Learned Inertial Odometry for Autonomous Drone Racing

Giovanni Cioffi, Leonard Bauersfeld, Elia Kaufmann, and Davide Scaramuzza

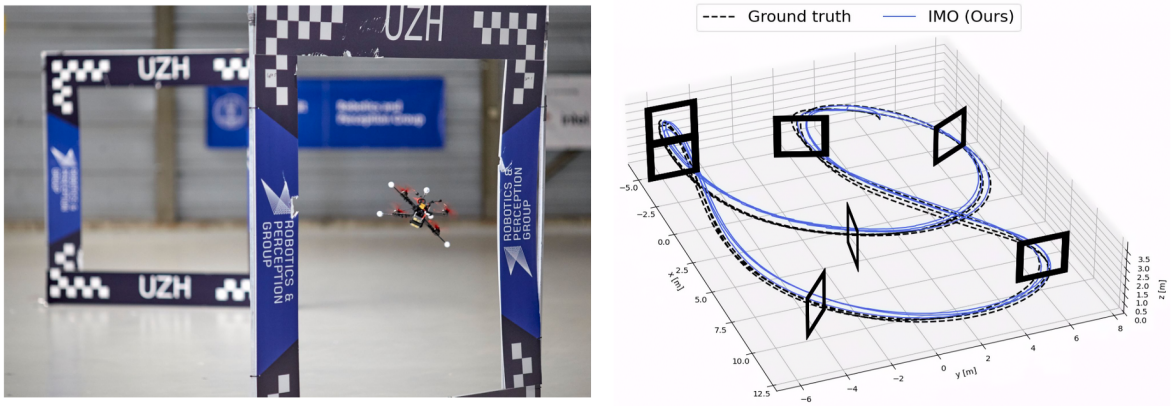

Fig. 1: By combining a temporal convolutional network in a model-based filter, our method is able to estimate the trajectory of an autonomous racing drone using an IMU as the only sensor modality. **Left**: Our autonomous drone flying in a race up to 70 $\frac{km}{h}$. **Right**: The trajectory estimated by our method.

***Abstract*—Inertial odometry is an attractive solution to the problem of state estimation for agile quadrotor flight. It is inexpensive, lightweight, and it is not affected by perceptual degradation. However, only relying on the integration of the inertial measurements for state estimation is infeasible. The errors and time-varying biases present in such measurements cause the accumulation of large drift in the pose estimates. Recently, inertial odometry has made significant progress in estimating the motion of pedestrians. State-of-the-art algorithms rely on learning a motion prior that is typical of humans but cannot be transferred to drones. In this work, we propose a learning-based odometry algorithm that uses an inertial measurement unit (IMU) as the only sensor modality for autonomous drone racing tasks. The core idea of our system is to couple a model-based filter, driven by the inertial measurements, with a learning-based module that has access to the thrust measurements. We show that our inertial odometry algorithm is superior to the state-of-the-art filter-based and optimization-based visual-inertial odometry as well as the state-of-the-art learned-inertial odometry in estimating the pose of an autonomous racing drone. Additionally, we show that our system is comparable to a visual-inertial odometry solution that uses a camera and exploits the known gate location and appearance. We believe that the application in autonomous drone racing paves the way for novel research in inertial odometry for agile quadrotor flight.**

***Index Terms*—Aerial Systems: Perception and Autonomy; Aerial Systems: Applications; Deep Learning Methods**

Manuscript received: October, 27, 2022; Revised January, 20, 2023; Accepted February, 15, 2023.

This paper was recommended for publication by Editor Pauline Pounds upon evaluation of the Associate Editor and Reviewers' comments.

This work was supported by the National Centre of Competence in Research (NCCR) Robotics through the Swiss National Science Foundation (SNSF) and the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No. 871479 (AERIAL-CORE) and the European Research Council (ERC) under grant agreement No. 864042 (AGILEFLIGHT).

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

Digital Object Identifier (DOI): see top of this page.

Supplementary Material

Video:https://youtu.be/2z2Slyt0WlE

Code:https://github.com/uzh-rpg/learned_inertial_model_odometry

I. Introduction

QUADROTORS are extremely agile. Making them autonomous is crucial for time-critical missions, such as search and rescue, aerial delivery, reconnaissance, and even flying cars [1], [2], [3]. State estimation is a core block of the autonomy pipeline.

Inertial odometry (IO) is an excellent solution to the problem of state estimation for agile quadrotor flight. Inertial Measurements Units (IMUs) are inexpensive and ubiquitous sensors that provide linear accelerations and angular velocities. An odometry algorithm only based on inertial measurements has low power and storage requirements, and it does not suffer in scenarios where vision-based odometry systems commonly fail, e.g. large motion blur, high dynamic range scenes, and low texture environments. These are the typical scenarios encountered in agile quadrotor flight. In theory, inertial measurements can be integrated to obtain 6-DoF poses. In practice, the measurements provided by off-the-shelf IMUs are affected by scale factor errors, axis misalignment errors, and time-varying biases [4]. Consequently, the integration accumulates large drift in a short time.

Recently, major progress has been made in inertial odometry for state estimation of pedestrian motion [5], [6], [7]. These works have shown that motion priors can be learned from the repetitive pattern of human gait using IMU measurements. The accuracy of these IO algorithms is comparable to the one of visual-inertial odometry (VIO) algorithms for pedestrian applications.

Differently from the pedestrian motion, the quadrotor motion is not characterized by any significant prior that can be learned from the IMU measurements. For this reason, the performance of the IO methods proposed for pedestrian navigation deteriorates when applied to quadrotors. In this work, we propose a learned inertial odometry algorithm to tackle the problem of state estimation in autonomous drone racing.

Why drone racing? Drone racing requires flying a drone through a sequence of gates in minimum time and has now become a benchmark for the development of new drone technologies that can be turned into real-world products [8], [9]. What makes drone racing so challenging is that the platform is flown at incredible speeds, close to a hundred kilometers per hour, pushing the boundaries of the physics of the vehicle. At such speeds, any little state estimation error can lead to a crash. Research works so far mainly focused on planning and control [10], [11], [12], and relied on external position-tracking systems for state estimation. However, tackling the state estimation problem with only onboard sensing is key to achieving full autonomy. In absence of external position-tracking systems, the only viable solution for state estimation of flying vehicles is VIO [13], [14]. However, VIO fails in scenarios characterized by motion blur, low texture, and high dynamic range due to the unreliability of the vision system. These failure cases are always present in drone racing. Conversely, inertial odometry is not affected by these challenges.

We propose a learned inertial odometry algorithm that uses an IMU as the only sensor modality. Our algorithm combines an Extended Kalman Filter (EKF), which is driven by the inertial measurements, with a learning-based module, which has access to measurements that are related to the drone dynamics in the form of mass-normalized collective thrust. The learning-based module is a temporal convolutional network (TCN) that takes as input a buffer of mass-normalized collective thrust and gyroscope measurements and outputs an estimate of the distance traveled by the quadrotor. These relative positional displacements are then used to update the filter.

We show that the proposed algorithm is superior to the state-of-the-art filter-based and optimization-based VIO algorithms as well as the state-of-the-art learned inertial odometry algorithm TLIO [7] in estimating the pose of a racing quadrotor. Additionally, our approach achieves comparable results to a VIO solution that uses a camera and exploits the known gate location and appearance. We believe that the application in autonomous drone racing shows the benefits of our method and paves the way for novel research in inertial odometry for agile quadrotor flight.

The main contributions of this work are:

- An inertial odometry algorithm based on an EKF that is propagated by the inertial measurements and is updated by the relative positional displacements predicted by a temporal convolutional network. We name our algorithm *IMU-Model Odometry (IMO)*.
- Validation of the proposed system in multiple agile quadrotor flights from the Blackbird dataset [15].
- Thorough experimental analysis for drone racing. Our analysis includes comparisons of the proposed approach against the state-of-the-art VIO and learned IO algorithms as well as against a VIO algorithm that uses a camera to localize to the gates.
- Ablation studies validate both the model-based component, EKF, and the learning-based component, TCN, of our system.

II. Related Work

The most common solution for state estimation of aerial vehicles using only onboard sensing is VIO thanks to its low power, low cost, and low weight requirements. VIO algorithms are divided into two categories according to how they fuse the camera and IMU measurements: filtering methods [16] and fixed-lag smoothing methods [17]. Filtering methods are based on the EKF. These methods propagate the state of the system using the IMU measurements and fuse the camera measurements in the update step. Filtering methods achieve a favorable trade-off between computational requirements and accuracy. Fixed-lag smoothing methods solve a non-linear optimization problem where camera reprojection, IMU, and marginalization residuals are optimized in a sliding window fashion. These methods accumulate less linearization error than filter-based approaches but they are more computationally expensive. We refer the reader to the survey work in [14] for more details.

Recent works have shown relevant progress in inertial odometry for pedestrian motion estimation by combining model-based approaches with deep learning [5], [6], [7]. The works in [5], [6] have shown that 2-DoF pedestrian trajectories can be accurately estimated using neural networks that are trained to predict velocities using IMU data collected from hand-held devices. These works are extended by [7], where it is proposed a 1-D residual convolutional network architecture that learns relative position displacements using IMU data collected from a VR headset device. These relative position displacements are then used as measurements to update an EKF. In robotic applications, deep learning approaches have been used to denoise gyro measurements that are afterward integrated for attitude estimation [18], to denoise IMU measurements before they are included in a VIO algorithm [19], and to compute IMU factors in a sensor fusion algorithm [20].

The works in [21], [22] propose using the quadrotor dynamics in VIO. The commanded collective thrust is integrated to derive pre-integrated positional factors that are included in an optimization-based VIO algorithm. The difference between these two works is in the stochastic model of the external force. In [23], the authors propose a system that estimates the full state of a quadrotor using an IMU and 4 tachometers attached to the rotors. Their approach relies on a heavy recurrent network that is trained to minimize the drift of the full trajectory. The main difference between our work and [23] is that our method uses a lightweight network, which can run on the computer onboard the drone, and does not rely on rotor speed measurements, which are often not available in real-time onboard drone platforms. Another work that relies on rotor speed measurements, as well as inertial measurements, is the one in [24]. This method estimates the tilt, linear and angular

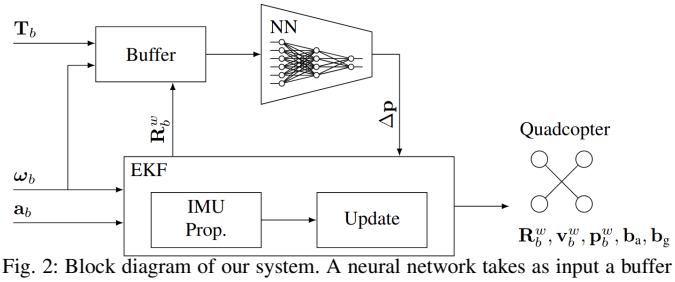

Fig. 2: Block diagram of our system. A neural network takes as input a buffer of collective thrust and gyroscope measurements and outputs relative 3-DoF positional displacements. These displacements are used to update an EKF, which is propagated using the IMU measurements.

velocities as well as additional model parameters, such as the moment of inertia, of the quadrotor but its position.

III. Methodology

An overview of our system is shown in Fig. 2. We train a temporal convolutional network [25] to regress 3-DoF relative displacements from a buffer of length Δt seconds containing mass-normalized collective thrust and gyroscope measurements. Collective thrust has been shown to be the preferred choice of control inputs for agile quadrotor flight [26]. These relative displacements represent the distance traveled by the quadrotor in the time interval Δt. We train the neural net in order to learn a prior on the translational motion of the quadrotor. The displacements predicted by the neural net are used as measurements to update an EKF. The EKF is propagated using a kinematic motion model of the IMU.

In this section, first, we introduce the notation used throughout the paper. Second, we describe how we leverage deep learning in the quadrotor model. Third, we describe the EKF. Fourth, we introduce Gate-IO, a VIO algorithm developed for the task of state estimation in drone racing. We use Gate-IO as a baseline to validate the performance of the proposed system. Last, we describe the main implementation details.

A. Notation

The reference frame $\mathcal{W}$ is the fixed world frame, whose z_w axis is aligned with gravity. The quadrotor body frame is $\mathcal{B}$. For simplicity, the IMU frame is assumed to be the same as $\mathcal{B}$. We use the notation $(\cdot)^w$ to represent a quantity in the world frame $\mathcal{W}$. A similar notation applies to each reference frame. The position, orientation, and velocity of $\mathcal{B}$ with respect to $\mathcal{W}$ at time t_k are written as $\mathbf{p}_{b_k}^w \in \mathbb{R}^3$, $\mathbf{R}_{b_k}^w \in \mathbb{R}^{3\times3}$ part of the rotation group $SO(3)$, and $\mathbf{v}_{b_k}^w \in \mathbb{R}^3$, respectively. The accelerometer and gyroscope bias are written as $\mathbf{b}_a$ and $\mathbf{b}_g$, respectively. The gravity vector in the world frame is written as $\mathbf{g}^w$. We denote estimated quantities with $\hat{(\cdot)}$ and measured quantities with $\tilde{(\cdot)}$.

B. Learned Quadrotor Model

1) Quadrotor Model: The evolution of the position and velocity of the quadrotor platform is described by the following model [21]:

$$\dot{\mathbf{p}}_{b_i}^w = \mathbf{v}_{b_i}^w, \quad \dot{\mathbf{v}}_{bi}^w = \mathbf{R}_{b_i}^w \cdot (\mathbf{T}_i^b + \mathbf{F}_{e_i}^b) + \mathbf{g}^w, \tag{1}$$

where $\mathbf{T}_i^b$ is the mass-normalized collective thrust and $\mathbf{F}_{e_i}^b$ is the external force acting on the platform. We will drop the term mass-normalized when referring to the collective thrust hereafter for the sake of conciseness. Since we do not know the dynamics of the external force, we assume it to be a random variable distributed according to a zero-mean Gaussian distribution [21]. Integrating Eq. 1 in the time interval $[t_i, t_{i+1}]$ with the assumption that $\mathbf{T}_i^b$ is constant in such an interval and using $\mathbf{F}_{e_i}^b = [0, 0, 0]$, we obtain an explicit relation between the relative positional displacement and the thrust:

$$\Delta\mathbf{p}_{i,i+1} = \mathbf{v}_{b_i}^w \cdot \Delta t + 0.5 \cdot \mathbf{g}^w \cdot \Delta t^2 + 0.5 \cdot \mathbf{R}_{b_i}^w \cdot \mathbf{T}_i^b \cdot \Delta t^2, \tag{2}$$

and

$$\mathbf{T}_i^b = 2 \cdot \mathbf{R}_w^{b_i} \left(\frac{\Delta\mathbf{p}_{i,i+1}}{\Delta t^2} - \frac{\mathbf{v}_i^w}{\Delta t} - \mathbf{g}^w \right), \tag{3}$$

where $\Delta\mathbf{p}_{i,i+1} = \mathbf{p}_{b_{i+1}}^w - \mathbf{p}_{b_i}^w$.

2) Neural Net Model: In this work, we use a TCN to learn the positional displacements $\Delta\mathbf{p}_{i,j}$. TCNs have been shown to be as powerful as recurrent networks to model temporal sequences [27] but they are easier to train and deploy on a robotic platform. The neural net takes as input a buffer of collective thrust and gyroscope measurements. These measurements are rotated to the world frame and the bias is removed from the gyroscope measurements. During training, we use ground-truth orientations obtained from a motion capture system. At deployment time, we use the orientations estimated by the EKF. To increase the robustness of the neural net to uncertainty in the estimated orientation, we perturb the ground-truth orientations at training time with zero-mean Gaussian noise. The standard deviation of this noise depends on the expected accuracy of the orientations estimated by the filter. We apply the same strategy to increase the robustness of the neural net with respect to uncertainty in the estimate of the gyroscope bias. Given as input a buffer of measurements in the time interval $\Delta t_{i,j} = t_j - t_i$, the neural net output is the relative displacement $\Delta\mathbf{p}_{i,j}$ in $\Delta t_{i,j}$. We train the neural net with the MSE loss:

$$\mathcal{L}(\Delta\mathbf{p}, \Delta\hat{\mathbf{p}}) = \frac{1}{N} \sum_{j=1}^{N} \|\Delta\mathbf{p}_k - \Delta\hat{\mathbf{p}}_k\|^2 \tag{4}$$

where $\Delta\mathbf{p}$ is the ground-truth positional displacement. We omitted the temporal indices i, j in Eq. 4 for the sake of conciseness.

C. Inertial Model Odometry

In this section, we describe the core components of the EKF. **Filter state**. Our EKF is based on [16]. The full state of the filter is $\mathcal{X} = \{\zeta_1, \cdots, \zeta_m, s\}$, where $s = \{\hat{\mathbf{R}}_{b_i}^w, \hat{\mathbf{v}}_{b_i}^w, \hat{\mathbf{p}}_{b_i}^w, \hat{\mathbf{b}}_{a_i}, \hat{\mathbf{b}}_{g_i}\}$ is the current filter state and $\zeta_j = \{\hat{\mathbf{R}}_{b_j}^w, \hat{\mathbf{p}}_{b_j}^w\}$ is the j-th past state. Following the error-based filtering approach [16], we linearize on a manifold of minimal parameterization of the rotation. The error-state representation of the current filter state is $\delta\mathbf{s} = \{\delta\theta_{b_i}^w, \delta\mathbf{v}_{b_i}^w, \delta\mathbf{p}_{b_i}^w, \delta\mathbf{b}_{a_i}, \delta\mathbf{b}_{g_i}\}$. For an arbitrary variable $\mathbf{x}$, we define $\delta\mathbf{x} = \mathbf{x} \boxminus \hat{\mathbf{x}}$. The operator $\boxminus$ is the difference between the ground-truth and the estimated value. This is the difference operator for variables $\in \mathbb{R}^3$ and the logarithm map operator, $\text{Log}(\cdot)$, such that

$\theta = \text{Log}(\mathbf{R} \cdot \hat{\mathbf{R}}^{-1})$ for $\mathbf{R} \in SO(3)$. Its inverse is the exponential map $\text{Exp}(\cdot)$.

Filter propagation. We use the IMU model as in [16]. The gyroscope model is: $\tilde{\boldsymbol{\omega}}_i = \boldsymbol{\omega}_i + \mathbf{b}_g + \mathbf{n}_g$, and the accelerometer is: $\tilde{\mathbf{a}}_i = \mathbf{a}_i + \mathbf{b}_a + \mathbf{n}_a$. This model represents the measurements as the ground-truth values corrupted by bias and zero-mean Gaussian noise. The accelerometer and gyroscope biases are modeled as a random walk processes with noise $\mathbf{n}_{\mathbf{b}_a}$ and $\mathbf{n}_{\mathbf{b}_g}$, respectively. The kinematic motion model of the filter propagation is

$$\begin{aligned}
\hat{\mathbf{R}}^w_{b_{i+1}} &= \hat{\mathbf{R}}^w_{b_i} \cdot \text{Exp}(\tilde{\boldsymbol{\omega}}_i - \hat{\mathbf{b}}_{g_i}) \cdot \Delta t \\
\hat{\mathbf{v}}^w_{b_{i+1}} &= \hat{\mathbf{v}}^w_{b_i} + \mathbf{g}^w \cdot \Delta t + \hat{\mathbf{R}}^w_{b_i} \cdot (\tilde{\mathbf{a}}_i - \hat{\mathbf{b}}_{a_i}) \cdot \Delta t \\
\hat{\mathbf{p}}^w_{b_{i+1}} &= \hat{\mathbf{p}}^w_{b_i} + \hat{\mathbf{v}}^w_{b_i} \cdot \Delta t + \frac{1}{2} \cdot \Delta t^2 \cdot (\mathbf{g}^w + \hat{\mathbf{R}}^w_{b_i} \cdot (\tilde{\mathbf{a}}_i - \hat{\mathbf{b}}_{a_i})) \\
\hat{\mathbf{b}}_{g_{i+1}} &= \hat{\mathbf{b}}_{g_i}, \hat{\mathbf{b}}_{a_{i+1}} = \hat{\mathbf{b}}_{a_i}.
\end{aligned} \tag{5}$$

To simplify the notation, we dropped the indices that refer to the value of the state between the propagation and update steps and wrote $\Delta t = \Delta t_{i,i+1}$. The linearized propagation model is

$$\delta \mathbf{s}_{i+1} = \mathbf{A}_i \cdot \delta \mathbf{s}_i + \mathbf{B}_i \cdot \mathbf{n}_s, \tag{6}$$

where $\mathbf{n}_s^\intercal = [\mathbf{n}_a^\intercal, \mathbf{n}_g^\intercal, \mathbf{n}_{\mathbf{b}_a}^\intercal, \mathbf{n}_{\mathbf{b}_g}^\intercal]$ is the vector containing the IMU noise and IMU biases noise. The covariance propagation model is

$$\mathbf{P}_{i+1} = \mathbf{A}_i^{\mathcal{X}} \cdot \mathbf{P}_i \cdot (\mathbf{A}_i^{\mathcal{X}})^t + \mathbf{B}_i^{\mathcal{X}} \cdot \mathbf{W}_i \cdot (\mathbf{B}_i^{\mathcal{X}})^t, \tag{7}$$

where $\mathbf{W}_i$ is the covariance matrix containing the IMU noise and IMU bias noise and

$$\mathbf{A}_i^{\mathcal{X}} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_i \end{bmatrix}, \mathbf{B}_i^{\mathcal{X}} = \begin{bmatrix} \mathbf{0} \\ \mathbf{B}_i \end{bmatrix}. \tag{8}$$

The matrix $\mathbf{I}$ denotes the identity matrix whose dimensions depend on the number of past states.

State augmentation. The state augmentation is performed at the filter update frequency. The full state is augmented with a new state that is obtained by propagation till the time when a new measurement from the TCN is available. The covariance propagation is similar as in Eq. 7 with

$$\mathbf{A}_i^{\mathcal{X}} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_i^{\zeta} \\ \mathbf{0} & \mathbf{A}_i \end{bmatrix}, \mathbf{B}_i^{\mathcal{X}} = \begin{bmatrix} \mathbf{0} \\ \mathbf{B}_i^{\zeta} \\ \mathbf{B}_i \end{bmatrix}. \tag{9}$$

Filter update. The measurement update is: $r(\mathcal{X}) = \hat{\mathbf{p}}^w_j - \hat{\mathbf{p}}^w_i = \Delta\tilde{\mathbf{p}}_{ij} + \mathbf{n}_{ij}$, where $\mathbf{n}_{ij}$ is a zero-mean Gaussian noise on the predictions of the network. We set it as a constant value. The Jacobian matrix $\mathbf{H}$ is straightforward to compute. Its entries are all zero except for

$$\mathbf{H}_{\hat{\mathbf{p}}^w_i} = \frac{\partial r(\mathcal{X})}{\partial \delta \hat{\mathbf{p}}^w_i} = -\mathbf{I}_3, \mathbf{H}_{\hat{\mathbf{p}}^w_j} = \frac{\partial r(\mathcal{X})}{\partial \delta \hat{\mathbf{p}}^w_j} = \mathbf{I}_3, \tag{10}$$

where $\mathbf{I}_3$ is the identity matrix with dimensions 3×3. The filter update is performed as

$$\mathbf{K} = \mathbf{P} \cdot \mathbf{H}^t \cdot (\mathbf{H} \cdot \mathbf{P} \cdot \mathbf{H}^t + \boldsymbol{\Sigma}_{ij})^{-1} \tag{11}$$

$$\mathcal{X} = \mathcal{X} \boxplus (\mathbf{K} \cdot (\hat{\mathbf{p}}^w_j - \hat{\mathbf{p}}^w_i - \Delta\tilde{\mathbf{p}}_{ij})) \tag{12}$$

$$\mathbf{P} = (\mathbf{I} - \mathbf{K} \cdot \mathbf{H}) \cdot \mathbf{P} \cdot (\mathbf{I} - \mathbf{K} \cdot \mathbf{H})^t + \mathbf{K} \cdot \boldsymbol{\Sigma}_{ij} \cdot \mathbf{K}^t. \tag{13}$$

The operator $\boxplus$ represents the addition for variables $\in \mathbb{R}^3$ and the update operation $\mathbf{R}' = \text{Exp}(\theta) \cdot \mathbf{R}$ for variables $\in SO(3)$.

D. Gate-IO

In this work, we develop a VIO algorithm for the task of state estimation in autonomous drone racing. We name this algorithm: Gate-IO. Gate-IO fuses the detection of the gate corners with IMU measurements in an EKF. The EKF state and the propagation step are the same as the ones described in Sec. III-C. The residual function of the update step is the reprojection of the gate corners onto the image frame. The world coordinates of the gate corners are known beforehand. The gate corners are detected using a convolutional neural network (CNN) that takes as input raw camera images and outputs the pixel coordinates of all the visible gate corners. The CNN is specifically trained for the type of gates used in our experiments. This approach is based on [9].

E. Implementation Details

We train our neural net on a laptop running Ubuntu 20.04 and equipped with an Intel Core i9 2.3GHz CPU and Nvidia RTX 4000 GPU. At test time, our system runs on an NVIDIA Jetson TX2, which is the computing platform onboard the quadrotor. All the baselines run on the laptop. The thrust and gyroscope measurements are sampled at 100 Hz and are fed to the neural net in an input buffer of length 0.5 seconds. The neural net inference, and consequently the EKF update frequency, is set to 20 Hz. The maximum number of the past states in the filter state is 10. With this setting, our system runs at $\sim$180 Hz on the Jetson TX2 onboard the quadrotor. The noise values are: $\sigma_{\mathbf{n}_a} = 0.01 \frac{\text{m}}{\text{s}^2}, \sigma_{\mathbf{n}_g} = 0.001 \frac{\text{rad}}{\text{s}}, \sigma_{\mathbf{n}_{\mathbf{b}_a}} = 0.001 \frac{\text{m}}{\text{s}^2} \frac{1}{\text{s}^{0.5}}, \sigma_{\mathbf{n}_{\mathbf{b}_g}} = 0.0001 \frac{\text{rad}}{\text{s}} \frac{1}{\text{s}^{0.5}}, \sigma_{\mathbf{n}_{ij}} = 0.01\text{m}$. The value of $\sigma_{\mathbf{n}_{ij}}$ has been chosen according to the expected error of the network predictions. We initialize our method as well as all the baselines with the ground-truth initial pose obtained from a motion caption system. In the experiments on the Blackbird dataset, the IMU biases are initialized to zero. In the drone racing experiments, the IMU biases are initialized using an offline calibration routine.

IV. Experiments

We compare our system to the following baselines:

- TLIO [7]. TLIO is the state-of-the-art inertial odometry algorithm. It uses a residual network that takes as input a buffer of IMU measurements in a time window and outputs the relative distance that the IMU has traveled in such a time window. The network output is used as a measurement to update an EKF. The EKF is propagated using the IMU measurements.
- SVO [28]. SVO is a semi-direct visual odometry front-end. In this work, we combine SVO with a sliding-window optimization-based backend [17]. The code is available open-source [1].
- OpenVINS [29]. OpenVINS is a state-of-the-art filter-based visual-inertial odometry algorithm. It ranks 1-st amongst the open-source algorithms on the UZH-FPV drone racing dataset [30].

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

TABLE I: Blackbird dataset evaluation. ATE_T is in meters and ATE_R is in degrees. In bold is the best value and in underlined is the second-best value.

Trajectory	Eval. metric	Algorithm			
		OpenVINS	SVO	TLIO	IMO (ours)
Clover	ATE_T [m]	0.50	0.77	0.75	**0.41**
	ATE_R [deg]	**2.62**	3.51	3.05	3.05
Egg	ATE_T [m]	**1.07**	2.49	1.31	1.15
	ATE_R [deg]	2.71	3.42	2.97	**2.45**
Half Moon	ATE_T [m]	**0.37**	1.10	1.20	0.76
	ATE_R [deg]	**2.29**	8.48	8.74	4.14
Star	ATE_T [m]	2.78	2.78	2.04	**1.22**
	ATE_R [deg]	7.43	10.16	2.96	**2.76**
Winter	ATE_T [m]	**0.12**	0.29	1.13	0.22
	ATE_R [deg]	**0.87**	1.18	12.15	2.32

- Gate-IO. Gate-IO is a VIO algorithm customized for drone racing. We refer the reader to Sec. III-D for more details.

Following the best practices in the evaluation of VIO algorithms [31], we use the evaluation metrics: translation absolute trajectory error (ATE_T) [m], rotation absolute trajectory error (ATE_R) [deg], relative translation and rotation errors. We refer the reader to [31] for a detailed description of these metrics.

A. Blackbird Dataset

Experiment Setup: In this set of experiments, we evaluate the performance of our system and the baselines on the Blackbird dataset [15]. The Blackbird dataset provides rotor speed measurements recorded onboard a quadrotor flying in a motion capture system, which we use to compute mass-normalized collective thrust measurements for our network. In addition, this dataset also contains IMU measurements and photorealistic images of synthetic scenes. We select 5 trajectories from the dataset: *clover*, *egg*, *half moon*, *star*, and *winter*, with peak velocities of 5, 8, 4, 5, 4 $\frac{m}{s}$, respectively. For each trajectory, 70% of the data is used for training, 15% of the data is used for validation and, 15% of the data is used for testing. In total, the training, validation, and test datasets contain approx. 10, 2.5, and 2.5 min of flight data, respectively. We use the training and validation dataset to train our network and the TLIO network and to tune the parameters of SVO and OpenVINS.

Evaluation: We report the absolute trajectory errors in Table I and the relative translation and rotation errors in Fig. 3 and Fig. 4, respectively. Our system outperforms TLIO in all the sequences. The smallest and the largest improvements of the ATE_T are equal to 12% and 80% in the sequences *egg* and *winter*, respectively. The best VIO algorithm is OpenVINS. The performance of our system is comparable to the one of OpenVINS in all the sequences. The largest difference in the ATE_T in favor of our system is in the *star* sequence. In this sequence, the high yaw rate and the resulting large optical flow render feature tracking more difficult and, consequently, degrades the estimation accuracy of the VIO system.

B. Drone Racing

Experiment Setup: To validate our system in drone racing tasks, we designed a custom-made quadrotor platform. The platform has a total weight of 750 grams and it can produce a maximum thrust larger than 40 N. The weight and power of

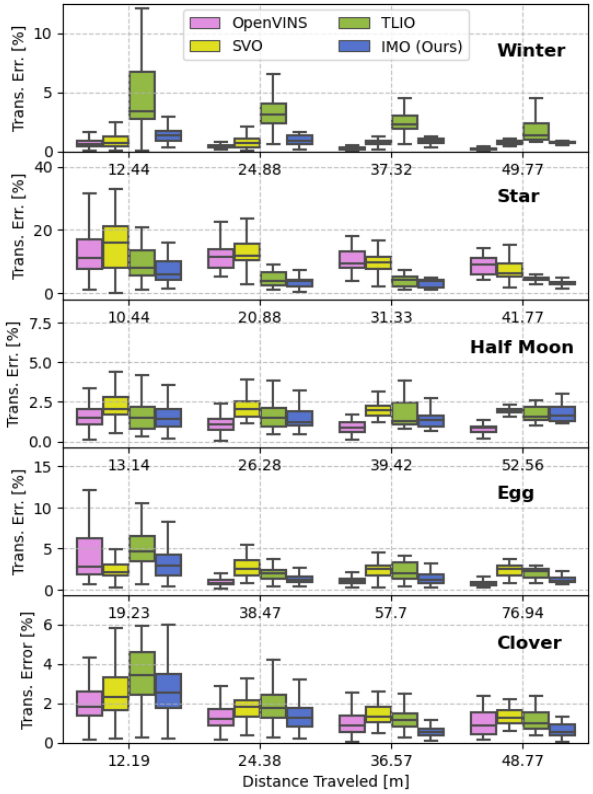

Fig. 3: Blackbird dataset evaluation. Relative translation errors achieved by OpenVINS, SVO, TLIO, and IMO (ours). The quantity on the x-axis is the distance traveled corresponding to 2.5, 5, 7.5, and 10 % of the total distance traveled.

this platform are comparable to the ones used by professional pilots in drone racing competitions. The main computational unit is an NVIDIA Jetson TX2. Our quadrotor is also equipped with an Intel RealSense T265 camera. This camera has a fisheye lens and provides grey-scale images with a resolution of 848×800 pixels. The camera also contains an integrated IMU. SVO, OpenVINS, and Gate-IO use the monocular grey-scale images and inertial measurements from the Intel RealSense T265 while TLIO only uses the inertial measurements. More details about the quadrotor platform and the onboard software stack can be found in [32]. In all our experiments, the quadrotor is flown in a motion capture system and controlled by the method proposed in [33]. The controller [33] outputs collective thrusts. These commanded collective thrusts are used as input to our network.

Evaluation: We evaluate the performance of our system in estimating the pose of the quadrotor in a drone racing scenario, cf. Fig. 1. The racing track is designed by a professional drone racing pilot and has been used in related works on drone racing [10], [11]. In each race, the quadrotor flies 3 laps of the track. In our experiments, the top speed of the autonomous drone is approx. $70\frac{km}{h}$. We use training, validation, and test datasets containing approx. 5, 1.5, and 1.5 min of flight data, respectively. It takes approx. 6 sec to complete a lap of the racing track. A visualization of the estimated trajectory by Gate-IO, TLIO, and our algorithm in a race is in Fig. 5. The two VIO baselines accumulate large drift, cf. Table II. We

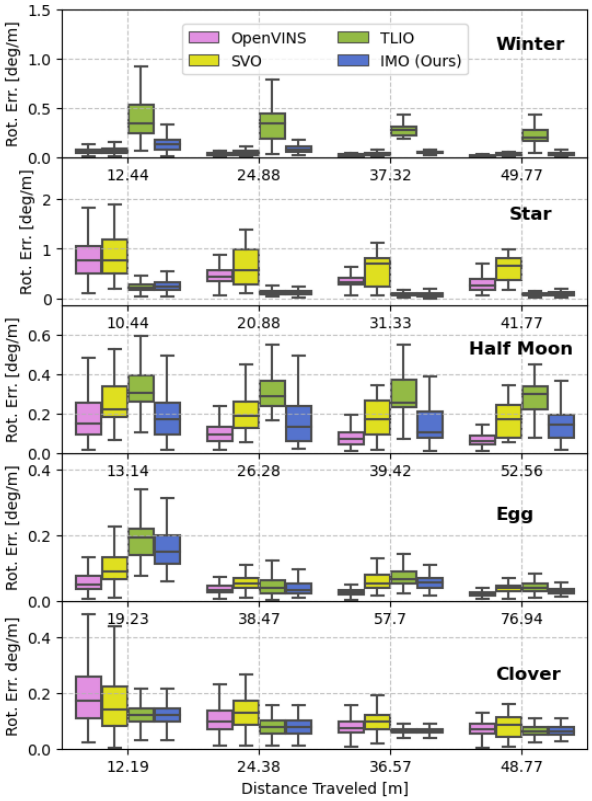

Fig. 4: Blackbird dataset evaluation. Relative rotation errors achieved by OpenVINS, SVO, TLIO, and IMO (ours). The quantity on the x-axis is the distance traveled corresponding to 2.5, 5, 7.5, and 10 % of the total distance traveled.

TABLE II: ATE_T in meters and ATE_R in degrees of the racing trajectory. In bold is the best value, and in underlined is the second-best value.

Trajectory	Eval. metric	Algorithm				
		OpenVINS	SVO	Gate-IO	TLIO	IMO (ours)
Racing	ATE_T [m]	98.20	47.20	**0.48**	1.21	<u>0.56</u>
	ATE_R [deg]	99.00	123.00	**2.20**	3.30	<u>2.80</u>

do not include them in Fig. 5 for the sake of clarity. These results confirm that the classical VIO algorithms fail in drone racing due to perception challenges. The relative translation and rotation errors in a race are shown in Fig. 6 and Fig. 7, respectively. The average absolute trajectory errors computed on the test dataset are in Table II. The ATE_T achieved by our system outperforms TLIO by 54% and is similar to the one achieved by Gate-IO.

1) Learning the Quadrotor Dynamics: In this section, we validate the proposed learning-based module, which predicts relative positional displacements from a window of commanded collective thrusts and gyroscope measurements, in two ablation studies.

In the first ablation study, we analyze the predictions of the force acting on the quadrotor platform. We compare the predictions of our neural net to the ones obtained by using a motion capture system and to the commanded collective thrust. The forces predicted by our neural net are derived from Eq. 3 where the position and velocity measurements are obtained from a 3-DoF trajectory that is computed by concatenating the predicted positional displacements of the neural net. The orientation measurements are from the motion capture system.

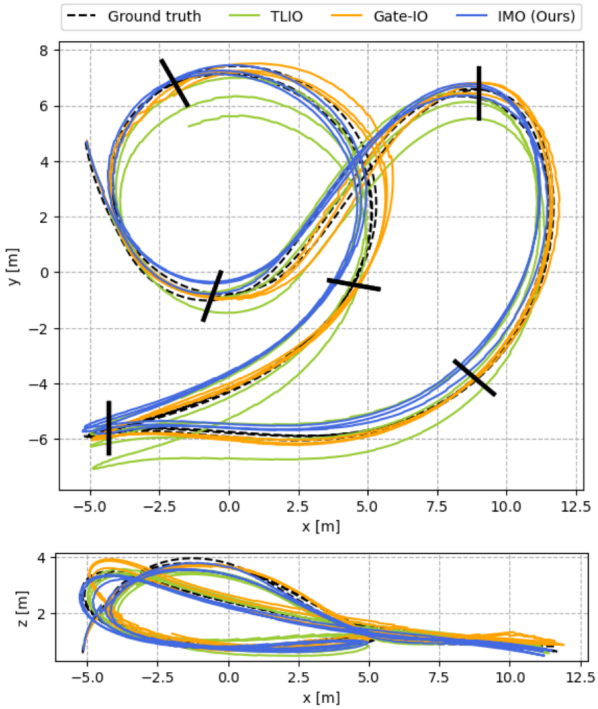

Fig. 5: Drone racing evaluation. Trajectory estimated by TLIO, Gate-IO and IMO.

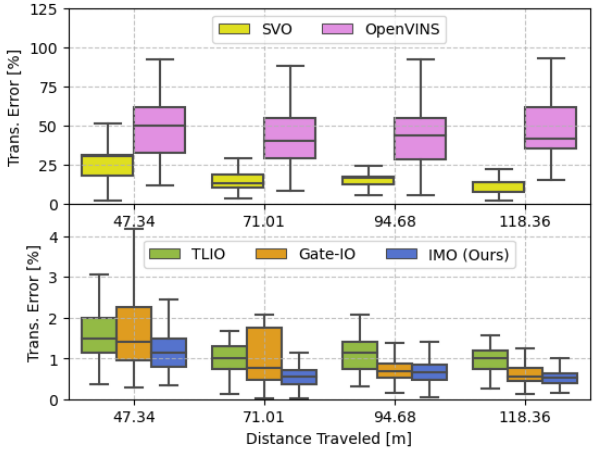

Fig. 6: Drone racing evaluation. Relative translation errors achieved by SVO, OpenVINS, TLIO, Gate-IO and IMO.

Similarly, the forces predicted by the motion capture system are derived from Eq. 3. We show the result of this comparison in Fig. 8. The forces predicted by our neural net match the ones derived from the motion capture system. We conclude that our neural net learns how to map from the commanded collective thrust to the actual force acting on the quadrotor. In particular, the neural net also learns to predict the drag force along the body x and y axes, cf. T_x and T_y in Fig. 8.

In the second ablation study, we compare our system against an algorithm that uses an EKF where the measurements $\Delta\mathbf{p}$ are obtained from Eq. 2 using the commanded collective thrust and orientation and velocity measurements from the motion capture system. We call this algorithm Model-EKF. The propagation step is driven by the IMU measurements as in Eq. 5. The purpose of this study is to show that learning

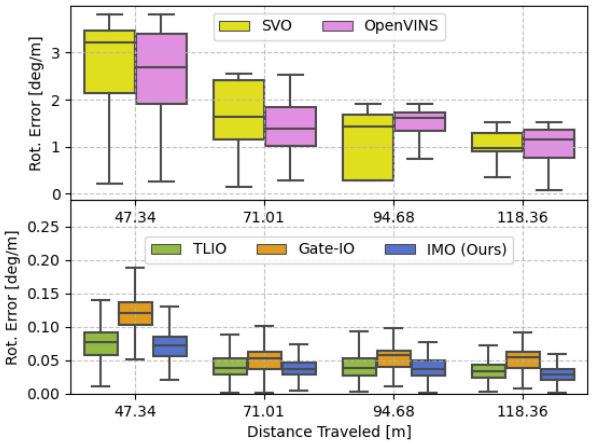

Fig. 7: Drone racing evaluation. Relative rotation errors achieved by SVO, OpenVINS, TLIO, Gate-IO, and IMO.

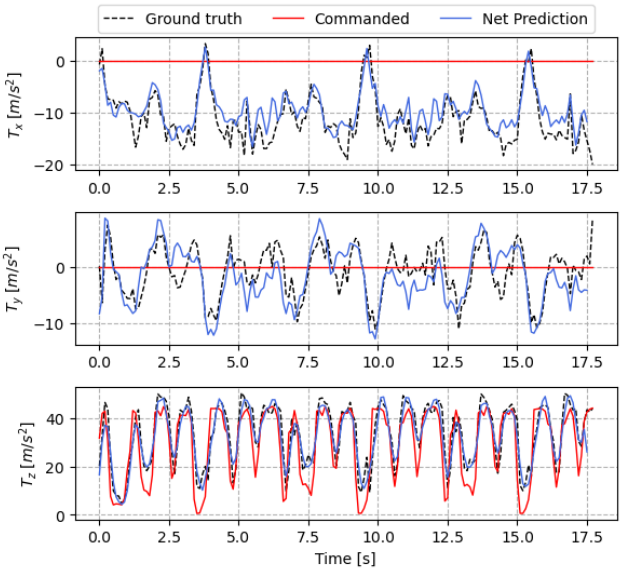

Fig. 8: Drone racing evaluation. Comparison of the predictions of the force acting on the quadrotor. *Ground truth* is the force derived from Eq. 3 where the position, velocity, and orientation are from the motion capture system. *Net Prediction* is the force derived from Eq. 3 where the position and velocity are from a 3-DoF trajectory obtained by concatenating the network outputs. The orientation is from the motion caption system. *Commanded* is the commanded collective thrust.

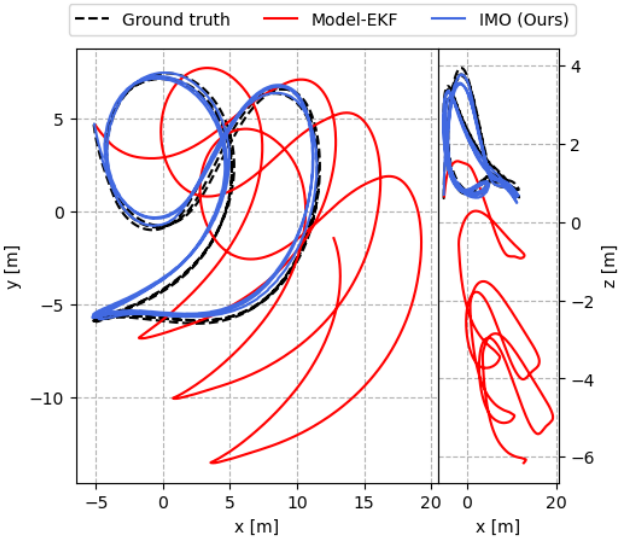

Fig. 9: Drone racing evaluation. Comparison against Model-EKF. In Model-EKF the measurements $\Delta\mathbf{p}$ are obtained from Eq. 2 using the commanded collective thrust as well as orientation and velocity measurements from the motion capture system.

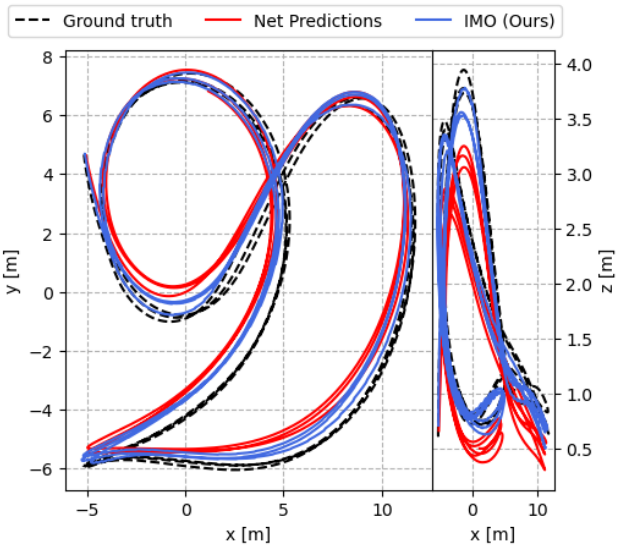

Fig. 10: Drone racing evaluation. Comparison of our system against a trajectory computed by concatenating the positional displacements predicted by the neural net.

to predict the mismatch between the commanded and applied thrust is essential for accurate state estimation. A visualization of the estimated trajectory by Model-EKF and the proposed system on one of the test sequences of the racing trajectory is in Fig. 9. The ATE_T and ATE_R achieved by our system are 0.56 m and 2.8 deg, respectively. The ATE_T and ATE_R achieved by Model-EKF are 10.10 m and 4.6 deg, respectively. These results confirm that the learning-based component of our system is essential to achieve accurate state estimation.

2) Filter validation: In this section, we validate the use of the EKF in combination with the neural net. To this end, we compare the 3-DoF trajectory estimated by our system against a trajectory computed by concatenating the positional displacements predicted by the neural net. A visualization of the estimated trajectories on one of the test sequences of the racing trajectory is in Fig. 10. The ATE_T achieved by our system is 0.56 m. The ATE_T achieved by concatenating the net predictions is 3.20 m. This result confirms the superior performance of using the neural net predictions as measurements in an IMU-driven EKF.

V. Discussion and Conclusion

In this work, we present a new approach to estimating the state of a quadrotor in autonomous drone racing. Our method uses a temporal convolutional network to predict the translational motion of the quadrotor using mass-normalized collective thrusts and gyroscope measurements. The network outputs are relative positional displacements used to update an EKF. The EKF is propagated using the IMU measurements.

We have demonstrated that the proposed approach is able to accurately estimate the state of the quadrotor during an autonomous race only relying on an off-the-shelf IMU. In our experiments, we validate the individual components of our system and we demonstrate that it is superior to the state-of-the-art VIO and IO algorithms in estimating the pose of a racing drone. Additionally, our system can achieve trajectory estimates similar to those estimated by a VIO that relies on a camera to perform gate detection and has access to the position of the gates.

The main limitation of our approach is that it cannot generalize to trajectories that have not been seen at training time. However, in drone racing competitions, the track is known beforehand. Human pilots spend hours or even days of practice on the race track before the competition. Similarly, our system can be trained with the data collected during practice time and then deployed during the competition. Future work will investigate how to generalize to trajectories that have not been seen at training time. A possible solution is to train the network to estimate the positional displacements in the drone body frame. These displacements can be integrated into a VIO system in order to reduce the dependency on visual inputs. For example, the learned displacement can compensate for informationless visual inputs, e.g. in low-texture scenarios, or low-rate camera measurements.

Although our work focuses specifically on autonomous drone racing, we believe that the proposed approach could have broader implications for reliable state estimation in agile drone flight. In several tasks such as routine inspection and surveillance, the drone is required to fly trajectories that are known beforehand. In these situations, our system can be integrated with a visual-based estimator in order to increase reliability when the visual measurements are degraded, e.g. in low-light conditions.

REFERENCES

[1] G. Loianno and D. Scaramuzza, "Special issue on future challenges and opportunities in vision-based drone navigation," *J. Field Robot.*, vol. 37, no. 4, pp. 495–496, 2020.

[2] S. Watkins, J. Burry, A. Mohamed, M. Marino, S. Prudden, A. Fisher, N. Kloet, T. Jakobi, and R. Clothier, "Ten questions concerning the use of drones in urban environments," *Building and Environment*, vol. 167, p. 106458, 2020.

[3] T. Rakha and A. Gorodetsky, "Review of unmanned aerial system (uas) applications in the built environment: Towards automated building inspection procedures using drones," *Automation in Construction*, vol. 93, pp. 252–264, 2018.

[4] Y. Yang, P. Geneva, X. Zuo, and G. Huang, "Online imu intrinsic calibration: Is it necessary?" *Proc. of Robotics: Science and Systems (RSS), Corvallis, Or*, 2020.

[5] 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, no. 1, 2018.

[6] S. Herath, H. Yan, and Y. Furukawa, "Ronin: Robust neural inertial navigation in the wild: Benchmark, evaluations, & new methods," *IEEE Int. Conf. Robot. Autom. (ICRA)*, pp. 3146–3152, 2020.

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

[8] R. Madaan, N. Gyde, S. Vemprala, M. Brown, K. Nagami, T. Taubner, E. Cristofalo, D. Scaramuzza, M. Schwager, and A. Kapoor, "Airsim drone racing lab," in *NeurIPS 2019 Comp. and Demo. Track*, 2020.

[9] P. Foehn, D. Brescianini, E. Kaufmann, T. Cieslewski, M. Gehrig, M. Muglikar, and D. Scaramuzza, "Alphapilot: Autonomous drone racing," *Autonomous Robots*, pp. 1–14, 2021.

[10] P. Foehn, A. Romero, and D. Scaramuzza, "Time-optimal planning for quadrotor waypoint flight," *Science Robotics*, vol. 6, no. 56, 2021.

[11] A. Romero, S. Sun, P. Foehn, and D. Scaramuzza, "Model predictive contouring control for time-optimal quadrotor flight," *IEEE Trans. Robot.*, 2022.

[12] R. Penicka and D. Scaramuzza, "Minimum-time quadrotor waypoint flight in cluttered environments," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 5719–5726, 2022.

[13] D. Scaramuzza and Z. Zhang, "Visual-inertial odometry of aerial robots," *Encyclopedia of Robotics*, 2019.

[14] G. Huang, "Visual-inertial navigation: A concise review," in *IEEE Int. Conf. Robot. Autom. (ICRA)*. IEEE, 2019, pp. 9572–9582.

[15] 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," in *Int. Symp. Experimental Robotics (ISER)*, 2018.

[16] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint kalman filter for vision-aided inertial navigation," in *IEEE Int. Conf. Robot. Autom. (ICRA)*. IEEE, 2007, pp. 3565–3572.

[17] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual-inertial SLAM using nonlinear optimization," *Int. J. Robot. Research*, 2015.

[18] 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.

[19] M. Zhang, M. Zhang, Y. Chen, and M. Li, "Imu data processing for inertial aided navigation: A recurrent neural network based approach," *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2021.

[20] R. Buchanan, M. Camurri, F. Dellaert, and M. Fallon, "Learning inertial odometry for dynamic legged robot state estimation," *Conf. on Robot. Learning (CoRL)*, 2021.

[21] B. Nisar, P. Foehn, D. Falanga, and D. Scaramuzza, "Vimo: Simultaneous visual inertial model-based odometry and force estimation," *IEEE Robot. Autom. Lett.*, vol. 4, no. 3, pp. 2785–2792, 2019.

[22] Z. Ding, T. Yang, K. Zhang, C. Xu, and F. Gao, "Vid-fusion: Robust visual-inertial-dynamics odometry for accurate external force estimation," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2021, pp. 14 469–14 475.

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

[24] J. Svacha, J. Paulos, G. Loianno, and V. Kumar, "Imu-based inertia estimation for a quadrotor using newton-euler dynamics," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 3861–3867, 2020.

[25] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[26] E. Kaufmann, L. Bauersfeld, and D. Scaramuzza, "A benchmark comparison of learned control policies for agile quadrotor flight," *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2022.

[27] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv preprint arXiv:1803.01271*, 2018.

[28] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza, "SVO: Semidirect visual odometry for monocular and multicamera systems," *IEEE Trans. Robot.*, vol. 33, no. 2, pp. 249–265, 2017.

[29] P. Geneva, K. Eckenhoff, W. Lee, Y. Yang, and G. Huang, "Openvins: A research platform for visual-inertial estimation," in *IEEE Int. Conf. Robot. Autom. (ICRA)*. IEEE, 2020, pp. 4666–4672.

[30] J. Delmerico, T. Cieslewski, H. Rebecq, M. Faessler, and D. Scaramuzza, "Are we ready for autonomous drone racing? the UZH-FPV drone racing dataset," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2019.

[31] Z. Zhang and D. Scaramuzza, "A tutorial on quantitative trajectory evaluation for visual(-inertial) odometry," in *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2018.

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

[33] Y. Song, M. Steinweg, E. Kaufmann, and D. Scaramuzza, "Autonomous drone racing with deep reinforcement learning," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 1205–1212.