Learned Inertial Odometry for Autonomous Drone Racing

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

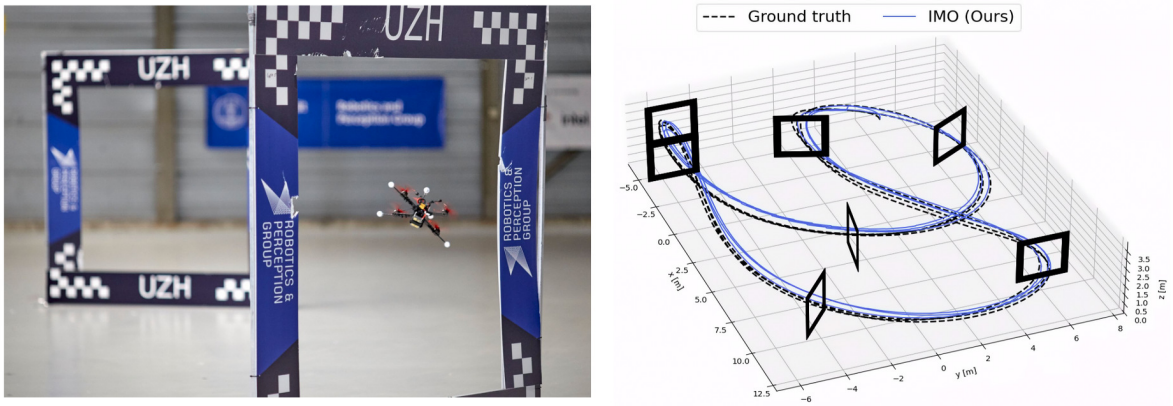

Fig. 1: By combining a temporal convolutional network in a model-based filter, our method is able to estimate the trajectory of an autonomous racing drone using an IMU as the only sensor modality. **Left**: Our autonomous drone flying in a race up to 70 $\frac{km}{h}$. **Right**: The trajectory estimated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

Supplementary Material

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

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

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).

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).

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 robotic applications, deep learning approaches have been used to denoise gyro measurements that are afterward integrated for attitude estimation [8], to denoise IMU measurements before they are included in a VIO algorithm [9], and to compute IMU factors in a sensor fusion algorithm [10]. In [11], 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.

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 [12], [13]. 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. Tackling the state estimation problem with only onboard sensing is key to achieving full autonomy. The main solution for state estimation of flying vehicles is VIO [14], [15]. 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. Differently, from [11], we propose 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.

We show that the proposed algorithm is superior to the state-of-the-art filter-based [16] and optimization-based [17] 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. Moreover, we validate the proposed system in multiple agile quadrotor flights from the Blackbird dataset [18]. 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. An extended version of this work can be found in [19].

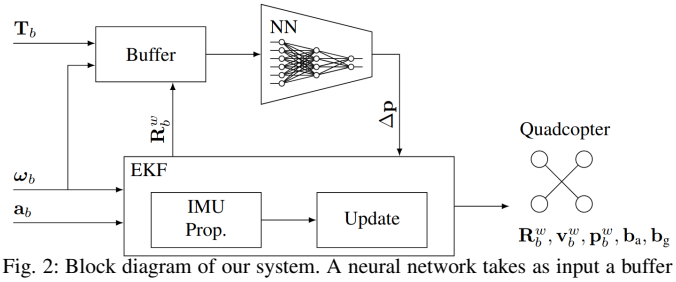

Fig. 2: Block diagram of our system. A neural network takes as input a buffer of collective thrust and gyroscope measurements and outputs relative 3-DoF positional displacements. These displacements are used to update an EKF, which is propagated using the IMU measurements.

II. Methodology

An overview of our system is shown in Fig. 2. We train a temporal convolutional network [20] 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 [21]. 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, we describe how we leverage deep learning in the quadrotor model. We refer to [19] for a more detailed description on the EKF and implementation details.

A. Learned Quadrotor Model

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 gravity vector in the world frame is written as $\mathbf{g}^w$.

1) Quadrotor Model: The evolution of the position and velocity of the quadrotor platform is described by the following model [22]:

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

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 [22]. 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 [23] 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 $\Delta \hat{\mathbf{p}}_{i,j}$ in $\Delta t_{i,j}$. We train the neural net with the MSE loss:

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

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.

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.

III. Experiments

We compare our system to TLIO [7], SVO [24] [1], OpenVINS [16], and Gate-IO [19].

We use the evaluation metrics: translation absolute trajectory error (ATE_T) [m], rotation absolute trajectory error (ATE_R) [deg], relative translation and rotation errors [25].

A. Blackbird Dataset

Experiment Setup: In this set of experiments, we evaluate the performance of our system and the baselines on the Blackbird dataset [18]. 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.

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]	<u>0.50</u>	0.77	0.75	**0.41**
	ATE_R [deg]	**2.62**	3.51	<u>3.05</u>	<u>3.05</u>
Egg	ATE_T [m]	**1.07**	2.49	1.31	<u>1.15</u>
	ATE_R [deg]	<u>2.71</u>	3.42	2.97	**2.45**
Half Moon	ATE_T [m]	**0.37**	1.10	1.20	<u>0.76</u>
	ATE_R [deg]	**2.29**	8.48	8.74	<u>4.14</u>
Star	ATE_T [m]	2.78	2.78	<u>2.04</u>	**1.22**
	ATE_R [deg]	7.43	10.16	<u>2.96</u>	**2.76**
Winter	ATE_T [m]	**0.12**	0.29	1.13	<u>0.22</u>
	ATE_R [deg]	**0.87**	<u>1.18</u>	12.15	2.32

Evaluation: We report the absolute trajectory errors in Table I. 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, degrade the estimation accuracy of the VIO system.

B. Drone Racing

Experiment Setup: To validate our system in drone racing tasks, we use a custom-made quadrotor platform [26]. Our quadrotor is also equipped with an Intel RealSense T265 camera. SVO, OpenVINS, and Gate-IO use monocular grey-scale images and inertial measurements from the Intel RealSense T265 while TLIO only uses the inertial measurements. In all our experiments, the quadrotor is flown in a motion capture system and controlled by the method proposed in [27]. The controller [27] 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 [28], [29]. 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

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

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>

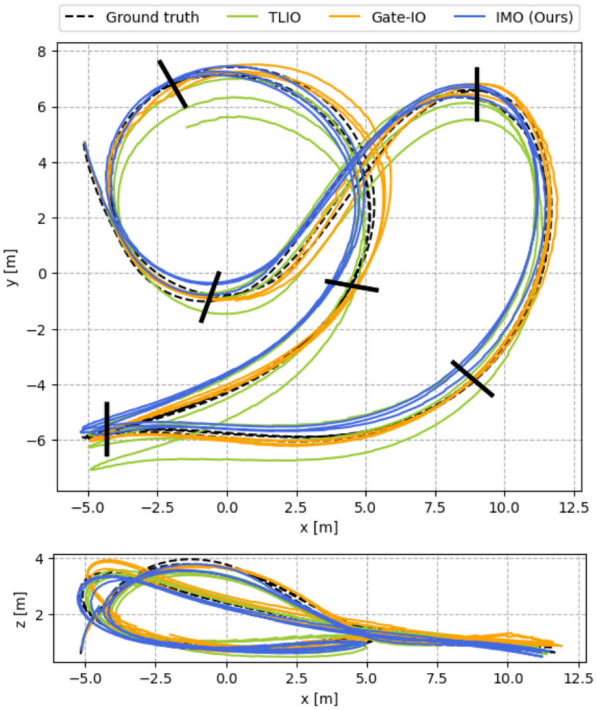

Fig. 3: Drone racing evaluation. Trajectory estimated by TLIO, Gate-IO and IMO.

racing track. A visualization of the estimated trajectory by Gate-IO, TLIO, and our algorithm in a race is in Fig. 3. The two VIO baselines accumulate large drift, cf. Table II. We do not include them in Fig. 3 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. 4 and Fig. 5, 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.

In [19], we include ablation studies to validate our system design choices.

IV. Discussion and Conclusion

In this work, we present a learned inertial odometry algorithm to estimate the state of a quadrotor in autonomous drone racing. We demonstrate that our system 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

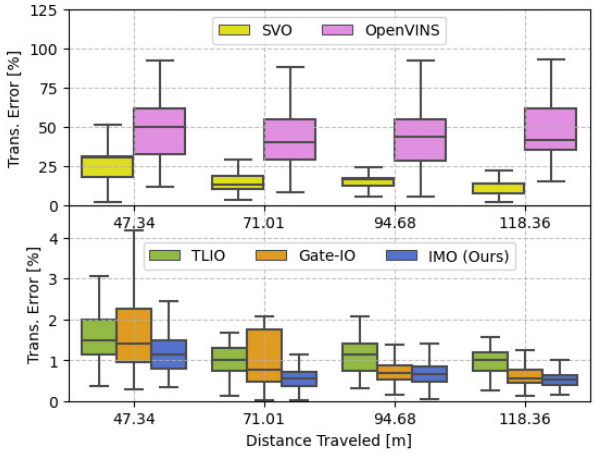

Fig. 4: Drone racing evaluation. Relative translation errors achieved by SVO, OpenVINS, TLIO, Gate-IO and IMO.

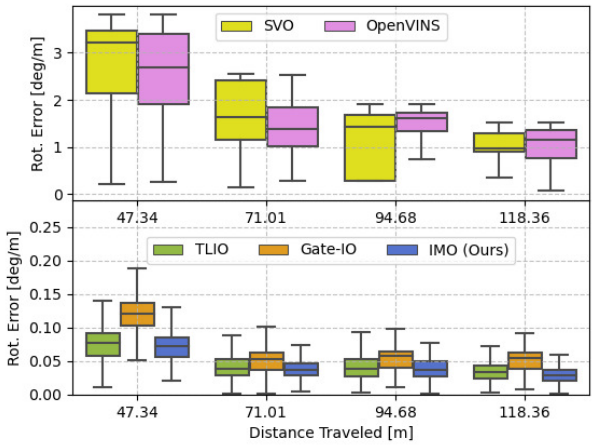

Fig. 5: Drone racing evaluation. Relative rotation errors achieved by SVO, OpenVINS, TLIO, Gate-IO, and IMO.

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] 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.

[9] 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.

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

[11] 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.

[12] 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.

[13] 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.

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

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

[16] 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.

[17] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual–inertial odometry using nonlinear optimization," *The Int. Journal of Robot. Research*, vol. 34, no. 3, pp. 314–334, 2015.

[18] 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.

[19] G. Cioffi, L. Bauersfeld, E. Kaufmann, and D. Scaramuzza, "Learned inertial odometry for autonomous drone racing," *IEEE Robot. Autom. Lett.*, 2023.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

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

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