DIVE: Deep Inertial-Only Velocity Aided Estimation for Quadrotors

Angad Bajwa, Charles Champagne Cossette, *Member, IEEE*,
Mohammed Ayman Shalaby, *Graduate Student Member, IEEE*, and James Richard Forbes

***Abstract*—This letter presents a novel deep-learning-based solution to the problem of quadrotor inertial navigation. Visual-inertial odometry (VIO) is often used for quadrotor pose estimation, where an inertial measurement unit (IMU) provides a motion prior. When VIO fails, IMU dead reckoning is often used, which quickly leads to significant pose estimation drift. Learned inertial odometry leverages deep learning and model-based filtering to improve upon dead reckoning. Efforts for quadrotors, however, rely on sensors other than, or in addition to, an IMU, or have only been proven on a specific set of trajectories. The proposed generalizable approach regresses a 3D velocity estimate from only a history of IMU measurements, and the learned outputs are applied as a correction to an on-manifold Extended Kalman Filter. The proposed algorithm is shown to be superior to the state-of-the-art in learned inertial odometry. A 42% improvement in localization accuracy is shown over the state-of-the-art on an in-distribution testing set, and a 22% improvement is shown on an out-of-distribution testing set. Additionally, the proposed algorithm shows a 43% improvement over dead reckoning in VIO failure scenarios.**

***Index Terms*—Localization, deep-learning methods, aerial systems: perception and autonomy, inertial state estimation.**

I. Introduction

Multirotors are versatile vehicles that offer a low-cost, maneuverable, and highly-controllable platform for a variety of applications. Generally, multirotors are operated by a human pilot with some sort of absolute positioning system, such as the Global Positioning System (GPS). In recent years, large strides have been made to commercialize autonomous multirotor navigation in GPS-denied environments using Visual-inertial and LiDAR-inertial odometry (VIO/LIO) [1], [2]. VIO, in particular, is able to achieve near-centimeter localization accuracy [3], and requires only a camera and an inertial measurement unit (IMU), which are both low-cost and lightweight sensors.

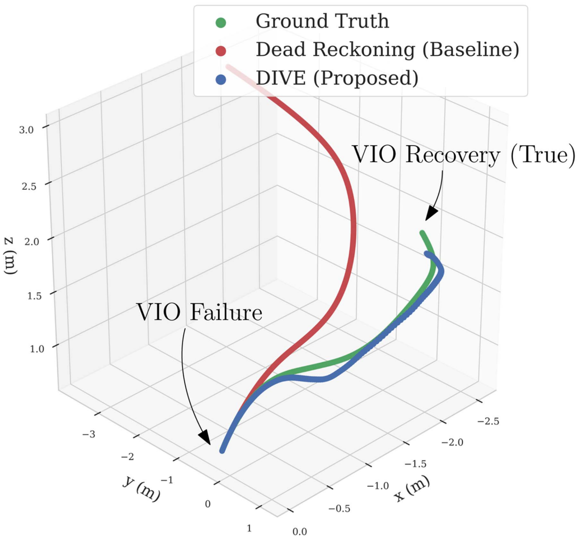


Fig. 1. Position estimation during a 6 s period of VIO failure by DIVE (blue) and IMU dead reckoning (red). The proposed algorithm shows significantly less drift than the baseline, and is able to generate accurate velocity estimates even without exteroceptive corrections.

Including an IMU instead of estimating purely from camera egomotion is useful as it allows for a completely proprioceptive backup when visual odometry (VO) fails, such as in low-texture environments and scenarios with occlusions or motion blur.

IMU-based dead reckoning, while accurate in the short term, is prone to long-term drift due to intrinsic IMU error and time-varying bias. To this end, significant effort has been made to improve the accuracy of inertial-only odometry, with particular success for pedestrian and ground-vehicle estimation [4], [5]. Pedestrians and ground-vehicles have repetitive motion patterns and strong dynamic constraints that can be exploited to yield a generalizable inertial odometry algorithm that is comparable in long-term accuracy to VIO.

Multirotors, however, are far less dynamically constrained compared to ground vehicles and do not show as much repetitive motion as pedestrians. As such, it is difficult for inertial-only multirotor navigation approaches to match VIO's accuracy. The research into improving inertial-only odometry for multirotors has involved other sensors in addition to the IMU [6], or has been trajectory specific [7]. In this work, a novel learned inertial navigation algorithm is proposed that is generalizable to any multirotor with an IMU. The proposed IMU-only algorithm couples an on-manifold Extended Kalman Filter (EKF), which is

Manuscript received 26 July 2023; accepted 30 December 2023. Date of publication 27 February 2024; date of current version 7 March 2024. This letter was recommended for publication by Associate Editor E. Asadi and Editor P. Pounds upon evaluation of the reviewers' comments. This work was supported in part by the NSERC Alliance Mission and Discovery Grant Programs, and in part by ARA Robotics. *(Corresponding author: Angad Bajwa.)*

Angad Bajwa is with the Department of Mechanical and Mechatronics Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: a29bajwa@uwaterloo.ca).

Charles Champagne Cossette, Mohammed Ayman Shalaby, and James Richard Forbes are with the Department of Mechanical Engineering, McGill University, Montreal, QC H3A 0C3, Canada (e-mail: charles.cossette@mail.mcgill.ca; mohammed.shalaby@mail.mcgill.ca; james.richard.forbes@mcgill.ca).

Lastly, this letter is accompanied by an open-source implementation at https://github.com/decargroup/DIVE.

Digital Object Identifier 10.1109/LRA.2024.3370006

propagated by the inertial measurements, with a convolutional neural net (CNN) based module. The CNN takes a history of orientations and linear accelerations and outputs an inferred velocity in the gravity-aligned frame, which is then used as a correction to the filter's state estimate.

The main contributions of this letter are

- a model for obtaining an inferred velocity from a history of orientations and linear accelerations using a CNN,
- a novel solution to the learned inertial odometry problem that fuses the inferred velocity from the CNN using an EKF, and
- an approach that is largely generalizable to any multirotor with an IMU.

The proposed algorithm, **D**eep **I**nertial-Only **V**elocity Aided **E**stimation (DIVE), avoids the complexity of stochastic cloning present in other IMU-only learned inertial approaches [4], [7] by learning the absolute velocity element of the extended pose, rather than learning a relative pose constraint. The proposed algorithm outperforms the SOTA in inertial-only odometry for multirotors, considered TLIO [4] herein. Improved performance is seen on an in-distribution testing set and out-of-distribution testing set, as well as outperforming standard dead reckoning in VIO failure scenarios.

II. Related Work

VIO is a well-explored topic in the literature, and is the de-facto standard for multirotor navigation in GPS-denied environments [1]. While extremely information-rich, aggressive multirotor flight can cause VIO to fail due to motion blur and textureless environments. In moments of failure, the navigation algorithm relies on its proprioceptive data, which is provided by the IMU. However, the standard practice during periods of VIO failure without any egomotion constraints or tracked features (before a successful reinitialization) is to propagate the state forward using the IMU kinematics in a dead-reckoned fashion, which allows for unconstrained 6DOF motion and does not leverage the multirotor's kinematics and dynamic constraints.

Learned inertial odometry has shown significant progress in pedestrian and ground-vehicle navigation. Pedestrian dead reckoning (PDR) has seen significant focus due to its strong motion prior, where algorithms such as RoNIN [8] regress velocity from gravity-aligned IMU measurements for position estimation in 2D via concatenation. Meanwhile, RIDI [9] uses a regressed velocity to correct a window of linear accelerations for position estimation via double integration. TLIO [4] exceeds the accuracy of both algorithms using a 1D residual CNN that regresses a relative-position constraint in the gravity-aligned frame, which is then applied as a correction to an EKF.

RINS-W [5] and AI-IMU [10] both use deep learning to leverage the dynamic constraints of a ground-vehicle. RINS-W proposes training a long short-term memory network (LSTM) motion-profile classifier that is used to select between different corrections that represent the common dynamic constraints of a ground-vehicle, as well as integrating a zero velocity detector, further described in [11]. AI-IMU Dead Reckoning, in a similar spirit, forms the dynamic constraints into a regression problem using a CNN to learn the covariance on a zero lateral and upwards velocity pseudomeasurement.

DIDO [6] and [12] use tachometers in addition to an IMU for multirotor state estimation. DIDO uses motor speeds and IMU measurements as inputs to a large recurrent network model, where residual CNNs are leveraged to learn IMU biases and a correction on the quadrotor thrust model, and gated recurrent units (GRUs) are used to learn integration error. IMO [7] uses collective thrust and gyroscope histories as inputs to a temporal convolutional network (TCN) to regress a relative-position constraint, but has only been proven on drone-racing paths and cannot generalize to unseen trajectories. This is due to the fact that the network output is a displacement in the world-frame, meaning that the TCN learns a set of inertial paths, and any deviation from these paths will result in a poor estimate. Both DIDO and IMO present TLIO as the inertial SOTA, showing that TLIO is generalizable to multirotor flight despite being formulated for PDR.

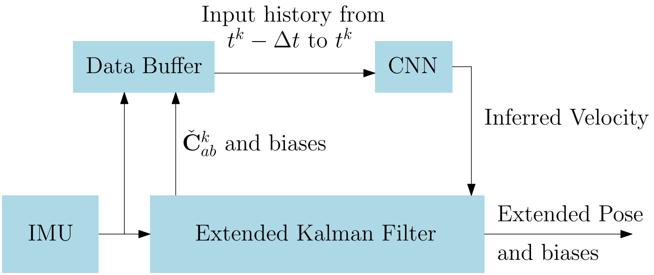


Fig. 2. Block diagram of the overall system. The CNN takes as input the history of orientations and linear accelerations provided by the data buffer and outputs an inferred velocity, which is used to update an EKF.

III. Methodology

There are two primary components of the proposed algorithm: a CNN-based learning model, and an EKF that fuses the output of the CNN with a 6DOF process model for extended pose and bias estimation. A block diagram of the overall system is shown in Fig. 2.

The learning model is a residual CNN [13]. The network inputs are a history of orientations and linear accelerations in the past Δt seconds from some time t^k. These inputs are generated from the IMU measurements and the EKF's current orientation and bias estimates. The network outputs an inferred velocity $\tilde{\mathbf{v}}_g^k$ at time t^k, and the learned uncertainty $\tilde{\mathbf{\Sigma}}_v^k$. The inferred velocity $\tilde{\mathbf{v}}_g^k$ and uncertainty $\tilde{\mathbf{\Sigma}}_v^k$, being treated as an uncertain measurement, are used to correct an EKF prediction, where the EKF prediction is generated using the IMU data. The CNN has no initial velocity prior, and is made to infer a motion prior from the multirotor's kinematics and dynamics. The EKF then outputs an extended pose estimate, which includes the 3D position, velocity, and orientation, as well as the gyroscope and accelerometer bias estimates.

A. Reference Frames

The frame $\mathcal{F}_a$ represents the inertial frame, the frame $\mathcal{F}_b$ represents the body frame, and the frame $\mathcal{F}_g$ represents the gravity-aligned frame. The frame $\mathcal{F}_g$ is constructed by rotating the inertial frame about the gravity direction by the same yaw angle as the body frame. $\mathbf{C}_{ab} \in SO(3)$ represents the direction cosine matrix (DCM) that transforms $\mathcal{F}_b$ to $\mathcal{F}_a$. Given one physical vector resolved in either $\mathcal{F}_a$ or $\mathcal{F}_b$, the relationship between the components is $\mathbf{r}_a = \mathbf{C}_{ab}\mathbf{r}_b$. The DCM $\mathbf{C}_{ab}$ can also be written as

$$\mathbf{C}_{ab} = \begin{bmatrix}\mathbf{b}_a^1 & \mathbf{b}_a^2 & \mathbf{b}_a^3\end{bmatrix},$$

where

$$\mathbf{b}_a^i = \begin{bmatrix}b_{ax}^i & b_{ay}^i & b_{az}^i\end{bmatrix}^\mathsf{T}, \quad i \in \{1,2,3\}$$

are the basis vectors defining $\mathcal{F}_b$, resolved in $\mathcal{F}_a$. Similarly, $\mathbf{C}_{ag}$ represents the transformation from $\mathcal{F}_g$ to $\mathcal{F}_a$, and is represented by

$$\mathbf{C}_{ag} = \begin{bmatrix}\mathbf{g}_a^1 & \mathbf{g}_a^2 & \mathbf{g}_a^3\end{bmatrix}.$$

The gravity-aligned frame $\mathcal{F}_g$ is defined such that its first axis is $\mathbf{b}_a^1$ projected onto the horizontal plane, its third axis is colinear with the gravity vector, and the second axis completes the triad. Mathematically,

$$\mathbf{g}_a^1 = \frac{\begin{bmatrix}b_{ax}^1 & b_{ay}^1 & 0\end{bmatrix}^\mathsf{T}}{\left\|\begin{bmatrix}b_{ax}^1 & b_{ay}^1 & 0\end{bmatrix}^\mathsf{T}\right\|_2},$$

$$\mathbf{g}_a^3 = \begin{bmatrix}0 & 0 & 1\end{bmatrix}^\mathsf{T}, \quad \mathbf{g}_a^2 = \left(\mathbf{g}_a^3\right)^\times \mathbf{g}_a^1,$$

Alternatively, $\mathbf{C}_{ag}$ can be obtained from the yaw angle γ with

$$\gamma = \arccos\left(\frac{b_{ax}^1}{\sqrt{\left(b_{ax}^1\right)^2 + \left(b_{ay}^1\right)^2}}\right) = \operatorname{arctan2}(b_{ay}^1, b_{ax}^1),$$

$$\mathbf{C}_{ag} = \exp\left(\begin{bmatrix}0 & 0 & \gamma\end{bmatrix}^{\mathsf{T}^\times}\right) = \mathbf{C}_3(\gamma), \tag{1}$$

where $\exp(\cdot)$ is the matrix exponential, $(\cdot)^\times$ describes the skew-symmetric cross product matrix, and $\mathbf{C}_3(\cdot)$ denotes a principal DCM about the third axis.

In the remainder of this letter, prior and posterior estimated quantities are denoted by $(\check{\cdot})$ and $(\hat{\cdot})$, respectively. Measured quantities are denoted by $(\tilde{\cdot})$.

B. Residual CNN Model

The residual CNN model is a residual network that takes a history of orientations in the form of rotation vectors and linear accelerations as six 1-dimensional channels. The input then proceeds through multiple convolutional and fully-connected layers in a lightweight form of ResNet18 [13].

The desired orientation input is a history of estimated DCMs $\hat{\mathbf{C}}_{gb}^i$ that describe the transformations from the body frame to the gravity-aligned frame, where i corresponds to the indicies between $t^k - \Delta t$ and t^k. The inertial window length Δt is a hyperparameter chosen during training to balance between temporal correlation and available information, as further discussed in Section V-B. Given an orientation estimate $\hat{\mathbf{C}}_{ab}^k$ at time t^k, $\hat{\mathbf{C}}_{ag}^k$ can be retrieved from (1), and $\hat{\mathbf{C}}_{gb}^k = \hat{\mathbf{C}}_{ag}^{k^\mathsf{T}} \hat{\mathbf{C}}_{ab}^k$. During training, $\hat{\mathbf{C}}_{ab}^k$ is supplied by ground-truth, but in deployment, it is supplied by the EKF. The desired orientation history can be computed by propagating Poisson's equation backwards using the gyroscope measurements, denoted by $\tilde{\boldsymbol{\omega}}_b^k$. Letting τ be $t^k - t^{k-1}$, $\hat{\mathbf{C}}_{ab}^{k-1}$ is then computed by

$$\hat{\mathbf{C}}_{ab}^{k-1} = \hat{\mathbf{C}}_{ab}^k \exp\left(\tau \tilde{\boldsymbol{\omega}}_b^{k-1^\times}\right)^\mathsf{T}.$$

$\hat{\mathbf{C}}_{gb}^{k-1}$ is then retrieved as

$$\hat{\mathbf{C}}_{gb}^{k-1} = \hat{\mathbf{C}}_{ag}^{k^\mathsf{T}} \hat{\mathbf{C}}_{ab}^{k-1}, \tag{2}$$

and this can be repeated until the entire rotational history from $t^k - \Delta t$ to t^k is obtained. Note that in (2), $\hat{\mathbf{C}}_{ag}^k$ is fixed at time t^k since the gravity-aligned frame $\mathcal{F}_g$ at the most recent timestep is chosen as the anchor frame to maintain yaw unobservability, as done in [4]. It is also found that this choice of anchor frame helps in learning the dynamics, and reduces the size of the output space. The accelerometer measurements are transformed into linear accelerations resolved in the gravity-aligned frame $\mathcal{F}_g$. Let $\tilde{\boldsymbol{\alpha}}_b^k$ denote the accelerometer measurement at time t^k and $\hat{\mathbf{C}}_{gb}^k$ denote the DCM at time t^k computed using (2). The linear acceleration input is then computed as

$$\mathbf{a}_g^k = \hat{\mathbf{C}}_{gb}^k \tilde{\boldsymbol{\alpha}}_b^k - \boldsymbol{\gamma}_g,$$

where $\boldsymbol{\gamma}_g = \boldsymbol{\gamma}_a$, and $\boldsymbol{\gamma}_a$ is the gravity vector resolved in $\mathcal{F}_a$. The corresponding rotation vectors are then generated from the orientation history by $\hat{\boldsymbol{\phi}}_{gb}^k = \log(\hat{\mathbf{C}}_{gb}^k)^\vee$, where $(\boldsymbol{\phi}^\times)^\vee = \boldsymbol{\phi}$ and $\log(\cdot)$ is the matrix logarithm. The vectors $\mathbf{a}_g^k$ and $\hat{\boldsymbol{\phi}}_{gb}^k$ are then concatenated to form the input at time t^k, and this process is repeated to generate a full input history from $t^k - \Delta t$ to t^k. The network then outputs an inferred velocity $\tilde{\mathbf{v}}_g^k$. Instead of using perfect data during training, ground-truth body-frame angular velocity and proper accelerations are augmented with white noise, axis misalignment, and a small bias. Doing so is intended to robustify the model to noise and errors in bias estimation, and to encourage the learning to be IMU-agnostic and avoid learning device-specific bias evolution. At runtime, the IMU samples are interpolated to the target frequency, and then proceed through the gravity-alignment to form the network input.

Following the convolutional layers, the network contains two separate fully-connected layers that output $\tilde{\mathbf{v}}_g^k$, the inferred velocity, and $\tilde{\boldsymbol{\Sigma}}_v^k$, the covariance. The network is trained until convergence with mean-squared-error (MSE) loss,

$$\mathcal{L}_{\text{MSE}}(\mathbf{v}_g^k, \tilde{\mathbf{v}}_g^k) = \frac{1}{N}\sum_{k=1}^{N}\left\|\mathbf{v}_g^k - \tilde{\mathbf{v}}_g^k\right\|^2,$$

where $\mathbf{v}_g^k$ is the ground-truth velocity in the gravity-aligned frame. Upon convergence, training continues with the negative-log-likelihood (NLL) loss,

$$\mathcal{L}_{\text{NLL}}(\mathbf{v}_g^k, \tilde{\mathbf{v}}_g^k, \tilde{\boldsymbol{\Sigma}}_v^k)$$
$$= \frac{1}{N}\sum_{k=1}^{N}\left(\frac{1}{2}\left(\mathbf{v}_g^k - \tilde{\mathbf{v}}_g^k\right)^\mathsf{T} \tilde{\boldsymbol{\Sigma}}_v^{k^{-1}} \left(\mathbf{v}_g^k - \tilde{\mathbf{v}}_g^k\right) + \frac{1}{2}\ln|\tilde{\boldsymbol{\Sigma}}_v^k|\right),$$

in order to converge to the uncertainty present in the data, as suggested in [14]. The network is therefore trained to output an associated covariance $\tilde{\boldsymbol{\Sigma}}_v^k = \operatorname{diag}(\tilde{\boldsymbol{\sigma}}_v^k)$, where $\tilde{\boldsymbol{\sigma}}_v^k \in \mathbb{R}^3$, which is used in the estimation loop. This two-step learning process is motivated by the fact that training with NLL does not converge if not preceded by MSE.

C. Inertial Model Odometry

1) Filter Propagation: An on-manifold EKF is implemented that operates on the $SE_2(3)$ Lie group, described in [15]. The vehicle extended pose is

$$\mathbf{T} = \begin{bmatrix}\mathbf{C} & \mathbf{v} & \mathbf{r}\\ \mathbf{0} & 1 & 0\\ \mathbf{0} & 0 & 1\end{bmatrix} \in SE_2(3),$$

where $\mathbf{C} \in SO(3)$ and $\mathbf{v}, \mathbf{r} \in \mathbb{R}^3$.

The full filter state is represented by $\hat{\mathcal{X}} = (\hat{\mathbf{T}}, \hat{\boldsymbol{\beta}}_g, \hat{\boldsymbol{\beta}}_a) \in SE_2(3) \times \mathbb{R}^6$, where $\hat{\mathbf{T}}$ is the estimated extended pose, $\hat{\boldsymbol{\beta}}_g$ is the estimated gyroscope bias, and $\hat{\boldsymbol{\beta}}_a$ is the estimated accelerometer bias. The error state is then defined as $\delta\boldsymbol{\mathcal{X}} = [\delta\boldsymbol{\xi}^\mathsf{T} \quad \delta\boldsymbol{\beta}_g^\mathsf{T} \quad \delta\boldsymbol{\beta}_a^\mathsf{T}]^\mathsf{T} \in \mathbb{R}^{15}$.

The generic continuous-time inertial navigation equations are used to propagate the filter's state, which are

$$\dot{\mathbf{C}}_{ab} = \mathbf{C}_{ab}(t)\left(\tilde{\boldsymbol{\omega}}_b(t)\right)^\times, \tag{3}$$

$$\dot{\mathbf{v}}_a = \mathbf{C}_{ab}(t)\tilde{\boldsymbol{\alpha}}_b(t) + \boldsymbol{\gamma}_a, \tag{4}$$

$$\dot{\mathbf{r}}_a(t) = \mathbf{v}_a. \tag{5}$$

(3)–(5) can also be written in compact form as

$$\dot{\mathbf{T}} = \mathbf{G}\mathbf{T} + \mathbf{T}\mathbf{U}, \tag{6}$$

and assuming that the IMU measurements $\tilde{\boldsymbol{\omega}}_b(t), \tilde{\boldsymbol{\alpha}}_b(t)$, without bias and noise, are constant over a small integration interval τ, (6) can be discretized as in [16, Section 9.4.7] to yield

$$\mathbf{T}^k = \exp\left(\tau\mathbf{G}\right)\mathbf{T}^{k-1}\exp\left(\tau\mathbf{U}\right) = \mathbf{G}^{k-1}\mathbf{T}^{k-1}\mathbf{U}^{k-1}. \tag{7}$$

The discrete-time bias evolution is

$$\boldsymbol{\beta}_g^k = \boldsymbol{\beta}_g^{k-1}, \quad \boldsymbol{\beta}_a^k = \boldsymbol{\beta}_a^{k-1}. \tag{8}$$

The complete linearized discrete-time process model is then written as

$$\delta\check{\boldsymbol{\mathcal{X}}}^k = \mathbf{F}^{k-1}\delta\hat{\boldsymbol{\mathcal{X}}}^{k-1} + \mathbf{G}^{k-1}\delta\mathbf{w}^{k-1}, \tag{9}$$

where $\mathbf{w}^{k-1} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}^{k-1})$, $\mathbf{Q}^{k-1}$ is the noise covariance determined by the intrinsic IMU noise parameters, and $\mathbf{F}^{k-1}$ and $\mathbf{G}^{k-1}$ are the state and noise jacobians, respectively. The state covariance is then propagated by

$$\check{\mathbf{P}}^k = \mathbf{F}^{k-1}\hat{\mathbf{P}}^{k-1}\mathbf{F}^{k-1^\mathsf{T}} + \mathbf{G}^{k-1}\mathbf{Q}^{k-1}\mathbf{G}^{k-1^\mathsf{T}}. \tag{10}$$

2) Filter Update: The output of the network is defined as the velocity resolved in the local gravity-aligned frame, denoted as $\tilde{\mathbf{v}}_g^k$. Treating the inferred velocity $\tilde{\mathbf{v}}_g^k$ as a measurement, the measurement function can then be written as

$$\tilde{\mathbf{v}}_g^k = \mathbf{h}\left(\mathcal{X}^k, \boldsymbol{\eta}_v^k\right) = \mathbf{C}_{ag}^{k^\mathsf{T}}\mathbf{v}_a^k + \boldsymbol{\eta}_v^k, \tag{11}$$

where $\boldsymbol{\eta}_v^k$ is assumed to be normally distributed according to $\mathcal{N}(\mathbf{0}, \tilde{\boldsymbol{\Sigma}}_v^k)$, and $\tilde{\boldsymbol{\Sigma}}_v^k$ is the associated covariance output by the network. The measurement Jacobian with respect to the state can then be written as $\mathbf{H}^k$, and the state update equations are

$$\mathbf{K}^k = \check{\mathbf{P}}^k\mathbf{H}^{k^\mathsf{T}}\left(\mathbf{H}^k\check{\mathbf{P}}^k\mathbf{H}^{k^\mathsf{T}} + \tilde{\boldsymbol{\Sigma}}_v^k\right)^{-1}, \tag{12}$$

$$\delta\hat{\boldsymbol{\mathcal{X}}}^k = \mathbf{K}^k\left(\tilde{\mathbf{v}}_g^k - \mathbf{h}\left(\check{\mathcal{X}}^k\right)\right), \tag{13}$$

$$\delta\hat{\boldsymbol{\mathcal{X}}}^k = \left[\delta\hat{\boldsymbol{\xi}}^{k^\mathsf{T}} \quad \delta\hat{\boldsymbol{\beta}}_g^{k^\mathsf{T}} \quad \delta\hat{\boldsymbol{\beta}}_a^{k^\mathsf{T}}\right]^\mathsf{T},$$

$$\check{\mathcal{X}}^k = (\check{\mathbf{T}}^k, \check{\boldsymbol{\beta}}_g^k, \check{\boldsymbol{\beta}}_a^k),$$

$$\hat{\mathbf{T}}^k = \exp(\delta\hat{\boldsymbol{\xi}}^{k^\wedge})\check{\mathbf{T}}^k, \tag{14}$$

$$\hat{\boldsymbol{\beta}}_g^k = \check{\boldsymbol{\beta}}_g^k + \delta\hat{\boldsymbol{\beta}}_g^k, \quad \hat{\boldsymbol{\beta}}_a^k = \check{\boldsymbol{\beta}}_a^k + \delta\hat{\boldsymbol{\beta}}_a^k, \tag{15}$$

$$\hat{\mathbf{P}}^k = (\mathbf{1} - \mathbf{K}^k\mathbf{H}^k)\check{\mathbf{P}}^k, \tag{16}$$

where $\exp((\cdot)^\wedge)$ defines the mapping from $\mathbb{R}^9 \to SE_2(3)$, and $(\cdot)^\wedge$ is the wedge operator for $SE_2(3)$ as defined in [16, Section 9.2.4]. In practice, the network uncertainty $\tilde{\boldsymbol{\Sigma}}_v^k$ is inflated to compensate for unmodelled temporal cross correlations, which is shown to be effective in [17]. This is further discussed in Section V-B.

D. Training and Data Specifications

The neural network is trained on a PC with an Intel Core i9-12900 K processor with 16 cores, and a NVIDIA GeForce RTX3070 graphics processing unit. The bias applied to an inertial window of ground-truth IMU measurements for training is generated from a uniform distribution of $[-1^{-2}, 1^{-2}]$ $\frac{rad}{s}$ and $[-5^{-2}, 5^{-2}]$ $\frac{m}{s^2}$. The axis misalignment norm is generated from a uniform distribution of $[0, 5]$ deg. The white noise parameters for an inertial window are generated from a uniform distribution of $[6^{-3}, 2^{-2}]$ $\frac{m}{s^2}\frac{1}{\sqrt{Hz}}$ and $[1^{-3}, 2^{-3}]$ $\frac{rad}{s}\frac{1}{\sqrt{Hz}}$. The network is trained with $\mathcal{L}_{\text{MSE}}$ for 10 epochs, and then switched to $\mathcal{L}_{\text{NLL}}$ for convergence, which takes around another 5 epochs at a learning rate of 10^{-4}. Training takes about one hour when using approximately 3 hours of flight data collected at 400 Hz. DIVE has approximately 8 million trainable parameters, and an inference time of 5.48 ms on an NVIDIA GeForce GTX 1650 with Max-Q. For comparison, TLIO has approximately 6 million trainable parameters, and an inference time of 5 ms. Note that, for all presented experiments, the network update rate is 10 Hz.

IV. Experiments

Three sets of experiments are run to validate the proposed algorithm. The first two use only the proposed algorithm to navigate a set of trajectories with a near-perfect state initialization, and contain both an in-sample testing set that is split from the same dataset as the training data, and a generalization test, which contains a completely unseen IMU and quadrotor. The third experiment is a VIO failure test that compares the proposed algorithm to standard dead reckoning given a near-perfect bias initialization at the beginning of a failure period.

A. Datasets

The training and in-sample testing sets are taken from the DIDO dataset [6]. DIDO is dynamically diverse, long-form, and contains high-frequency ground truth, which makes it suitable for training. The proposed methodology of synthesizing ground-truth IMU measurements and augmenting with noise and bias helps prevent overfitting to IMU intrinsics, but the dynamic mapping of the network is still dependent on mass and physical structure. DIDO assembled a custom drone for data acquisition.

The Blackbird and EuRoC datasets are chosen to test cross-dataset generalization. The Blackbird dataset is chosen as it has highly-varying velocity and high-frequency ground truth [18]. Blackbird also assembles a custom drone for data acquisition, and has an IMU frequency of 100 Hz, as compared to the 400 Hz IMU frequency of DIDO, allowing for validation that the proposed algorithm functions even with upsampled data and a different physical system. The EuRoC dataset is chosen as it provides a complete batched ground truth, allowing for testing with an accurate initial bias estimate. Additionally, the faster trajectories are dynamic enough to induce motion blur and test estimation under VIO failure cases. EuRoC uses the AscTec

Firefly as it's flight platform, and has an IMU frequency of 200 Hz [19].

B. Algorithm Performance

Having introduced the testing datasets, the performance of the proposed algorithm is compared to TLIO [4], the learned inertial SOTA. An overview of TLIO is given in Section II.

1) Accuracy Study: The proposed algorithm and TLIO are trained on a subset of trajectories from the DIDO dataset that are chosen for their diversity of movement. The testing trajectories are conceptually similar but unseen to the algorithm at training time. A $75/15/10$ split is used for testing/training/validation, and there are 127 training trajectories, 26 validation trajectories, and 12 testing trajectories. Evaluating the relative error is useful as it is less time-sensitive than absolute error [20], and provides more information about the localized drift, which is more in the spirit of inertial navigation. Instead of absolute error, relative pose error (RPE) is used, which is a measure of the localized drift of the estimate over a particular time period $\Delta\tau$, as presented in [20], [21]. For some period $\Delta\tau$, which represents some state interval Δ, the RPE is computed as

$$\mathbf{T}^{\Delta} = \left(\mathbf{T}^{k^{-1}}\mathbf{T}^{k+\Delta}\right)^{-1}\left(\hat{\mathbf{T}}^{k^{-1}}\hat{\mathbf{T}}^{k+\Delta}\right), \tag{17}$$

where $\mathbf{T}^k$ is the ground-truth pose at timestep k, and $\hat{\mathbf{T}}^k$ is the estimated extended pose at timestep k. The individual error values are then retrieved from the components of $\mathbf{T}^{\Delta}$ as

$$\text{RPE}_{\text{trans}} = \frac{1}{n-\Delta}\sum_{i=1}^{n-\Delta}\left\|\mathbf{r}_{\Delta}^{i}\right\|^2, \tag{18}$$

$$\text{RPE}_{\text{vel}} = \frac{1}{n-\Delta}\sum_{i=1}^{n-\Delta}\left\|\mathbf{v}_{\Delta}^{i}\right\|^2, \tag{19}$$

$$\text{RPE}_{\text{rot}} = \frac{1}{n-\Delta}\sum_{i=1}^{n-\Delta}\left\|\log\left(\mathbf{C}_{\Delta}^{i}\right)^{\vee}\right\|^2. \tag{20}$$

In the results presented, $\Delta\tau$ is chosen to be 2 seconds.

Both TLIO and the proposed algorithm are tested on DIDO with a highly uncertain initial bias estimate of zero, and an otherwise perfect state initialization. The results are shown in Fig. 3, and an example of a testing trajectory is shown in Fig. 5(a). The proposed algorithm shows a 42% improvement in $\text{RPE}_{\text{trans}}$ and a 62% improvement in RPE_{vel} relative to TLIO. As direct integration of the gyroscope provides an accurate rotational estimate without additional information, the RPE_{rot} is similarly low for both. Overall, DIVE shows significantly improved performance over TLIO in both translational drift and velocity estimation, indicating that the velocity constraint is a superior learned mapping to the relative position constraint for multirotors.

2) Generalization Study: The Blackbird and EuRoC trajectories represent the cross-dataset generalization test for the proposed algorithm. All Blackbird trajectories provide a 3-second stationary period prior to flight for initialization of gyroscope bias. Both TLIO and the proposed algorithm are initialized with a highly uncertain zero accelerometer bias estimate, and a near-perfect gyroscope bias estimate. EuRoC provides a complete batched ground truth, so both TLIO and the proposed algorithm are initialized with an accurate extended pose and bias estimate. The results are shown in Fig. 4. The proposed algorithm outperforms TLIO by 22% and 18% in $\text{RPE}_{\text{trans}}$ on Blackbird and EuRoC, respectively, and the RPE_{rot} is similarly low for both. The results of the study show that DIVE is able to generalize well to various physical systems with upsampled data from a different IMU, indicating that the velocity regression mapping can be applied to both hexrotors and quadrotors if learned on a different flight platform. Lastly, DIVE outperforms TLIO in translational accuracy on both datasets, indicating that the velocity constraint maintains its superiority in the generalized case.

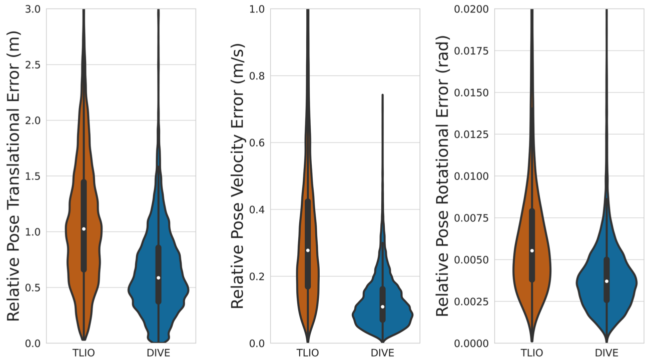


Fig. 3. Relative Pose Errors on DIDO testing trajectories by TLIO and DIVE. The width of the envelope shows the relative frequency of the data. The median is represented as a white dot, and the first and third percentile of the data are represented by the thick black bar.

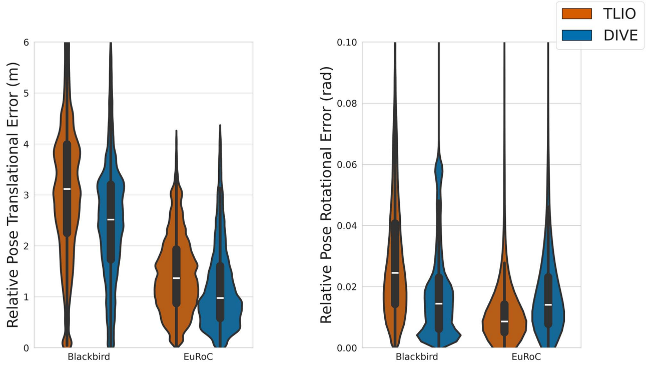
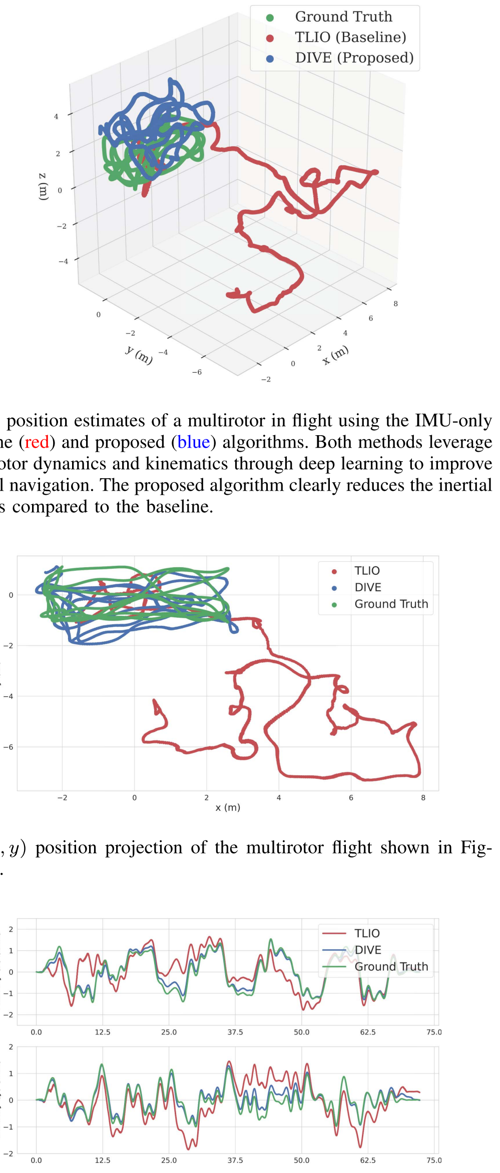


Fig. 4. Relative Pose Errors on Blackbird and EuRoC trajectories by TLIO and DIVE.

3) VIO Failure Study: In the preceding sections, it is demonstrated that the proposed algorithm outperforms the chosen learned inertial SOTA, TLIO, in relative pose error in both in-distribution and out-of-distribution tests. However, it is also useful to consider the proposed algorithm's application as a replacement for the inertial component of a VIO system upon failure. The failure case that will be considered for this study is an immediate VIO failure where no exteroceptive corrections are applied to the system for some failure period. Generally, when this occurs, the proprioceptive backup is standard dead reckoning, which is the baseline to which the proposed algorithm is compared. It is assumed that, as VIO contains a complementary set of sensors, that the sensor bias estimates are accurate at the beginning of the failure period. The DIDO testing trajectories are used for this study, where noisy and biased IMU measurements are synthesized from ground-truth body-frame angular velocity and proper acceleration in a similar method to Section III-D. The



(a) 3D position estimates of a multirotor in flight using the IMU-only baseline (red) and proposed (blue) algorithms. Both methods leverage multirotor dynamics and kinematics through deep learning to improve inertial navigation. The proposed algorithm clearly reduces the inertial drift as compared to the baseline.

(b) (x, y) position projection of the multirotor flight shown in Figure 5a.

(c) (x, y) velocity estimates of the multirotor flight shown in Figure 5a.

Fig. 5. Visualization of a DIDO flight trajectory as estimated by TLIO and DIVE.

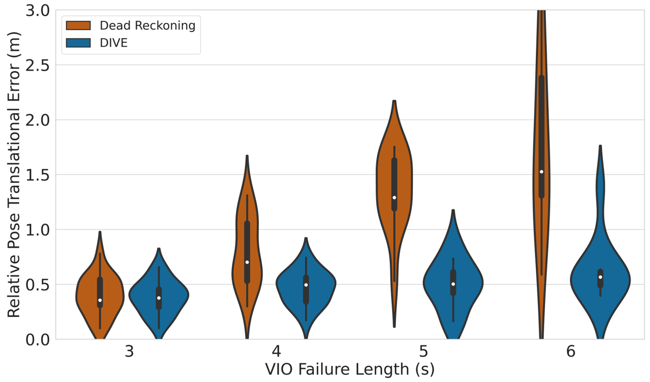


Fig. 6. Translational drift during VIO failure periods by DIVE and standard dead reckoning.

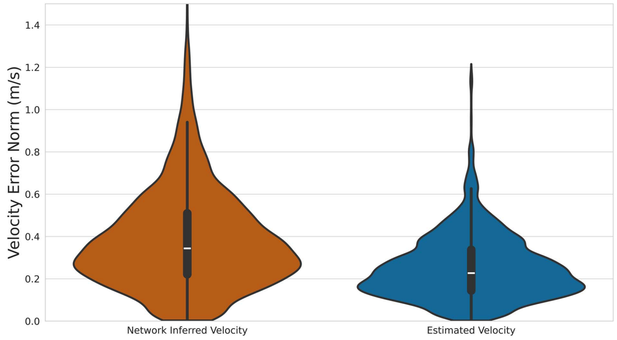


Fig. 7. Velocity error norms from both the yaw-corrected network inferred velocity $\bar{\mathbf{v}}_a$ and the yaw-corrected a posteriori velocity $\hat{\mathbf{v}}_a$.

results are shown in Fig. 6. The proposed algorithm demonstrates a $3\%, 42\%, 63\%$ and 65% improvement in absolute position drift over standard dead reckoning for $3, 4, 5$ and 6 s failure period lengths, respectively. A visualization of the baseline vs proposed performance during a VIO failure is shown in Fig. 1.

V. Network Studies

A. EKF Ablation Study

In order to validate the contribution of the EKF, the accuracy of the network inferred velocity and the a posteriori velocity estimates are compared. This was done by applying a yaw-correction to both the inferred velocity in the local gravity-aligned frame $\bar{\mathbf{v}}_g$, and the a posteriori velocity in the estimated body frame $\hat{\mathbf{v}}_b$, and then comparing the error to the ground-truth velocity. The results are shown in Fig. 7. The a posteriori velocity shows a 34% improvement in velocity error over the inferred velocity, demonstrating the importance of the information fusion provided by the EKF in the estimation loop. The improvement likely comes from the high accuracy of the standard IMU propogation model in the short-term and and more consistent state estimates due to it's fusion with the inferred velocity using its learned uncertainty.

B. Window Length Study

The noise and bias augmentation applied in Section III-D prevents the proposed algorithm from overfitting to the noise intrinsics of the IMU used in the training and validation datasets. However, overfitting to the nature of the training trajectories is still possible. Learning velocity from shorter input windows is more generalizable as there are less possible distinct trajectories. At one extreme, learning to regress velocity from a single IMU sample would be maximally generalizable as the motion is simple and commonly occuring. At the other extreme, learning to regress velocity from a 10 s input window is minimally generalizable as that trajectory will not occur often, and will likely cause the network to overfit to that specific motion.

A hyperparameter study on the input window length, represented by Δt in Section III, is conducted to show that the algorithm is not overfitting to the training trajectories. In this study, the CNN is trained on different input window lengths of $\{0.5, 1, 1.5, 2.5, 3.5\}$ seconds. Then, the CNN is put in the

TABLE I
MEAN RELATIVE POSE TRANSLATIONAL ERROR (M) ACHIEVED BY DIVE ON DIDO AND BLACKBIRD AS A FUNCTION OF THE INPUT WINDOW LENGTH

Window Length (s)	$\overline{\text{RPE}}_{\text{trans}}$ (m)	
	Datasets	
	DIDO	Blackbird
0.5	0.63	2.62
1	0.64	**2.49**
1.5	0.60	7.29
2.5	**0.51**	3.76
3.5	1.25	5.84

Best values are bolded.

estimation loop with the EKF to measure its performance. The results are shown in Table 1. Overall, as the algorithm's input window length increases, its performance on the DIDO dataset (used for training) improves, but its performance on the Blackbird dataset degrades. As the input window grows even larger, performance on both falls significantly below the baseline. This shows that there is some degree of overfitting to the nature of the trajectory itself, as the model learns to take advantage of the additional information provided by the longer input window length, but is unable to apply that same learning to the Blackbird dataset, as the flight patterns are inherently different. Additionally, at longer window lengths, the temporal correlation of measurements increases, and without a way to scale the covariance accordingly, this likely contributes to the degradation in performance. Therefore, in order to train a generalizable model, it is important to choose a shorter input window length. However, shortening the window length too much can result in a lack of data by which to regress an inferred velocity. To balance this, a window length of 1 s is chosen for the experiments presented in this work.

VI. DISCUSSION AND CONCLUSION

In this letter, the problem of learned inertial estimation for quadrotors has been addressed by proposing a novel algorithm, DIVE. DIVE regresses a velocity estimate from a history of IMU-derived inputs and applies it as a correction to an EKF formulated on $SE_2(3)$. DIVE is then evaluated against TLIO on the DIDO, Blackbird, and EuRoC datasets, and shows significant improvement in localization accuracy on all, indicating the superiority of the inferred velocity mapping to the relative position constraint for both in-dataset accuracy and generalizability. Additionally, DIVE is evaluated against dead reckoning in VIO failure scenarios, showing significantly less translational drift.

This work poses a deep learning architecture that is limited in accuracy by the amount of data that it is trained on. Given the generalizability of the proposed model, it is believed that by open-sourcing the source code and training data, DIVE can be continually improved by training on additional quadrotors and trajectories.

REFERENCES

[1] G. Huang, "Visual-inertial navigation: A concise review," in *Proc. Int. Conf. Robot. Automat.*, 2019, pp. 9572–9582.

[2] D. V. Nam and K. Gon-Woo, "Solid-state LiDAR based-SLAM: A concise review and application," in *Proc. IEEE Int. Conf. Big Data Smart Comput.*, 2021, pp. 302–305.

[3] T. Qin, S. Cao, J. Pan, and S. Shen, "A general optimization-based framework for global pose estimation with multiple sensors," 2019, *arXiv:1901.03642*.

[4] W. Liu et al., "TLIO: Tight learned inertial odometry," *IEEE Robot. Automat. Lett.*, vol. 5, no. 4, pp. 5653–5660, Oct. 2020.

[5] M. Brossard, A. Barrau, and S. Bonnabel, "RINS-W: Robust inertial navigation system on wheels," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 2068–2075.

[6] K. Zhang et al., "DIDO: Deep inertial quadrotor dynamical odometry," *IEEE Robot. Automat. Lett.*, vol. 7, no. 4, pp. 9083–9090, Oct. 2022.

[7] G. Cioffi, L. Bauersfeld, E. Kaufmann, and D. Scaramuzza, "Learned inertial odometry for autonomous drone racing," *IEEE Robot. Automat. Lett.*, vol. 8, no. 5, pp. 2684–2691, May 2023.

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

[9] H. Yan, Q. Shan, and Y. Furukawa, "RIDI: Robust IMU double integration," in *Proc. ECCV 15th Eur. Conf.*, 2018, pp. 621–636.

[10] M. Brossard, A. Barrau, and S. Bonnabel, "Ai-IMU dead-reckoning," *IEEE Trans. Intell. Veh.*, vol. 5, no. 4, pp. 585–595, Dec. 2020.

[11] A. Ramanandan, A. Chen, and J. A. Farrell, "Inertial navigation aiding by stationary updates," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 1, pp. 235–248, Mar. 2012.

[12] J. Svacha, J. Paulos, G. Loianno, and V. Kumar, "IMU-Based inertia estimation for a quadrotor using Newton-Euler dynamics," *IEEE Robot. Automat. Lett.*, vol. 5, no. 3, pp. 3861–3867, Jul. 2020.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[14] R. L. Russell and C. Reale, "Multivariate uncertainty in deep learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 12, pp. 7937–7943, Dec. 2022.

[15] A. Barrau and S. Bonnabel, "The invariant extended Kalman filter as a stable observer," *IEEE Trans. Autom. Control*, vol. 62, no. 4, pp. 1797–1812, Apr. 2017.

[16] T. D. Barfoot, *State Estimation for Robotics*, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 2022.

[17] S. Julier and J. Uhlmann, "A non-divergent estimation algorithm in the presence of unknown correlations," in *Proc. Amer. Control Conf.*, 1997, vol. 4, pp. 2369–2373.

[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 *Proc. Springer Proc. Adv. Robot.*, 2020, vol. 11, pp. 130–139.

[19] M. Burri et al., "The EuRoC micro aerial vehicle datasets," *Int. J. Robot. Res.*, vol. 35, no. 10, pp. 1157–1163, Sep. 2016.

[20] Z. Zhang and D. Scaramuzza, "A tutorial on quantitative trajectory evaluation for Visual(-Inertial) odometry," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 7244–7251.

[21] D. Prokhorov, D. Zhukov, O. Barinova, K. Anton, and A. Vorontsova, "Measuring robustness of visual SLAM," in *Proc. 16th Int. Conf. Mach. Vis. Appl.*, 2019, pp. 1–6.