpzou}@sjtu.edu.cn

from the quadrotor aerodynamic model, offering a physics-based explanation of estimation errors and validating rotor speed as a critical measurement to enhance inertial odometry accuracy.
- Building on this perspective, we design the **Aerodynamics-Inspired Inertial Odometry (AI-IO)**, a lightweight transformer-based velocity estimator coupled with an uncertainty-aware EKF, enabling real-time, high-precision pose estimation.
- We establish a highly maneuverable quadrotor dataset using our self-built platform, including IMU data, rotor speed, and ground-truth poses and velocities, thereby filling a critical gap in datasets for agile quadrotor motions.

II. RELATED WORK

A. Inertial Odometry

Traditional model-based IO methods rely on integrating inertial measurements for state estimation, which is often unreliable because measurement errors and time-varying biases can cause significant accumulated drift in pose estimation. To mitigate the drift, IMUs are combined with exteroceptive sensors such as cameras [7] and LiDARs [8]. Although these algorithms achieve high accuracy, their dependence on external sensors makes them vulnerable to disturbances caused by agile motion or dynamic environments.

With the development of deep learning, data-driven methods have shown great potential for addressing noise and drift in low-cost IMU data, as well as for learning velocity or displacement directly from IMU measurements. Considering sensor errors, OriNet [9] employs a LSTM network to calibrate gyroscope data, thereby improving the accuracy of inertial navigation. Similarly, Chen et al. [10] propose a deep learning-based approach to suppress multiple error sources in the sensor signals. Many other studies adopt a data-driven paradigm to estimate displacement or velocity directly from IMU data. For example, IONet [11], RoNIN [12] and RIDI [13] are end-to-end learning-based methods for pose estimation. Subsequent works integrate learned displacement or velocity into EKF frameworks [14], [15], [16] or batched optimization frameworks [17], aiming to build a more comprehensive state estimation pipeline. Recently, transformers have increasingly been adopted as the backbone of IO networks due to their powerful global attention mechanism. CTIN [18] and iMoT [19] have demonstrated the superiority of transformer-based IO architectures. Most of the aforementioned IO studies focus on pedestrian motion, limiting their applicability to platforms with more demanding motion dynamics, such as quadrotors.

B. Inertial Odometry for Quadrotors

Unlike pedestrian state estimation, quadrotors operate in six degrees of freedom in three-dimensional space, more complex motion patterns, and possess greater agility, making state estimation more challenging. The blade-element-momentum (BEM) theory [20], [21] combines blade-element theory and momentum theory to alleviate the difficulty of calculating induced velocity and to accurately capture the aerodynamic forces and torques acting on a single rotor under a wide range of operating conditions. Building on dynamic modeling of quadrotors, Sartori et al. [5] and Wang et al. [6] use the drag model to improve the quadrotor state estimation.

Although quadrotor aerodynamics and IMU observation model characterize the relationship between quadrotor body-frame velocity and IMU measurements, contemporary IO methods for quadrotors generally neglect aerodynamic drag models as a critical component. DIDO [22] introduces a state estimation framework that integrates quadrotor dynamics and network observations in a two-stage EKF to jointly estimate IMU biases together with kinematic and dynamic states. IMO [3] employs collective thrust and gyroscope histories as inputs to a temporal convolutional network (TCN) to regress displacements, implicitly capturing quadrotor dynamics, but generalizing poorly to unseen trajectories due to its use of world-frame representation and the absence of acceleration inputs. Wang et al. [23], DIVE [16], and AirIO [4] rely solely on IMU data for prediction, disregarding rotor aerodynamic drag in the quadrotor dynamics model [24]. As the current state-of-the-art, AirIO makes an important step forward by adopting body-frame representation. However, it still overlooks rotor speed as a key physical variable in velocity estimation, a limitation that our method explicitly addresses. In addition, all of these methods are evaluated on pre-collected datasets rather than deployed on resource-constrained quadrotor platforms for real-time tests.

III. METHODOLOGY

A. Aerodynamics for Velocity Estimation

1) Notation: The reference frame $\mathcal{W}$ is a fixed world coordinate system with its z-axis aligned with gravity. The quadrotor body frame and the IMU frame are denoted by $\mathcal{B}$ and $\mathcal{I}$, respectively, and all frames follow the right-hand convention. For any vector $\mathbf{x}$, the notation ${}^{W}\mathbf{x}$ denotes the representation of $\mathbf{x}$ in the $\mathcal{W}$ frame, with analogous notation applied to other frames. The position, velocity, and orientation of the body frame $\mathcal{B}$ with respect to the world frame $\mathcal{W}$ are denoted by ${}^{W}\mathbf{p}_B^W \in \mathbb{R}^3$, ${}^{W}\mathbf{v}_B^W \in \mathbb{R}^3$, and $\mathbf{R}_B^W \in \mathbb{R}^{3\times3}$, respectively. For brevity, these quantities are hereafter denoted as $\mathbf{p}, \mathbf{v}$, and $\mathbf{R}$ when the reference and target frames are clear from the context. Estimated and measured quantities are distinguished using the symbols $\hat{\mathbf{x}}$ and $\tilde{\mathbf{x}}$, respectively.

2) IMU Model: IMU measurements include the non-gravitational force in inertial space and the angular velocity of rigid frame given by:

$$\begin{aligned} {}^{I}\tilde{\boldsymbol{\omega}} &= {}^{I}\boldsymbol{\omega} + {}^{I}\mathbf{b}_g + \mathbf{n}_g \\ {}^{I}\tilde{\mathbf{a}} &= {}^{I}\mathbf{a} + {}^{I}\mathbf{b}_a - (\mathbf{R}_I^W)^\top {}^{W}\mathbf{g} + \mathbf{n}_a \end{aligned} \tag{1}$$

where ${}^{I}\tilde{\boldsymbol{\omega}}$ and ${}^{I}\tilde{\mathbf{a}}$ are the measurements of on-board gyroscope and accelerometer, ${}^{I}\boldsymbol{\omega}$ and ${}^{I}\mathbf{a}$ are the ground-truth angular velocity and acceleration, $\mathbf{n}_g$ and $\mathbf{n}_a$ are Guassian noise, and ${}^{W}\mathbf{g} = [0, 0, -9.81]\ (m/s^2)$ is the gravity vector, ${}^{I}\mathbf{b}_g$ and ${}^{I}\mathbf{b}_a$ are biases modeled as random walk processes with noise $\mathbf{n}_{\mathbf{b}_g}$ and $\mathbf{n}_{\mathbf{b}_a}$.

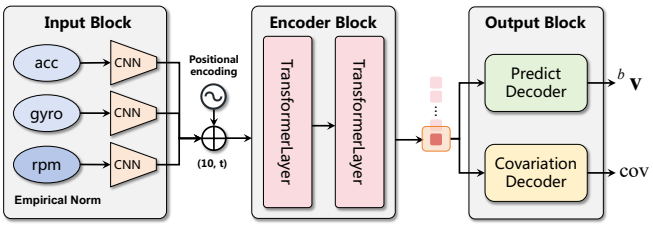

Fig. 2: Proposed transformer-based architecture: Verified measurements are normalized, encoded, and fed into a transformer to capture spatiotemporal dependencies. Finally, two fully connected decoders predict velocity and uncertainty.

3) Quadrotor Aerodynamics: A quadrotor has four propellers fixed in a rigid body symmetrically, with one pair of propellers rotating counter-clock-wise and the other pair clock-wise. To reveal the relationship between accelerometer measurements and quadrotor velocity, we model the observation equation (1) of IMU sensor combined with aerodynamics of quadrotor following [5], [6] as follows:

$$\begin{cases} \tilde{a}_x = \underbrace{-\lambda_x \omega_m^2 v_x}_{induced\ drag} \underbrace{-k_1 v_x - k_2 v_x^2}_{air\ drag} + \underbrace{b_{a_x} + n_{a_x}}_{noise}, \\ \tilde{a}_y = -\lambda_y \omega_m^2 v_y - k_3 v_y - k_4 v_y^2 + b_{a_y} + n_{a_y}, \\ \tilde{a}_z = \underbrace{4\alpha\omega_m^2}_{total\ thrust} - k_5 v_z - k_6 v_z^2 + b_{a_z} + n_{a_z} \end{cases} \tag{2}$$

where $w_m^2 = \frac{\sum_{i=1}^{4} \omega_i^2}{4}$, ω_i denotes the rotor speed of the i-th rotor. k_i, i=1,2,...,6, λ_j, j={x,y}, and α are approximately constant parameters depend on material, structure of propellers and airframe, and air density. Note that aerodynamic forces depend on the relative airflow velocity rather than the body-frame velocity. However, since we consider windless conditions in this work, the two are equivalent and the above formulation is valid. Further considering the effect of the non-inertial frame, the Coriolis acceleration term $\boldsymbol{a}_c = -2\boldsymbol{\omega} \times \boldsymbol{v}$ is added to (2) to compensate for quadrotor's rotation, which explains why angular velocity is required as one of the observations for network inputs.

4) Rotor Speed: According to (2), if the environment and quadrotor body properties remain constant over a short time horizon and noise is neglected, IMU measurements and body velocity are mutually derivable under our modeled aerodynamics when other observations (e.g., rotor speeds) are available. Therefore, assuming that the network can model the aerodynamics from temporal observations (i.e., coefficients such as $k_{i=1,2,...,6}, \lambda_{j=x,y}$ and α in (2)), beyond the acceleration and angular velocity measured by IMU, we argue that introducing rotor speeds as inputs to ensure observability of the estimator is critical to enhancing velocity estimation accuracy—a claim validated by subsequent experiments. This property forms the basis of our method.

B. Network Design & Training

Although the proposed equation (2) indicates a single-frame correspondence between IMU measurements and quadrotor speed, prior studies have employed historical multi-frame data buffer for estimation [16], [22], [3], [4]. We attribute this to the use of historical data buffer, which aids in better estimating IMU noise and better aerodynamics modeling. In recent years, transformer networks have demonstrated superior denoising capabilities across various modal data [25], [26], [27], so we integrate a transformer network with our input and output blocks to process historical data buffer, as shown in Fig. 2.

Directly feeding raw noisy data into the encoder may fail to capture the complex relationships between individual feature dimensions. Therefore, we employ CNNs to process raw sensor data sequences. We posit that applying multiple temporal convolutions acts as a low-pass filter, removing specific high-frequency noise [28]. Furthermore, the local inductive bias inherent in CNNs aligns with the local correlation characteristics of IMU data, enabling effective extraction of local features [29]. Note that the rotor speed values are often much larger than other measurements; thus, we normalize the mean and variance of rotor speeds online using empirical values to align with the standard normal distribution and enhance training stability.

After fusion with position encoding, the processed features is fed into a transformer for temporal feature extraction and aerodynamics modeling. Unlike the original transformer, we eliminate the attention mechanism in its decoder and directly employ two fully connected layers to process the encoder's temporal features of the last moment. These fully connected layers independently generate the predicted velocity and uncertainty of the last frame, enabling lightweight and real-time inference.

To supervise the velocity prediction and uncertainty estimation of the last frame, we adopt two loss functions [14], [30]:

$$\begin{aligned} \mathcal{L}_V &= \begin{cases} \frac{1}{2}({}^B\mathbf{v}_i - {}^B\hat{\mathbf{v}}_i)^2, & \text{if } |{}^B\mathbf{v}_i - {}^B\hat{\mathbf{v}}_i| < \delta, \\ \delta \cdot (|{}^B\mathbf{v}_i - {}^B\hat{\mathbf{v}}_i| - \frac{1}{2}\delta), & \text{otherwise} \end{cases} \\ \mathcal{L}_C &= \left({}^B\mathbf{v}_i - {}^B\hat{\mathbf{v}}_i\right)^\top \hat{\Sigma}_i^{-1} \left({}^B\mathbf{v}_i - {}^B\hat{\mathbf{v}}_i\right) + \ln\left(\det \hat{\Sigma}_i\right) \end{aligned} \tag{3}$$

where the $\mathcal{L}_V$ is the Huber loss between the ground-truth velocity ${}^B\mathbf{v}_i$ and the predicted velocity ${}^B\hat{\mathbf{v}}_i$ for the i-th data inputs, the $\mathcal{L}_C$ is the negative-log-likelihood (NLL) loss used for uncertainty supervision, $\hat{\Sigma}_i = diag(\hat{\sigma}_i)$ is the predicted uncertainty in the form of covariance matrix with only diagonal values $\hat{\sigma}_i$ output by the decoder. Once the velocity prediction loss has converged, training proceeds with the NLL loss term as suggested by [16].

C. Dataset Preparation

Among the currently available quadrotor datasets, only the Blackbird [31], VID [32], and DIDO [22] datasets include rotor speeds. However, the flight trajectories in the Blackbird are overly repetitive, limiting network generalization. The VID and DIDO primarily contain non-aggressive flight data with relatively low speeds. To meet the requirements of this study for high-aggressiveness flight data with rotor speed, we equip a quadrotor as a data acquisition platform (shown in Fig. 3) and collect a large amount of high-speed flight data.

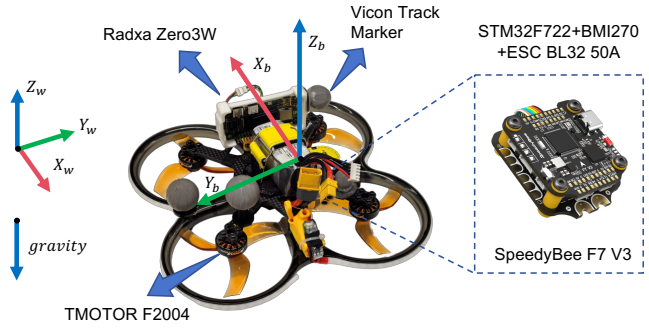

Fig. 3: Hardware setup which assembles a low-cost BMI270 IMU for acceleration and angular velocity and ESCs using bidirectional Dshot for rotor speed.

The quadrotor is built on a SpeedyBee[1] Bee35 chassis with an F7 V3 Stack flight controller. IMU data is obtained from the BMI270[2] sensor, rotor speed is recorded via telemetry feedback from the electronic speed controller, and the ground truth is provided by a VICON[3] system.

To capture flight data with a diverse range of speeds, we manually control the quadrotor to fly at three levels: low, medium, and high, with maximum speeds of approximately 5m/s, 8m/s, and 14m/s, respectively. Four sequences are recorded for each speed, resulting in a total of 12 sequences, covering a flight distance of 6,704m and a duration of 1,960s. In addition, we collect low-speed data (below 3 m/s) from autonomous flights. These include planar circular and figure-eight patterns, non-planar circular and figure-eight patterns, and random flight trajectories. Two sequences are recorded for each type, yielding a total of 10 sequences, covering a flight distance of 563m and a duration of 676s.

D. Extended Kalman Filter

1) System State: Our EKF builds upon the framework established in [4]. The state at time i is defined as $\mathbf{X}_i = \{\hat{\mathbf{R}}_i, \hat{\mathbf{v}}_i, \hat{\mathbf{p}}_i, \hat{\mathbf{b}}_{a_i}, \hat{\mathbf{b}}_{g_i}\}$. Following the error-state formulation in [33], the corresponding error-state vector is defined as $\delta\mathbf{X}_i = \{\delta\boldsymbol{\theta}_i, \delta\mathbf{v}_i, \delta\mathbf{p}_i, \delta\mathbf{b}_{a_i}, \delta\mathbf{b}_{g_i}\}$. For Euclidean vector variables $\mathbf{x} \in \mathbb{R}^3$, the error vector is given by $\delta\mathbf{x} = \mathbf{x} - \hat{\mathbf{x}}$. For rotational components, the minimal error $\delta\boldsymbol{\theta} \in \mathbb{R}^3$ is obtained via the logarithmic map $\delta\boldsymbol{\theta} = \mathrm{Log}(\mathbf{R}\cdot\hat{\mathbf{R}}^{-1})$ whose inverse is given by the exponential map $\mathrm{Exp}(\delta\boldsymbol{\theta})$.

2) State Propagation: We propagate the state using high-frequency IMU measurements via:

$$\begin{cases} \hat{\mathbf{R}}_{i+1} = \hat{\mathbf{R}}_i \cdot \mathrm{Exp}(\tilde{\boldsymbol{\omega}}_i - \hat{\mathbf{b}}_{g_i}) \cdot \Delta t \\ \hat{\mathbf{v}}_{i+1} = \hat{\mathbf{v}}_i + {}^W\mathbf{g} \cdot \Delta t + \hat{\mathbf{R}}_i \cdot (\tilde{\mathbf{a}}_i - \hat{\mathbf{b}}_{a_i}) \cdot \Delta t \\ \hat{\mathbf{p}}_{i+1} = \hat{\mathbf{p}}_i + \hat{\mathbf{v}}_i \cdot \Delta t + \frac{1}{2} \cdot [{}^W\mathbf{g} + \hat{\mathbf{R}}_i(\tilde{\mathbf{a}}_i - \hat{\mathbf{b}}_{a_i})] \cdot \Delta t^2 \\ \hat{\mathbf{b}}_{a_{i+1}} = \hat{\mathbf{b}}_{a_i}, \hat{\mathbf{b}}_{g_{i+1}} = \hat{\mathbf{b}}_{g_i} \end{cases} \tag{4}$$

where Δt denotes the integration interval. The linearized error-state and covariance propagation are modeled as:

$$\delta\mathbf{X}_{i+1} = \mathbf{A}_i \cdot \delta\mathbf{X}_i + \mathbf{B}_i \cdot \mathbf{n}_i \tag{5}$$

$$\mathbf{P}_{i+1} = \mathbf{A}_i \cdot \mathbf{P}_i \cdot \mathbf{A}_i^\top + \mathbf{B}_i \cdot \mathbf{W}_i \cdot \mathbf{B}_i^\top, \tag{6}$$

where $\mathbf{n}_i = [\mathbf{n}_{a_i}, \mathbf{n}_{g_i}, \mathbf{n}_{\mathbf{b}_{a_i}}, \mathbf{n}_{\mathbf{b}_{g_i}}]^\top$ represents sensor noise and bias noise, $\mathbf{W}_i$ is the propagation noise matrix.

3) Measurement Update: The observation model based on velocity in $\mathcal{B}$ is:

$${}^B\mathbf{v}_B^W = h(\mathbf{X}) = (\mathbf{R}_b^W)^\top \cdot {}^W\mathbf{v}_B^W \tag{7}$$

where $\mathbf{n}_\mathbf{v}$ denotes zero-mean Gaussian noise on the predictions of the network, with its covariance also provided by the network. Linearizing (7), the Jacobian matrix $\mathbf{H}$ is straightforward to compute, with all entries zero except

$$\begin{aligned} \mathbf{H}_{\boldsymbol{\theta}} &= \frac{\partial h(\mathbf{X})}{\partial \delta\boldsymbol{\theta}} = \hat{\mathbf{R}}^\top \lfloor \hat{\mathbf{v}} \rfloor_\times, \\ \mathbf{H}_\mathbf{v} &= \frac{\partial h(\mathbf{X})}{\partial \delta\mathbf{v}} = \hat{\mathbf{R}}^\top \end{aligned} \tag{8}$$

where $\lfloor \boldsymbol{\theta} \rfloor_\times$ denotes the skew-symmetric matrix of vector $\boldsymbol{\theta}$.

Finally, $\mathbf{H}$ are used to compute the Kalman gain to update the state $\mathbf{X}$ and the covariance $\mathbf{P}$ as follows:

$$\begin{cases} \mathbf{K} = \mathbf{P} \cdot \mathbf{H}^\top \cdot (\mathbf{H} \cdot \mathbf{P} \cdot \mathbf{H}^\top + \boldsymbol{\Sigma})^{-1} \\ \mathbf{X} \leftarrow \mathbf{X} + \mathbf{K} \cdot ({}^B\tilde{\mathbf{v}} - \hat{\mathbf{R}}^\top \cdot \hat{\mathbf{v}}) \\ \mathbf{P} \leftarrow (\mathbf{I} - \mathbf{K} \cdot \mathbf{H}) \cdot \mathbf{P} \cdot (\mathbf{I} - \mathbf{K} \cdot \mathbf{H})^{-1} + \mathbf{K} \cdot \boldsymbol{\Sigma} \cdot \mathbf{K}^\top \end{cases} \tag{9}$$

where $\boldsymbol{\Sigma}$ is the measurement covariance provided by the network.

During filter initialization, the insufficient data length prevents network inference, restricting the system to inertial propagation. Once sufficient data accumulates, the network performs inference according to the filter update frequency to provide body-frame velocity estimates.

IV. EXPERIMENTAL RESULTS

We evaluate our method on two quadrotor datasets: the relatively low-speed DIDO [22] dataset and our own customized aggressive high-speed dataset. Both datasets provide rotor speed measurements, enabling us to verify the effectiveness of the proposed algorithm under different maneuvering conditions. We compare our system with the following baselines:

- IMU preintegration [34]. A model-based method that pre-integrates the IMU measurements on the SO(3) manifold and simultaneously efficiently corrects the IMU bias estimate online.
- IMO [3]. A learning-based inertial odometry approach that embeds control signals specifically designed for drone racing, with high positioning accuracy on a fixed track trajectory.
- AirIO [4]. The current state-of-the-art learning-based inertial odometry algorithm for quadrotors, which validates the effectiveness of predicting body velocity directly from IMU data.

[1]https://www.speedybee.com/
[2]https://www.mouser.sg/new/bosch/bosch-sensortec-bmi270/
[3]https://www.vicon.com/

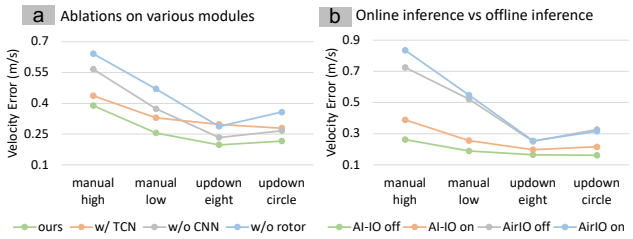

Fig. 4: (a) Ablation results for each module on the test dataset. (b) Comparison of online and offline inference results between AI-IO and AirIO with a window length set to 1 second.

Among these methods, IMU preintegration represents the model-based approach, while IMO and AirIO are representative learning-based methods for quadrotors.

A. Metrics Definitions

To assess the performance of our system, we adopt two widely used metrics to quantify the localization performance [12], [14], [4]: Absolute Translation Error (**ATE**, m) computes the Root Mean Squared Error (RMSE) between the estimated and ground-truth velocities over all time points, and Relative Translation Error (**RTE**, m) computes the RMSE of the relative velocities over predefined time intervals. In our experiments, we adopt a 5-second interval. Similar to position, we use Absolute Velocity Error (**AVE**, m/s) and Relative Velocity Error (**RVE**, m/s) to evaluate the accuracy of the velocity estimation.

B. Ablation Studies

1) Ablations on each module: We conduct a series of ablation experiments to analyze the effects of different components on the network, with a focus on assessing the influence of key observed values from rotor speed measurements. All experiments are performed using our self-collected dataset, with results presented in the Fig. 4 (a):

- **Model Backbone**: Compared with TCN, using transformer as the network backbone reduces the average body-frame velocity prediction error on the test set by 22.4%. We attribute this reduction to transformer's enhanced ability to model temporal dependencies, such as estimating biases and capturing aerodynamic parameters from historical observations.
- **Feature Extractor**: Placing CNN before the transformer backbone enables denoising and feature extraction, leading to a 23.9% reduction in the network's average velocity prediction error. This effect is particularly obvious on high-speed trajectories with higher noise.
- **Rotor Speed**: As with the aerodynamic modeling discussed in (2), introducing rotor speed measurements ensures the system's observability. Therefore, we compare velocity predictions without rotor speeds and observe a 36.9% reduction in velocity prediction error. This experimental result provides a better explanation for neural network learning.

2) Online inference vs offline inference: Although offline inference is not practically applicable in real-world scenarios, due to AirIO's excellent offline inference performance [4], we conduct a comprehensive comparison and analysis of velocity prediction errors between offline and online inference.

The key distinction between online and offline inference lies in how the sliding window is updated: online inference shifts the window by one time step, while offline inference shifts the entire window. Additionally, the supervised signals for both methods are derived from either the velocity at the window's end (online) or the velocities at multiple time steps within the window (offline). To validate this, we set the window length to 1 second. Fig. 4 (b) shows that offline inference achieves lower average velocity prediction error on its trajectory. This improvement stems from the denser supervision signals within the full window and its use of global window updates, with the effect being particularly pronounced on AI-IO. In order to fully compare the accuracy of real-time processing, the subsequent comparisons on datasets all use a 1-second window for online inference.

C. Comparison on Datasets

1) DIDO Dataset: We evaluate our method AI-IO and the state-of-the-art method AirIO on the DIDO dataset. We use all random flight data starting with "v" for training, and perform validation and testing on other periodic flight data and random flight data.

Table I shows the evaluation results on 5 types of sequences. The results of each type of sequence are the average of multiple sequences. AirIO Net and AI-IO Net use the real attitude information to convert the network output from body frame to world frame, and then directly integrate to obtain the position, while AirIO EKF and AI-IO EKF use attitude estimates instead of the true attitude. Because observation updates rely on body-frame velocity, errors in attitude estimates, particularly those introduced by low-cost IMUs such as the BMI270 we employed, can propagate to velocity updates and subsequently degrade position accuracy. Therefore, AI-IO EKF decreases 12.1% and 47.0% in AVE and ATE respectively compared to AI-IO Net. Since AirIO uses attitude as inputs, it introduces larger errors when the attitude estimation is inaccurate, resulting in a 109.6% and 52.9% decrease in AVE and ATE, respectively. Overall, AI-IO Net improves AVE and ATE by 41.3% and 32.5% respectively compared to AirIO Net, and AI-IO EKF improves AVE and ATE by 68.6% and 35.1% respectively compared to AirIO EKF.

2) Our Dataset: In order to fully verify the generalization of AI-IO across different speed distributions and different flight trajectories, we use our own collected data for training and evaluation. All manual flight data sequences are divided into training, validation, and testing data sets according to the ratio of 7:1.5:1.5, and all non-manual flight data are divided into the testing set. To avoid the influence of the filter parameter settings, the results of the unfiltered inference of AirIO and AI-IO are compared here. IMO's network output is a displacement within a window, which makes it difficult

TABLE I: Comparison on DIDO datasets. The best and second best results are bold and underlined respectively.

Sequence	AirIO Net[1]				AI-IO Net[1]				AirIO EKF				AI-IO EKF			
	AVE	RVE	ATE	RTE	AVE	RVE	ATE	RTE	AVE	RVE	ATE	RTE	AVE	RVE	ATE	RTE
circle	0.588	0.647	4.827	1.146	**0.327**	**0.364**	**2.588**	**0.724**	1.129	1.229	6.600	2.150	0.385	0.432	3.333	0.915
updown circle	0.563	0.742	4.999	1.153	**0.321**	**0.403**	**3.608**	**0.836**	1.960	1.126	6.085	1.806	0.355	0.427	5.319	1.158
eight	0.506	0.733	2.610	1.252	**0.321**	**0.392**	**1.448**	**0.684**	0.870	1.239	5.417	2.442	0.396	0.532	2.833	1.105
updown eight	0.639	0.940	3.535	1.517	**0.407**	**0.474**	**1.750**	**0.753**	1.057	1.443	6.194	2.766	0.431	0.593	2.419	1.167
random	0.522	0.772	**3.553**	1.294	**0.278**	0.363	3.788	**0.721**	0.895	1.366	5.564	2.408	0.287	**0.342**	5.474	0.975
Average	0.564	0.767	3.905	1.272	**0.331**	**0.399**	**2.636**	**0.744**	1.182	1.281	5.972	2.314	0.371	0.465	3.876	0.903

1 leverage ground truth orientation to transform the net output to world frame for inference.

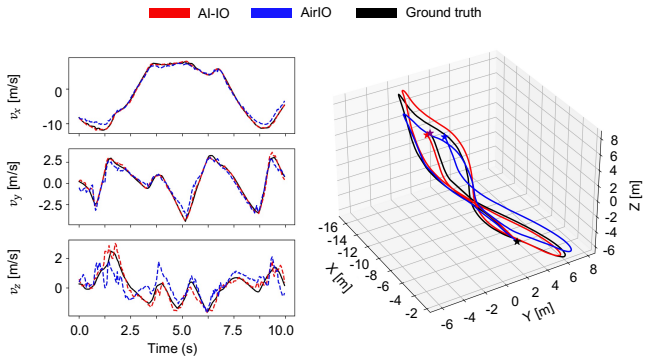

Fig. 5: Comparison of estimated velocity and trajectory in a manual high sequence, with the maximum speed exceeding 10m/s. The ATEs of AI-IO and AirIO are 0.860 and 1.762 respectively, and AI-IO outperforms AirIO by 51.2% in ATE.

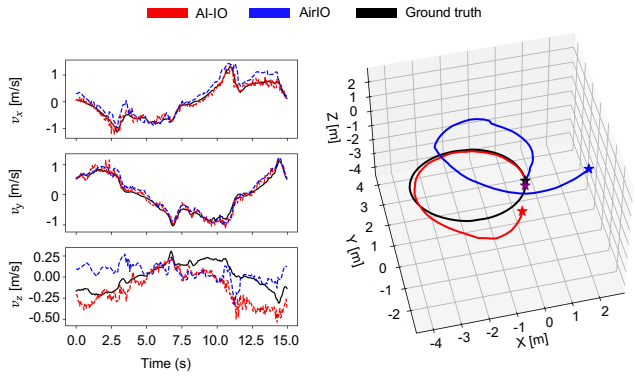

Fig. 6: Comparison of estimated velocity and trajectory in a updown circle sequence, the maximum speed doesn't exceed 2m/s. The ATEs of AI-IO and AirIO are 1.005 and 1.808 respectively, and AI-IO outperforms AirIO by 44.4% in ATE.

to directly infer velocity and position, so its original filter fusion is still used.

As shown in TABLE II, IMU preintegration has huge cumulative errors in both velocity and position, and IMO is completely ineffective for flight data on non-fixed tracks. AirIO and AI-IO have good generalization properties, achieving high accuracy across diverse datasets using only a subset of random flight data. However, our AI-IO method significantly outperforms other methods, achieving 57.4% and 29.3% improvements on AVE and ATE, respectively, compared to the state-of-the-art method, AirIO. Fig. 5 and 6 show the velocity comparison and flight trajectory of a manual high and updown circle sequence, respectively. Whether comparing estimated velocity or position, AI-IO outperforms AirIO. The IMU preintegration and IMO divergence is too severe and is no longer shown.

D. Real World Validation

To demonstrate how AI-IO supports real-world applications, we deploy it on the quadrotor platform shown in Fig. 3, which is equipped with a Radxa Zero3W computer (Quad-Core Arm Cortex-A55, up to 1.6GHz)[4]. We design the real-time system demonstrated in Fig. 7 and then conduct indoor flight tests:

[4]https://radxa.com/products/zeros/zero3w

1) How does AI-IO generalize to real-time systems?: The training data—randomly sampled by human pilots—covers trajectories with diverse characteristics, enabling AI-IO to naturally generalize to real-world flights. As shown in Fig. 8, the deployed AI-IO achieves real-time pose estimation on a human-piloted quadrotor. The body-frame velocity prediction network runs with an average inference time of 8.9 ms while maintaining high accuracy. Pose estimation is performed via an EKF at 20 Hz, and the results remain highly consistent over long-range and agile trajectories.

2) How physical inductive bias facilitate training?: To investigate the inductive bias of physical information learned during network training, we compare the velocity prediction errors of training from scratch versus fine-tuning. First, we train AI-IO on the dataset mentioned before. We then collect a small volume of 7-minute flight data using a quadrotor with modified mass and appearance, which is used to fine-tune the pre-trained model. As shown in Fig. 9, compared to training from scratch with limited data, fine-tuning demonstrates higher data utilization efficiency and converges to lower errors more rapidly by leveraging the physical information embedded in the pre-trained base model. This indicates that for any quadrotor, only a small amount of data is required to quickly train a usable inertial odometry.

3) How does AI-IO perform compared to other methods?: We mount the T265 camera on the original quadrotor plat-

TABLE II: Comparison on our datasets. The best results are bold.

Sequence	IMU preintegration[1]				IMO[1]				AirIO Net[2]				AI-IO Net[2]			
	AVE	RVE	ATE	RTE	AVE	RVE	ATE	RTE	AVE	RVE	ATE	RTE	AVE	RVE	ATE	RTE
manual high	54.68	12.01	104.1	24.61	6.033	8.572	22.72	12.91	1.093	1.511	2.514	1.891	**0.387**	**0.537**	**0.849**	**0.699**
manual medium	54.79	15.39	104.3	31.47	6.178	7.824	36.99	17.65	0.836	1.087	1.773	1.235	**0.285**	**0.363**	**0.994**	**0.569**
manual low	38.40	25.35	72.95	50.17	3.882	5.439	12.37	8.397	0.715	0.909	2.072	1.161	**0.268**	**0.336**	**0.783**	**0.434**
eight	56.09	7.991	108.2	17.69	3.987	3.473	80.61	14.32	0.274	0.379	2.720	0.550	**0.181**	**0.254**	**2.167**	**0.494**
updown eight	55.41	7.546	107.0	16.26	2.739	2.643	65.06	9.517	0.356	0.438	1.678	0.776	**0.182**	**0.236**	**1.256**	**0.407**
circle	56.11	6.538	108.3	14.34	3.770	4.343	68.32	15.57	0.307	0.417	2.134	0.801	**0.180**	**0.220**	**1.694**	**0.524**
updown circle	56.10	6.568	108.7	14.46	3.648	2.723	110.4	16.10	0.418	0.525	4.326	1.147	**0.204**	**0.236**	**3.380**	**0.643**
random	53.37	15.78	103.2	31.45	3.937	4.022	94.44	14.62	0.448	0.588	**3.255**	0.941	**0.206**	**0.235**	3.347	**0.594**
Average	53.12	12.15	102.1	25.06	4.272	4.880	61.37	13.63	0.556	0.732	2.559	1.063	**0.237**	**0.287**	**1.809**	**0.546**

1 leverage ground truth orientation to transform the IMU data to world frame for inference.
2 leverage ground truth orientation to transform the net output to world frame for inference.

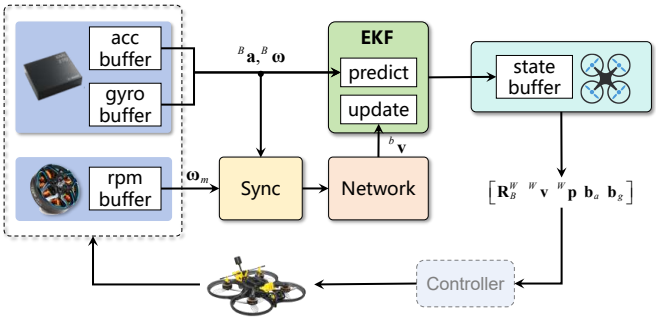

Fig. 7: Real-time system overview: IMU-derived acceleration, angular velocity, and rotor speed (via bidirectional DShot) are time-synchronized and fed into the network for velocity prediction. Subsequently, EKF performs prediction and update steps, outputting pose information to the controller for closed-loop control.

form and evaluate AI-IO (fine-tuned as described previously) by comparing its localization accuracy with state-of-the-art AirIO (fine-tuned in the same way) and the T265's VIO in indoor flight scenarios. To mitigate potential degradation in the T265's VIO, we place textured objects on one side of the environment and then turn off the lights during flight. As shown in Fig. 1, AI-IO consistently outperforms AirIO. Before the lights are turned off, AI-IO achieves localization accuracy comparable to the T265's VIO. After light extinction, however, the T265's VIO degrades and diverges from the ground truth, whereas AI-IO maintains high consistency.

V. CONCLUSION & DISCUSSION

In this work, we propose a novel perspective grounded in quadrotor aerodynamics and IMU models to explain and guide the design of learning-based inertial odometry. Guided by this framework, we develop the AI-IO, a lightweight transformer-based inertial odometry that, when combined with an EKF, achieves real-time inference on real-world and delivers high pose estimation accuracy. Additionally, we collect and release a high-maneuverability IMU dataset for quadrotors. Experimental results across diverse datasets and real-world platforms demonstrate that our framework achieves a 49.4% improvement in velocity estimation and a 30.9% improvement in position estimation compared to state-of-the-art methods. Future work may focus on developing network architectures more tightly integrated with physical priors (e.g., PINNs) to enhance interpretability and enable integration with multi-sensor frameworks such as VIO for further improvement in position estimation.

REFERENCES

[1] Z. Wang, H. Zhao, S. Qiu, and Q. Gao, "Stance-phase detection for zupt-aided foot-mounted pedestrian navigation system," *IEEE/ASME Transactions On Mechatronics*, vol. 20, no. 6, pp. 3170–3181, 2015.

[2] H. Li, H. Liu, Z. Li, C. Li, Z. Meng, N. Gao, and Z. Zhang, "Adaptive threshold-based zupt for single imu-enabled wearable pedestrian localization," *IEEE Internet of Things Journal*, vol. 10, no. 13, pp. 11749–11760, 2023.

[3] G. Cioffi, L. Bauersfeld, E. Kaufmann, and D. Scaramuzza, "Learned inertial odometry for autonomous drone racing," *IEEE Robotics and Automation Letters*, vol. 8, no. 5, pp. 2684–2691, 2023.

[4] Y. Qiu, C. Xu, Y. Chen, S. Zhao, J. Geng, and S. Scherer, "Airio: Learning inertial odometry with enhanced imu feature observability," *arXiv preprint arXiv:2501.15659*, 2025.

[5] D. Sartori, D. Zou, L. Pei, and W. Yu, "A revisited approach to lateral acceleration modeling for quadrotor uavs state estimation," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5711–5718, IEEE, 2018.

[6] R. Wang, D. Zou, C. Xu, L. Pei, P. Liu, and W. Yu, "An aerodynamic model-aided state estimator for multi-rotor uavs," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2164–2170, IEEE, 2017.

[7] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE transactions on robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[8] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus, "Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping," in *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 5135–5142, IEEE, 2020.

[9] M. A. Esfahani, H. Wang, K. Wu, and S. Yuan, "Orinet: Robust 3-d orientation estimation with a single particular imu," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 399–406, 2019.

[10] H. Chen, P. Aggarwal, T. Taha, and V. Chodavarapu, "Improving inertial sensor by reducing errors using deep learning methodology. naecon 2018-ieee national aerospace and electronics conference," 2018.

[11] C. Chen, X. Lu, A. Markham, and N. Trigoni, "Ionet: Learning to cure the curse of drift in inertial odometry," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, 2018.

[12] S. Herath, H. Yan, and Y. Furukawa, "Ronin: Robust neural inertial navigation in the wild: Benchmark, evaluations, & new methods," in *2020 IEEE international conference on robotics and automation (ICRA)*, pp. 3146–3152, IEEE, 2020.

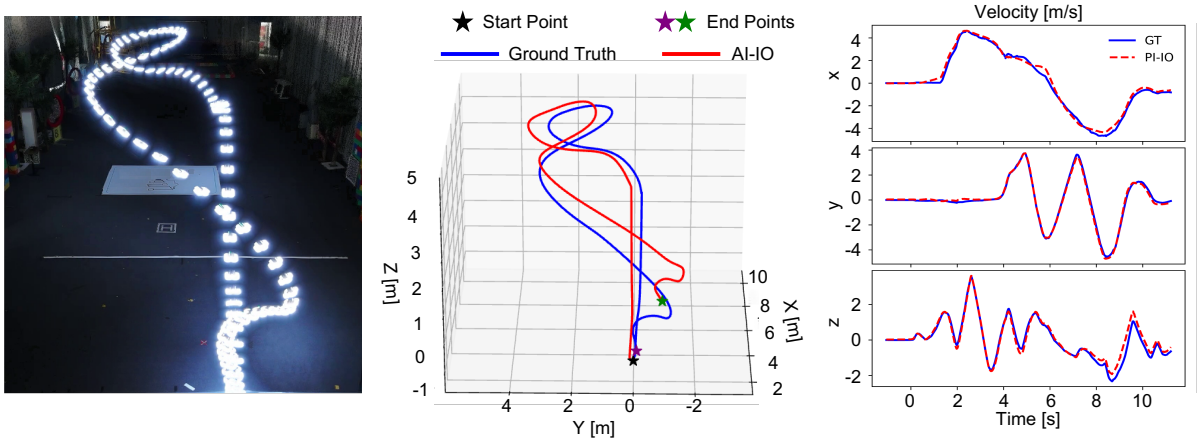

Fig. 8: We deploy the trained lightweight network on a low-compute platform for real-time body-frame velocity estimation, then integrate it with an extended Kalman filter to estimate the pose. The quadrotor is maneuvered randomly by human pilots in low-light indoor environments.

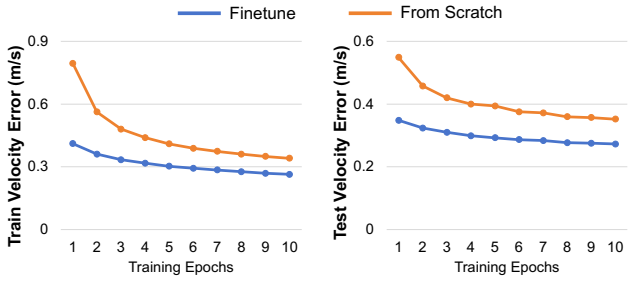

Fig. 9: Comparison of velocity prediction errors between training from scratch and fine-tuning on the training and testing datasets.

[13] H. Yan, Q. Shan, and Y. Furukawa, "Ridi: Robust imu double integration," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 621–636, 2018.

[14] W. Liu, D. Caruso, E. Ilg, J. Dong, A. I. Mourikis, K. Daniilidis, V. Kumar, and J. Engel, "Tlio: Tight learned inertial odometry," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5653–5660, 2020.

[15] S. Sun, D. Melamed, and K. Kitani, "Idol: Inertial deep orientation-estimation and localization," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, pp. 6128–6137, 2021.

[16] A. Bajwa, C. C. Cossette, M. A. Shalaby, and J. R. Forbes, "Dive: Deep inertial-only velocity aided estimation for quadrotors," *IEEE Robotics and Automation Letters*, vol. 9, no. 4, pp. 3728–3734, 2024.

[17] R. Buchanan, M. Camurri, F. Dellaert, and M. Fallon, "Learning inertial odometry for dynamic legged robot state estimation," in *Conference on robot learning*, pp. 1575–1584, PMLR, 2022.

[18] B. Rao, E. Kazemi, Y. Ding, D. M. Shila, F. M. Tucker, and L. Wang, "Ctin: Robust contextual transformer network for inertial navigation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, pp. 5413–5421, 2022.

[19] S. M. Nguyen, D. V. Le, and P. Havinga, "imot: Inertial motion transformer for inertial navigation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, pp. 6209–6217, 2025.

[20] M. Bangura, M. Melega, R. Naldi, and R. Mahony, "Aerodynamics of rotor blades for quadrotors," *arXiv preprint arXiv:1601.00733*, 2016.

[21] R. Gill and R. D'Andrea, "Computationally efficient force and moment models for propellers in uav forward flight applications," *Drones*, vol. 3, no. 4, p. 77, 2019.

[22] K. Zhang, C. Jiang, J. Li, S. Yang, T. Ma, C. Xu, and F. Gao, "Dido: Deep inertial quadrotor dynamical odometry," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 9083–9090, 2022.

[23] R. Wang, D. Zou, L. Pei, P. Liu, and C. Xu, "Velocity prediction for multi-rotor uavs based on machine learning," in *China Satellite Navigation Conference (CSNC) 2016 Proceedings: Volume II*, pp. 487–500, Springer, 2016.

[24] R. Mahony, V. Kumar, and P. Corke, "Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor," *IEEE robotics & automation magazine*, vol. 19, no. 3, pp. 20–32, 2012.

[25] J. Yang, K. Z. Luo, J. Li, C. Deng, L. Guibas, D. Krishnan, K. Q. Weinberger, Y. Tian, and Y. Wang, "Denoising vision transformers," in *European Conference on Computer Vision*, pp. 453–469, Springer, 2024.

[26] Z. Wang, X. Cun, J. Bao, W. Zhou, J. Liu, and H. Li, "Uformer: A general u-shaped transformer for image restoration," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 17683–17693, 2022.

[27] J. Chen, Q. Mao, and D. Liu, "Dual-path transformer network: Direct context-aware modeling for end-to-end monaural speech separation," *arXiv preprint arXiv:2007.13975*, 2020.

[28] M. Brossard, S. Bonnabel, and A. Barrau, "Denoising imu gyroscopes with deep learning for open-loop attitude estimation," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4796–4803, 2020.

[29] Z. Wang and L. Wu, "Theoretical analysis of the inductive biases in deep convolutional networks," *Advances in Neural Information Processing Systems*, vol. 36, pp. 74289–74338, 2023.

[30] R. L. Russell and C. Reale, "Multivariate uncertainty in deep learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 12, pp. 7937–7943, 2021.

[31] A. Antonini, W. Guerra, V. Murali, T. Sayre-McCord, and S. Karaman, "The blackbird dataset: A large-scale dataset for UAV perception in aggressive flight," *CoRR*, vol. abs/1810.01987, 2018.

[32] K. Zhang, T. Yang, Z. Ding, C. Xu, and F. Gao, "The visual-inertial-dynamical UAV dataset," *CoRR*, vol. abs/2103.11152, 2021.

[33] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint kalman filter for vision-aided inertial navigation," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pp. 3565–3572, 2007.

[34] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "Imu preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation," in *Robotics: Science and Systems (RSS)*, (Rome, Italy), pp. 1–20, Unknown, July 2015.