Deep IMU Bias Inference for Robust Visual-Inertial Odometry With Factor Graphs

Russell Buchanan, *Graduate Student Member, IEEE*, Varun Agrawal, Marco Camurri, *Member, IEEE*, Frank Dellaert, and Maurice Fallon, *Senior Member, IEEE*

***Abstract*—Visual Inertial Odometry (VIO) is one of the most established state estimation methods for mobile platforms. However, when visual tracking fails, VIO algorithms quickly diverge due to rapid error accumulation during inertial data integration. This error is typically modeled as a combination of additive Gaussian noise and a slowly changing bias which evolves as a random walk. In this work, we propose to train a neural network to learn the true bias evolution. We implement and compare two common sequential deep learning architectures: LSTMs and Transformers. Our approach follows from recent learning-based inertial estimators, but, instead of learning a motion model, we target IMU bias explicitly, which allows us to generalize to locomotion patterns unseen in training. We show that our proposed method improves state estimation in visually challenging situations across a wide range of motions by quadrupedal robots, walking humans, and drones. Our experiments show an average 15% reduction in drift rate, with much larger reductions when there is total vision failure. Importantly, we also demonstrate that models trained with one locomotion pattern (human walking) can be applied to another (quadruped robot trotting) without retraining.**

***Index Terms*—Visual-inertial SLAM, sensor fusion, deep learning methods.**

I. Introduction

State estimation for lightweight, mobile systems is a fundamental problem in robotics. Visual Inertial Odometry (VIO) is a common solution due to the small size and low cost of cameras and IMUs. The main weak-point of VIO is that when visual feature tracking fails, only the Inertial Measurement Unit (IMU) can be used.

Manuscript received 13 July 2022; accepted 3 November 2022. Date of publication 17 November 2022; date of current version 28 November 2022. This letter was recommended for publication by Associate Editor F. Pomerleau and Editor S. Behnke upon evaluation of the reviewers' comments. This work was supported in part by the EPSRC ORCA Robotics Hub under Grant EP/R026173/1, in part by EU H2020 Project THING under Grant 780883, and in part by Royal Society University Research Fellowship (Fallon). For the purpose of Open Access, the author has applied a CC BY public copyright licence to any Author Accepted Manuscript version arising from this submission. *(Corresponding author: Russell Buchanan.)*

Russell Buchanan, Marco Camurri, and Maurice Fallon are with the Oxford Robotics Institute, University of Oxford, OX1 2JD Oxford, U.K. (e-mail: russell@robots.ox.ac.uk; mcamurri@robots.ox.ac.uk; mfallon@robots.ox.ac.uk).

Varun Agrawal and Frank Dellaert are with the Institute for Robotics and Intelligent Machines, College of Computing, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: varunagrawal@gatech.edu; dellaert@cc.gatech.edu).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2022.3222956, provided by the authors.

Digital Object Identifier 10.1109/LRA.2022.3222956

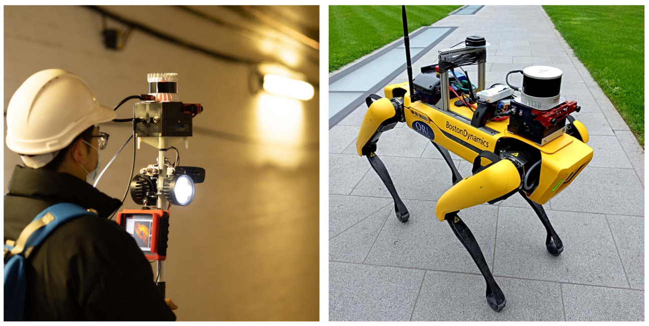

Fig. 1. Two of the platforms used in this work. Left: handheld device including a SevenSense AlphaSense visual-inertial sensor. Right: the same device mounted on a Boston Dynamics Spot quadruped. An IMU bias model from a neural network trained on handheld data was used to infer the biases on the Spot dataset without any retraining — despite the significantly different motion pattern. The lidar was not used in these experiments.

Inexpensive and miniaturized micro-electromechanical systems (MEMS) IMUs have become ubiquitous in robotics and pedestrian tracking [1], [2], [3], [4], however IMU-only state estimation is severely affected by drift. This drift is the result of the accumulation of various errors in IMU integration collectively modeled as a combination of additive zero-mean Gaussian noise and a slowly changing bias. As a result, estimation which relies purely on IMU measurement integration is not feasible for more than a few seconds due to explosive accumulation of drift.

VIO is effective so long as visual features are available because they constrain the drift of IMU integration [5] and estimate the biases. When visual tracking fails completely, the system relies purely on IMU integration. When there is error in visual tracking, the estimator may update the bias estimates to make sense of the data. In other words, the estimated bias is not necessarily related to the underlying physical process, but a quantity that minimizes the residuals. If there were another way to infer the IMU biases, these problems could be mitigated.

Most VIO works have focused on improving visual tracking, however in this work we take a novel approach of *improving IMU bias modeling*. We propose a new method which uses deep learning to estimate IMU biases directly from IMU measurements and past biases. We train a neural network to learn the evolution process of the biases of a specific IMU rather than assuming Brownian Motion. As a result, our method is *device specific, not locomotion specific*, unlike similar IMU learning approaches, and does not require periodic motion. Our contributions can be listed as follows:

- A neural network capable of estimating IMU biases from a history of measurements and biases. To the best of the

authors' knowledge, this is the first method capable of explicitly inferring the bias evolution of an IMU.
- A performance comparison of two different network implementations (LSTM and Transformer) and their integration as unary factors into a state estimator based on factor graphs, for improved estimation in visually challenging scenarios.
- Real-world experiments on three different platforms with different types of motions: pedestrian handheld, quadrupedal robot and drones. To the best of the authors' knowledge, this is the first work that demonstrates an IMU learned model trained on one locomotion modality and tested on another (handheld to quadrupedal).

An additional minor contribution is the development of a ROS compatible, open-source tool for calibrating an IMU using the Allan Variance method.[1]

II. Related Work

In this section, we summarize the growing field of inertial learning from which our work follows. In Section II-A, we discuss methods which learn motion models from inertial data, while Section II-B covers methods which learn IMU noise models primarily for drone state estimation.

A. Learning Inertial Motion Models

Recent works have trained neural networks with IMU data to learn motion models (typically of pedestrians) and to output estimates of velocity directly from IMU measurements.

The first data driven Inertial Navigation System (INS) was IoNet by Chen et al. [6] which inferred 2D displacement and orientation change from buffered IMU data. Later, Herath et al. [7] proposed RoNIN, a similar network which inferred 2D velocity and orientation from raw IMU data. Liu et al. [8] proposed TLIO, a method to infer 3D motion and estimate high-fidelity trajectories in a filtering framework. They used an Extended Kalman Filter (EKF) to estimate the full 6 DoF state. Process updates were performed by traditional IMU mechanization while measurement updates from the network were used as relative position measurements. This allowed the filter to implicitly estimate IMU biases.

In our prior work [9], we adapted the approach of [8] to incorporate additional sensors such as cameras, lidar or legged robot kinematics in a factor graph. However, this method, like the ones discussed above, was based on learning a motion model and was therefore susceptible to failure if applied outside of the training domain. For example, in Fig. 2 we show results of TLIO which was trained with handheld walking data on flat ground. As shown in Fig. 2(a)), the velocity estimates are reasonable for IMU-only odometry. In Fig. 2(b)), we applied the same network to stair climbing, which was not present in the training set and as a result, the z velocity estimation fails completely. In Fig. 2(c)) we tested the network with a quadruped and the model fails completely. This motivates the need for methods which are more IMU specific rather than locomotion specific.

B. Learning IMU Noise Models

A different approach is to learn the IMU noise model. In this case, the IMU measurements are passed through a network

[1][Online]. Available: https://github.com/ori-drs/allan_variance_ros

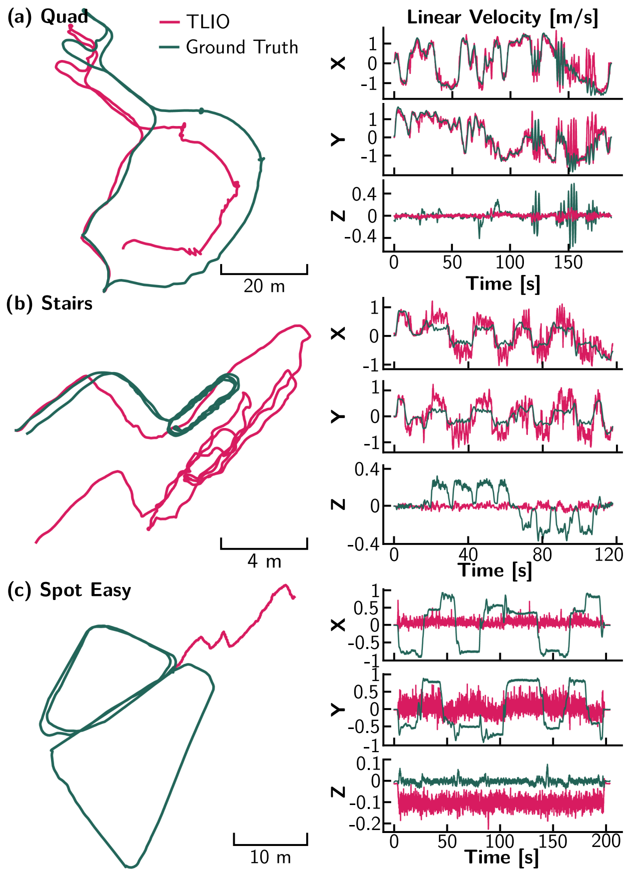

Fig. 2. Top: TLIO [8] trained on handheld data on flat ground, tested on flat ground. Middle: same network tested on a sequence in which the person walks up and down stairs. Because stair climbing was not present in the training set, inference fails significantly (Note z velocity estimates). Bottom: when applied to a quadrupedal robot, because of the completely different locomotion modality, TLIO is unable to estimate velocity.

trained to "denoise" them and output an estimate of the perfect IMU measurements. These denoised signals can then be used directly as input to a VIO pipeline.

Brossard et al. [10] used a Convolutional Neural Network (CNN) to denoise a gyroscope using the output to correct for the true angular velocity. The CNN used dilated convolutions to increase the temporal coverage over longer sequences. Zhang et al. [11] trained a Recurrent Neural Network (RNN) to denoise both gyroscope and accelerometer measurements. Their network used IMU measurements as input to estimate a corrected measurement which, when integrated, reduced pose error. This eliminated the need for the model to learn the underlying mechanization equations. Similarly, Steinbrener et al. [12] performed denoising on IMU measurements, comparing LSTM and Transformer architectures, finding the LSTM to be more effective.

The main drawback of these methods is that they do not distinguish between different noise sources and, as a result, it is not clear what the network has learned to remove from the IMU measurements. For example, most results were primarily demonstrated on drones which, when flying, introduce high frequency vibrations affecting the IMU measurements. It is unclear if the noise characteristics being estimated were caused by the vibrations, the internal state of the IMU (i.e. the true bias), the motion of the drone, or a combination of the above.

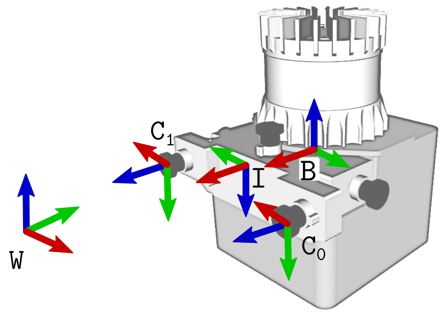

Fig. 3. Reference frames convention for the handheld device. The world frame W is a fixed frame, while the base B, camera optical C and IMU I frames are attached to the moving device.

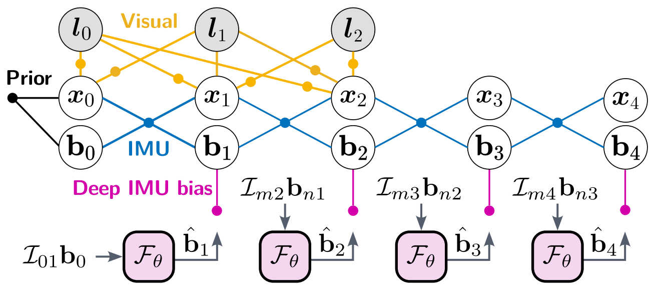
Visual
Prior
IMU
Deep IMU bias
l_0 l_1 l_2
x_0 x_1 x_2 x_3 x_4
b_0 b_1 b_2 b_3 b_4
$\mathcal{I}_{m2}\mathbf{b}_{n1}$ $\mathcal{I}_{m3}\mathbf{b}_{n2}$ $\mathcal{I}_{m4}\mathbf{b}_{n3}$
$\mathcal{I}_{01}\mathbf{b}_0$
$\mathcal{F}_\theta$
$\hat{\mathbf{b}}_1$ $\hat{\mathbf{b}}_2$ $\hat{\mathbf{b}}_3$ $\hat{\mathbf{b}}_4$

Fig. 4. Proposed factor graph framework with learned IMU estimates. A neural network $\mathcal{F}_\theta$ trained on ground truth IMU biases takes a window of past IMU measurements $\mathcal{I}_{mj}$ and previous bias estimates $\mathbf{b}_{ni}$ from the optimizer and outputs the current bias estimate $\hat{\mathbf{b}}_j$. This is integrated into the factor graph as unary factor (magenta).

By explicitly modeling the IMU bias, in this letter we seek a motion independent method with more explainable outputs.

III. Problem Statement

We aim to estimate the trajectory of a mobile platform equipped with an IMU and a stereo camera using sliding window based smoothing. The relevant reference frames of the platform are shown in Fig. 3 and includes an Earth-fixed world frame W, the platform-fixed base frame B, left and right camera frames $\mathtt{C}_0, \mathtt{C}_1$, and IMU sensor frame I.

A. Definition of State Vector and Measurements

The state of the platform at time t_i is defined as:

$$\boldsymbol{x}_i \triangleq [\mathbf{R}_i, \mathbf{p}_i, \mathbf{v}_i, \mathbf{b}_i^g \;\; \mathbf{b}_i^a] \in \mathrm{SO}(3) \times \mathbb{R}^{12} \tag{1}$$

where: $\mathbf{R}_i = \mathbf{R}_{\mathtt{WB}}(t_i)$ is the orientation of B with respect to W, $\mathbf{p}_i = {}_{\mathtt{W}}\mathbf{p}_{\mathtt{WB}}(t_i)$ is the position, $\mathbf{v}_i = {}_{\mathtt{B}}\mathbf{v}_{\mathtt{WB}}(t_i)$ is the linear velocity, and $\mathbf{b}_i^g = {}_{\mathtt{I}}\mathbf{b}^g(t_i)$, $\mathbf{b}_i^a = {}_{\mathtt{I}}\mathbf{b}^a(t_i)$ are the IMU gyroscope and accelerometer biases, respectively.

We indicate the set of all states in the window as $\mathcal{X}_k = \{\boldsymbol{x}_i\}_{i \in \mathsf{K}_k}$ where K_k are all the keyframe indices up to t_k. Similarly, the measurements within the window are $\mathcal{Z}_k = \{\mathcal{I}_{ij}, \mathcal{C}_i\}_{i,j} \in \mathsf{K}_k$, where $\mathcal{I}_{ij}$ are the IMU measurements between two camera keyframes (with $i = j - 1$), while $\mathcal{C}_i$ include the stereo image pairs at time t_i.

B. Maximum-a-Posteriori (MAP) Estimation

We maximize the likelihood of the measurements $\mathcal{Z}_k$, given the history of states $\mathcal{X}_k$:

$$\mathcal{X}_k^* = \underset{\mathcal{X}_k}{\arg\max}\, p(\mathcal{X}_k|\mathcal{Z}_k) \propto p(\mathcal{X}_0)p(\mathcal{Z}_k|\mathcal{X}_k) \tag{2}$$

The measurements are formulated as conditionally independent and corrupted by white Gaussian noise. Therefore, (2) can be expressed as the following least squares minimization:

$$\mathcal{X}_k^* = \underset{\mathcal{X}_k}{\arg\min} \|\mathbf{r}_0\|^2_{\Sigma_0} + \sum_{i \in \mathsf{K}_k} \Bigg(\|\mathbf{r}_{\mathcal{I}_{ij}}\|^2_{\Sigma_{\mathcal{I}_{ij}}} + \sum_{\ell \in \mathsf{M}_i} \|\mathbf{r}_{\boldsymbol{x}_i,\boldsymbol{m}_\ell}\|^2_{\Sigma_{\boldsymbol{x}_i,\boldsymbol{m}_\ell}} + \|\mathbf{r}_{\mathbf{b}_j^a}\|^2_{\hat{\Sigma}_{\mathbf{b}^a}} + \|\mathbf{r}_{\mathbf{b}_j^g}\|^2_{\hat{\Sigma}_{\mathbf{b}^g}} \Bigg) \tag{3}$$

where each term is the residual associated to a measurement type, weighted by the inverse of its covariance matrix, and will be detailed in Section IV.

IV. Factor Graph Formulation

A factor graph can be used to graphically represent (3) as in Fig. 4. With slight abuse of notation, the IMU biases are shown separately from the state nodes $\boldsymbol{x}_i$ to highlight our contribution. In addition to the prior factors (black), the graph includes: IMU preintegration (blue), visual tracking (yellow) and our novel deep IMU bias (magenta) factors. For convenience, the first two are briefly reported in this section, while the last one is detailed in Section V.

A. Preintegrated IMU Factors

We follow the standard manner of IMU measurement integration from [5] to constrain the pose, velocity, and biases between two consecutive nodes of the graph. The residual has the following form:

$$\mathbf{r}_{\mathcal{I}_{ij}} = \left[\mathbf{r}^{\mathsf{T}}_{\Delta\mathbf{R}_{ij}}, \mathbf{r}^{\mathsf{T}}_{\Delta\mathbf{v}_{ij}}, \mathbf{r}^{\mathsf{T}}_{\Delta\mathbf{p}_{ij}}, \mathbf{r}^{\mathsf{T}}_{\mathbf{b}^a_{ij}}, \mathbf{r}^{\mathsf{T}}_{\mathbf{b}^g_{ij}}\right]^{\mathsf{T}} \tag{4}$$

For a detailed definition of the above residuals, see [5].

B. Stereo Landmark Factors

A visual landmark in Euclidean space $\boldsymbol{m}_\ell \in \mathbb{R}^3$ is projected onto the image plane by the function $\pi : \mathrm{SE}(3) \times \mathbb{R}^3 \mapsto \mathbb{R}^2$, given the platform pose $\mathbf{T}_i = \{\mathbf{p}_i, \mathbf{R}_i\} \in \mathrm{SE}(3)$. Given a landmark $\boldsymbol{m}_\ell$ and its coordinates $(u_\ell, v_\ell) \in \mathbb{R}^2$ on the image plane, the residual at state is computed as:

$$\mathbf{r}_{\boldsymbol{x}_i,\boldsymbol{m}_\ell} = \begin{pmatrix} \pi^L_u(\mathbf{T}_i, \boldsymbol{m}_\ell) - u^L_{i,\ell} \\ \pi^R_u(\mathbf{T}_i, \boldsymbol{m}_\ell) - u^R_{i,\ell} \\ \pi_v(\mathbf{T}_i, \boldsymbol{m}_\ell) - v_{i,\ell} \end{pmatrix} \tag{5}$$

where $(u^L, v), (u^R, v)$ are the pixel locations of the landmark. $\Sigma_{\boldsymbol{m}}$ is computed using an uncertainty of 0.25px with the Dynamic Covariance Scaling robust loss function as in [13].

V. Deep IMU Bias Factors

A MEMS IMU measures its proper acceleration (e.g., equal to Earth's gravity at rest) $\tilde{\mathbf{a}} \in \mathbb{R}^3$ and rotational rate $\tilde{\boldsymbol{\omega}} \in \mathbb{R}^3$. The absolute linear acceleration $\mathbf{a}$ (e.g., null at rest) and the rotation rate $\boldsymbol{\omega}$ of a body expressed in an Earth-fixed coordinate frame $\mathtt{W}$ can be recovered as follows:

$$\begin{aligned}\tilde{\mathbf{a}}(t) &= \mathbf{R}_{\mathtt{WI}}^{\mathsf{T}}(t)(\mathbf{a}(t) - {}_{\mathtt{W}}\mathbf{g}) + \mathbf{b}^a(t) + \boldsymbol{\eta}^a(t)\\ \tilde{\boldsymbol{\omega}}(t) &= \boldsymbol{\omega}(t) + \mathbf{b}^g(t) + \boldsymbol{\eta}^g(t) \end{aligned} \tag{6}$$

Where $\mathbf{R}_{\mathtt{WI}} \in \mathrm{SO}(3)$ is the absolute orientation of the IMU and ${}_{\mathtt{W}}\mathbf{g}$ is the acceleration due to gravity expressed in $\mathtt{W}$. The quantities $\boldsymbol{\eta}^a$ and $\boldsymbol{\eta}^g$ represent additive noise present in all IMUs and is modeled as zero-mean Gaussian distributions.

A. Bias Noise Model

The IMU biases $\mathbf{b}^a$ and $\mathbf{b}^g$ are due to physical properties of the IMU. They change with each power cycle and continue to change slowly during operation [14]. Typically, their evolution is modeled as a Brownian noise process, whose derivatives are sampled from zero-mean Gaussian distributions:

$$\dot{\mathbf{b}}^a(t) = \boldsymbol{\eta}^{ba} \qquad \dot{\mathbf{b}}^g(t) = \boldsymbol{\eta}^{bg} \tag{7}$$

This can be re-written in discrete time by integrating between two time steps $[t_i, t_j]$:

$$\mathbf{b}^a_j = \mathbf{b}^a_i + \boldsymbol{\eta}^{bad} \qquad \mathbf{b}^g_j = \mathbf{b}^g_i + \boldsymbol{\eta}^{bgd} \tag{8}$$

Where $\boldsymbol{\eta}^{bad}$ and $\boldsymbol{\eta}^{bgd}$ are discrete zero mean Gaussian distributions with covariance $\boldsymbol{\Sigma}^{bad}$ and $\boldsymbol{\Sigma}^{bgd}$ respectively [5].

B. Learning the Bias Process Model

The Brownian noise model for IMU biases in (8) is an approximation which does not hold for long periods of time. In practice, the true underlying dynamics are a highly nonlinear function depending on vibrations, impacts, and physical properties of the device [14]. Thus, modeling the IMU dynamics is a problem that lends itself to deep learning as it allows us to approximate highly nonlinear functions. To this end, instead of (8), we propose a deep neural network $\mathcal{F}$ with parameters θ defined as follows:

$$\mathcal{F}_\theta : (\mathcal{I}_{mj}, \mathbf{b}^a_{ni}, \mathbf{b}^g_{ni}) \mapsto \left(\hat{\mathbf{b}}^a_j, \hat{\mathbf{b}}^g_j\right) \tag{9}$$

where the inputs are a buffer of IMU measurements $\mathcal{I}_{mj}$ between times t_m and t_j, and the previous bias values $\mathbf{b}^a_{ni}, \mathbf{b}^g_{ni}$ between t_n and t_i. The network outputs $(\hat{\mathbf{b}}^a_j, \hat{\mathbf{b}}^g_j)$ are the estimates of the IMU bias value at time t_j. Without loss of generality, in this section we assume the output of the network is generated at the camera keyframe rate.

C. Deep IMU Bias Factors

We incorporate the bias estimates from the network into a factor graph-based state estimator. The estimates are modeled as unary factors on the bias state, as shown in Fig. 4. The last two residuals of (3), $\mathbf{r}_{\mathbf{b}^a_j}$ and $\mathbf{r}_{\mathbf{b}^g_j}$ correspond to:

$$\mathbf{r}_{\mathbf{b}^a_j} = \mathbf{b}^a_j - \hat{\mathbf{b}}^a_j \qquad \mathbf{r}_{\mathbf{b}^g_j} = \mathbf{b}^g_j - \hat{\mathbf{b}}^g_j \tag{10}$$

The covariances $\hat{\boldsymbol{\Sigma}}_{\mathbf{b}^a}$ and $\hat{\boldsymbol{\Sigma}}_{\mathbf{b}^g}$ were tuned to fixed values for our experiments, with $\hat{\boldsymbol{\Sigma}}_{\mathbf{b}^a} = \mathbf{I}_3 \cdot 2.5\mathrm{e}^{-3}$ and $\hat{\boldsymbol{\Sigma}}_{\mathbf{b}^g} = \mathbf{I}_3 \cdot 2.5\mathrm{e}^{-5}$. In future work, we intend to train the networks to provide measurement uncertainty estimates as in [8], [9].

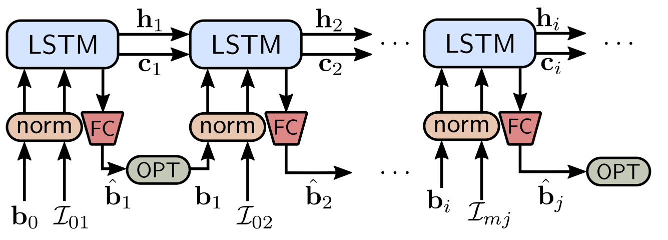

Fig. 5. LSTM architecture. A IMU data window $\mathcal{I}_{mj}$ of size w (with $m = j - w$) and the previous bias $\mathbf{b}_i$ are first normalized then passed to the LSTM. The hidden state is preserved for the next inference step and the output is passed through a fully connected layer to predict a bias.

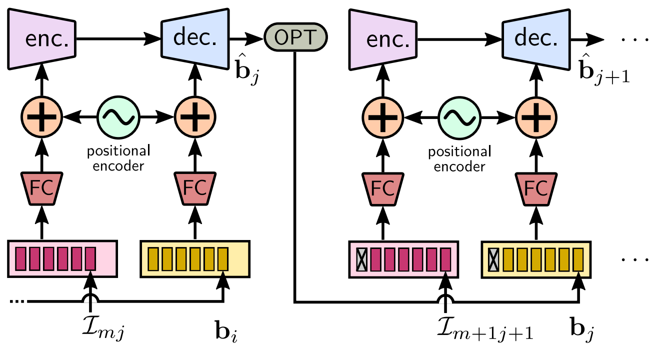

Fig. 6. Transformer architecture. A history of l IMU windows $\mathcal{I}_{mj}$ and biases $\mathbf{b}_i$ are concatenated and summed with a positional encoding before being passed to the Transformer.

VI. Network Architecture

Since our learning task involves sequential data, two common network architectures are well suited when implementing (9). The first one is based on Long Short-Term Memory (LSTM) [15] which is lightweight (see Fig. 5) and the second is based on Transformers [16] which have been shown to have better performance for longer-term sequences (see Fig. 6).

A. LSTM

The input to the LSTM is a single window of size w of IMU measurements $\mathcal{I}_{mj}$ (with $m = j - w$) and the previous bias estimate $\mathbf{b}_i$ (i.e., $t_n = t_i$), which comes from the factor graph optimizer. This exposes the network to bias estimates resulting from the fusion of additional sensors. These are normalized then passed to the LSTM with states $\mathbf{h}_i$ and $\mathbf{c}_i$ which are preserved for the next bias estimate. In this way, the LSTM receives batches of w IMU measurements while the memory can observe the bias evolution over time. One limitation to this method is that, over a long trajectory, the LSTM will eventually forget information from old inputs.

B. Transformer

The Transformer input is a history of l windows of IMU measurements and biases (i.e., $m = j - l \cdot w, n = l$). Similar to the LSTM, biases added to the history come from the estimator's optimizer. A history of information allows the Transformer attention mechanism to recall older information.

TABLE I
DATASETS USED FOR EXPERIMENTS

Dataset	Duration [s]	Length [m]	IMU	Locom.
NCD-Multi	3163	4512	BMI085*	Handheld
Spot	515	399	BMI085*	Quadruped
EuRoC	1349	893	ADIS16448	Drone

*Same IMU device.

C. Loss Function

We use the Mean Square Error (MSE) as a loss function:

$$\mathcal{L}(\mathbf{b}, \hat{\mathbf{b}}) = \frac{1}{n} \sum_{k=1}^{n} \|\mathbf{b}_k - \hat{\mathbf{b}}_k\|^2 \tag{11}$$

where two separate instances of the network are trained with (11) for accelerometer and gyroscope biases, respectively.

VII. Network Implementation

A. Datasets

The datasets used for training and experiments are listed in Table I. The first is the Newer College Multi-Camera Dataset (NCD-Multi) [17] which uses an AlphaSense inertial and multi-camera sensor (see Fig. 1), although we only use the front facing stereo cameras. NCD-Multi was collected by a human operator walking while carrying the device inside New College, Oxford. An additional non-public sequence was collected using the same device in a limestone mine [18].

The second dataset, Spot, was recorded to demonstrate generalization to motions patterns unseen in training. We placed the same sensor payload used in NCD-Multi on a quadruped robot (see Fig. 1, right) and recorded two sequences only for testing.

Finally, we present results with the public EuRoC dataset [19]. We included this dataset to show results alongside similar methods and to demonstrate application to a very different platform: a drone.

B. Training

We trained one LSTM and one Transformer each for the public datasets (NCD-Multi and EuRoC), for a total of four models. For each dataset, we selected hard sequences for testing (see VIII-A) and the remaining were split 75:25 for training and validation. The Spot dataset was only used for testing because, as detailed in Section VIII, we demonstrate generalization to different locomotion modalities by training a model on NCD-Multi and testing it on the Spot dataset.

For training, we used teacher forcing [20], providing the network with ground truth biases. The ground truth biases for NCD-Multi were estimated using the lidar and a high resolution 3D model of the environment while the EuRoC dataset provides IMU bias estimates. We added noise to the ground truth biases which was sampled from a zero-mean Gaussian distribution with covariances $\mathbf{\Sigma}^{bad}$ and $\mathbf{\Sigma}^{bgd}$ estimated from Allan Variance analysis.

As in [8], we rotate the IMU measurement windows $\mathcal{I}_{mj}$ into a gravity aligned frame. This prevents the effect of gravity from significantly changing the apparent accelerometer biases.

We used the Adam optimizer with a learning rate of 10^{-5} and batch size of 32. Training lasted 200 epochs for both LSTM and Transformer with the model minimizing validation error used. Both networks take ~ 2 h to train on a desktop computer with one Nvidia Titan X with 12 GB of memory.

C. Networks Models

1) LSTM: We used a 2-layer single direction LSTM with hidden state size of 256. We found that a larger network provided no improvement in performance while a lighter-weight LSTM is capable of running online with several inferences per second on a standard laptop with low-grade GPU. After testing different options, we found that 1 s of IMU data (i.e., $w = 10$ for 100 Hz IMU) and an inference rate of 2 Hz (50% data overlap between consecutive inferences) to give the best performance and used these settings for all experiments.

2) Transformer: We used an 8 headed Transformer with 2 encoder and decoder layers and an embedding size of 512. Similar to LSTM, we used 1 s IMU windows with a maximum history size of $l = 100$. This was based on analysis of the Allan Variance plots which found the IMU bias instability to dominate noise generally around 100 s sampling times. Inference was kept at 1 Hz since using a history of IMU windows makes overlapping of input data unnecessary.

VIII. Experimental Results

In this section, we describe the experiments and results of the proposed state estimator. We report relative position and orientation errors as described in [21]. We compare our method which uses bias predictions as in Fig. 4 to our baseline method, which consists of the same factor graph but without the unary factor described in Section IV.

A. Handheld Experiments

There are many common situations or environments in which visual odometry systems degrade or even fully fail such as door-opening, narrow spaces or fast rotations. We selected several sequences from NCD-Multi which exhibit these challenges to test our method. These sequences were only used for testing and contained situations unseen in training.

1) NCD-Multi Stairs: This sequence included narrow spaces, mirrors, door opening and large rotations as shown in Fig. 7. As a result, the baseline stereo visual odometry struggled and lost track of all features several times.

We show in Fig. 8 that the learned bias predictions reduced relative pose errors. This is most clearly seen in the magnified area where the door opening caused a significant drop in tracked visual features. Our proposed approach improves upon pure IMU-only mechanization in these situations where visual feature tracking fails.

2) NCD-Multi Mine: The Mine sequence was recorded in an abandoned limestone mine in Corsham, U.K.. It was dark and dusty and the handheld platform was intentionally held up very close to walls as shown in Fig. 7 on several occasions. This was also the longest trajectory.

The results can be seen in Fig. 9, where we highlight two particular sections in which the baseline VIO became unstable. In one section the camera faced a wall for several seconds and another involved large rotations in the dark. We see that by adding the bias prediction, trajectories are smoother and have lower error overall.

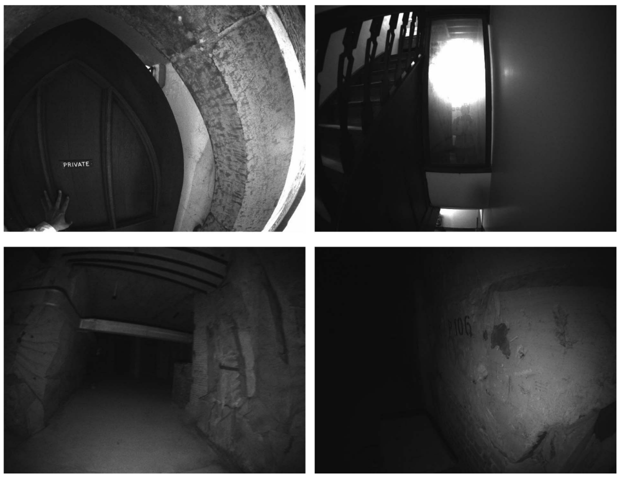

Fig. 7. Visual challenges in the NCD-Multi Dataset. Top Row: Images from NCD-Multi Stairs sequence with door opening (obscuring the cameras) shown on the left. On the right is a tight space about 1.5 m wide and a mirror. Bottom Row: Images from NCD-Multi Mine sequence with darkness and the challenge of camera being held up close to a wall.

TABLE II
10M RELATIVE TRANSLATION/ROTATION ERROR [M]/[°]

Experiment	VIO baseline	LSTM	Transformer
NCD-Multi Stairs*	0.23 / 3.22	0.19 / 2.88	**0.17** / **2.70**
NCD-Multi Mine	0.46 / 2.59	**0.33** / **1.71**	0.42 / 2.26
NCD-Multi Quad	0.35 / 1.38	0.28 / 1.36	**0.25** / **1.29**
Spot Easy	0.31 / 1.21	**0.27** / 1.02	0.31 / **0.88**
Spot Hard	0.38 / 1.57	0.34 /**1.42**	**0.30** / **1.42**

*5 m RPE due to shorter total distance traveled.

3) NCD-Multi Quad: This sequence is long and in a large-scale open space and includes several dynamic motions when the handheld platform was shaken. Our method reduces error compared to the baseline. A summary of these results and for the other NCD-Multi sequences is provided in Table II.

B. Quadruped Robot Experiments

For the Spot dataset, the same handheld sensor used in the NCD-Multi dataset was mounted on a Boston Dynamics Spot (See Fig. 1) which was then teleoperated around a courtyard in two separate trajectories. The Easy trajectory is entirely on flat ground in good lighting conditions while the Hard trajectory includes transitions from well-lit to shady areas and walking on small ramps.

For these experiments, both the LSTM and Transformer were trained *using only handheld data*. There was no fine-tuning for the quadrupedal robot. The information used for training and testing was therefore exactly the same as in Fig. 2. Numerical results are provided in Table II. This demonstrates that our method is independent from the locomotion modality and that the model can generalize across different modalities.

C. Drone Experiments

Two similar works to ours are Zhang et al. [11] and Brosard et al. [10]. Zhang et al. used an LSTM to denoise accelerometer and gyroscope measurements which are then passed to their visual-inertial pipeline based on [22]. Brossard et al. used a CNN to denoise gyroscope measurements only which are then passed to Open-VINS [3]. Since both works presented their results using the EuRoC dataset [19], we also use this dataset to demonstrate that our method can be applied to drones. Because the baselines were evaluated differently, drawing conclusions from direct comparisons is difficult. Therefore, we show relative improvement for each method.

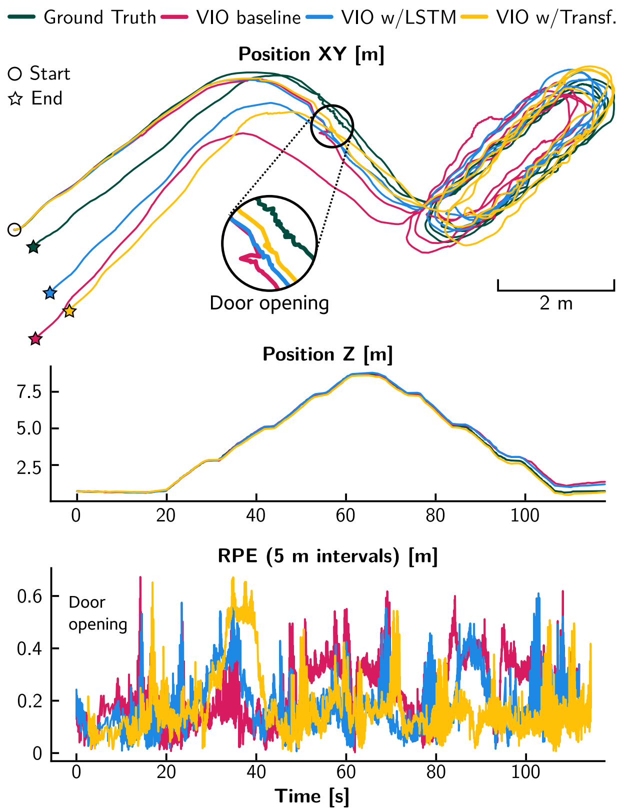

Fig. 8. NCD-Multi Stairs results. Top: Position of the baseline VIO, LSTM and Transformer methods on xy-plane. We magnify the door opening section of the trajectory where the baseline clearly diverges due to loss of visual feature tracking and poor bias estimation. Middle: Position of the trajectory for the z-axis. Bottom: 5 m RPE for each algorithm.

TABLE III
EUROC: ABSOLUTE POSITION ERROR & IMPROVEMENT [M (%)]

	Zhang et al. [11]		Ours		
Seq.	Baseline	Method	Baseline	LSTM	Transformer
MH02	0.19	0.15 (21%)	0.37	0.23 (38%)	0.13 (**65%**)
MH04	0.15	0.14 (7%)	0.33	0.25 (**24%**)	0.25 (**24%**)
V101	0.08	0.07 (13%)	0.17	0.08 (**53%**)	0.08 (**53%**)
V103	0.27	0.15 (**44%**)	0.27	0.27 (0%)	0.17 (37%)
V202	0.11	0.10 (9%)	0.29	0.23 (21%)	0.10 (**66%**)

TABLE IV
EUROC: ABSOLUTE ROTATION ERROR & IMPROVEMENT [DEG (%)]

	Brossard et al. [10]		Ours		
Seq.	Baseline	Method	Baseline	LSTM	Transformer
MH02	1.11	1.21 (-9%)	3.21	3.10 (3%)	2.86 (**11%**)
MH04	1.60	1.40 (13%)	0.89	1.00 (-12%)	0.76 (**15%**)
V101	0.80	0.80 (0%)	2.56	1.17 (**54%**)	1.66 (35%)
V103	2.32	2.25 (3%)	4.78	3.54 (26%)	1.87 (**61%**)
V202	1.85	1.81 (2%)	3.78	2.53 (33%)	1.31 (**65%**)

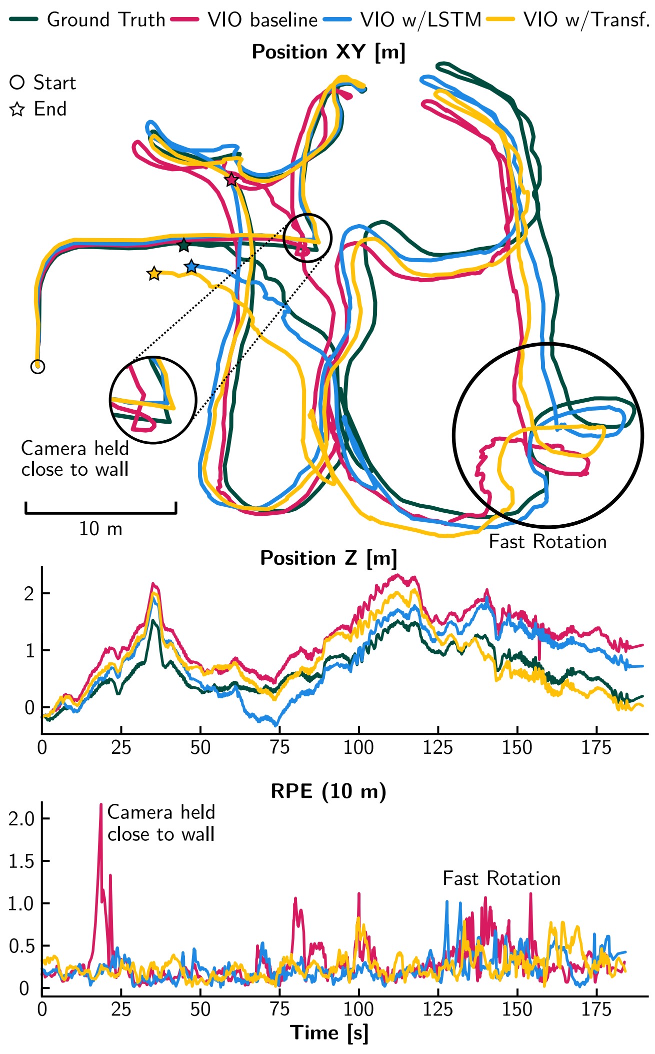

Fig. 9. Results from the Mine sequence. Top: Position on xy-plane of the baseline VIO, LSTM and Transformer methods. We magnify a part of the sequence where the camera was held facing and very close to a wall. The baseline diverges due to loss of visual feature tracking and poor bias estimation. We also highlight a section with large rotations where the baseline has multiple divergences. Middle: Position on z-axis. Bottom: 10 m RPE for each algorithm.

Tables III and IV summarize these results in terms of absolute error. Note that [11] reported positional RMSE while [10] reported orientation RMSE and therefore we report both. For the V101 sequence, we use the updated ground truth from [3] as did Brossard et al. [10].

Brossard et al. additionally reported relative orientation errors by averaging 7m, 21m and 32m relative orientation error for all trajectories. Averaged over the three distances they report 1.59° while we achieve 1.41° and 1.36° for LSTM and Transformer respectively.

D. Illustrative Experiments

We present two additional experiments to further analyze our system's performance. In both cases we used the Spot Easy trajectory and models trained with only handheld data.

1) Blackout: We "black out" the images for 5 s, setting all pixels to 0. As shown in Fig. 10 there is an error spike during the blackout where the estimator must rely on IMU data only.

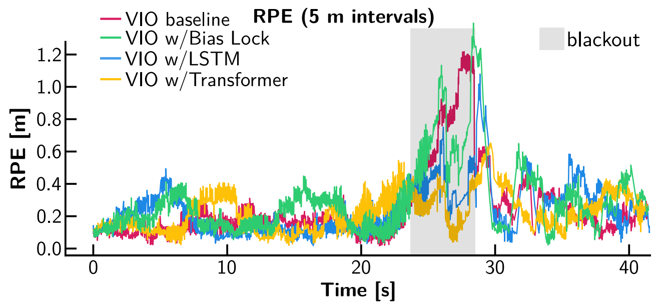

Fig. 10. Blackout Experiment: 5 m RPE Performance where the cameras were disabled for 5 s (gray area) during the Spot Easy trajectory. Mean 5 m RPE was 0.26 m, 0.28 m, 0.22 m and 0.20 m for the baseline, bias-lock, LSTM and Transformer methods respectively.

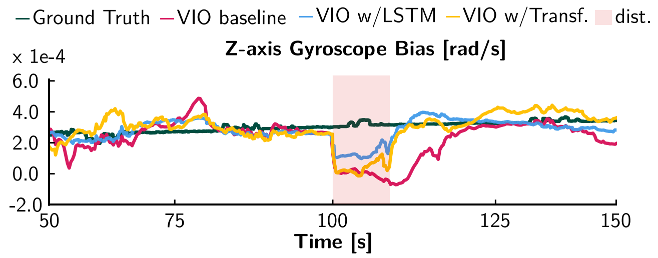

Fig. 11. Image Distortion Experiment: At 100 s of the Spot Easy trajectory we apply a small distortion to both stereo images (pink area). The baseline VIO develops a significant offset in z bias as the error is attributed to the gyroscope. LSTM and Transformer methods have reduced magnitude of error and recovery time.

We also experiment with "locking" the IMU biases during the blackout rather than allowing the random walk.

2) Image Distortion: When there are errors in visual tracking, such as poor camera calibration or wrong correspondences, the associated error can be erroneously attributed to the IMU bias. Learned bias estimates can mitigate this by providing estimates of the correct bias. To demonstrate this, we intentionally inject visual tracking error by left shifting both stereo images by 10 pixels for 10s. This is interpreted as an apparent rotation on the z-axis, attributed by the system to a change in yaw gyro bias. In Fig. 11 we plot the bias estimates during this time. Even though we initialize all methods with the ground truth bias and use a robust loss function, each method's estimate of the gyroscope z bias incorrectly drifts during the period of image distortion.

IX. Discussion

In Section VIII-A we show our methods achieving reduced error compared to the baseline in several visually challenging situations. On average, the LSTM and Transformer have reduced error in Table II by 15%, but in certain situations, for example the camera held close to the wall in the Mine trajectory, the improvement is over 300%. This is both because the learned estimates improve bias estimation and because they stabilize the optimizer in situations where error is incorrectly attributed to the IMU bias.

In Section VIII-B we take a model trained with handheld data and apply it to quadrupedal data to demonstrate our method is locomotion agnostic. Additionally, it should be noted the Stairs trajectory is the only handheld trajectory with stair climbing and therefore also demonstrates locomotion agnosticism. The reduced error of our method compared to the baseline shows our method is sensor specific and not locomotion specific.

In the Section VIII-C drone experiments, our method has similar accuracy as other methods. However, each method uses a different VIO baseline. Our method improves upon our baseline VIO more so than other methods improve on their baselines. Also, as the EuRoC dataset provides ground truth bias estimates, we examine the accelerometer and gyroscope bias $(\hat{\mathbf{b}}^a, \hat{\mathbf{b}}^g)$ estimation errors and find that, on average, error is reduced by (35%, 23%) for LSTM and (32%, 12%) for Transformer.

In the illustrative experiments in Section VIII-D we first present the Blackout experiment where our proposed methods reduced error during IMU-only estimation when no visual features could be tracked. Locking the bias did not improve on the baseline. This is because the bias random walk is already sufficiently constrained due to calibration. There is a second spike in error for all methods after vision is restored. This is because the pose estimate is suddenly corrected by a large amount, which locally appears as a relative error spike.

In the Image Distortion experiment, a bias shift affects all methods but the learned methods experience less error over the 10s and a faster recovery period. During distortion, the error of LSTM and Transformer methods are 41% and 12% lower (RMSE) than the baseline.

LSTM and Transformer provide similar improvements to positional error with Transformer slightly better at rotational error. This makes sense because the gyroscope biases change more slowly and we expect the Transformer to be more accurate with longer-term dependencies. In the distortion experiment the Transformer instead performs more poorly than the LSTM, which we believe is due to the LSTM operating at 2 Hz allowing it to react to disturbances more quickly. In terms of model size, the LSTM is superior with learned models as small as 30 MB compared to 200 MB for the transformer. Additionally, the LSTM forward pass takes less than 6 ms as compared to 20 ms for the Transformer.

X. Conclusion

We present a novel application of machine learning to inertial navigation which is more interpretable than similar methods. By learning bias predictions the proposed method is more generally applicable because it is not robot specific and does not require periodic locomotion which we demonstrate with a wide variety of experiments on quadrupeds, handheld sensors and drones. We show that our method reduces 5 m RPE of our baseline VIO system by 15% on average in handheld and quadrupedal experiments with RPE being reduced by as much as 300% in certain situations where vision fails. In the future, we will build on this work by integrating uncertainty estimation into the model and experimenting with non-patterned locomotion such as wheeled robots.

References

[1] D. Wisth, M. Camurri, S. Das, and M. Fallon, “Unified multi-modal landmark tracking for tightly coupled lidar-visual-inertial odometry,” *IEEE Robot. Automat. Lett.*, vol. 6, no. 2, pp. 1004–1011, Apr. 2021.

[2] S. Zhao, H. Zhang, P. Wang, L. Nogueira, and S. Scherer, “Super odometry: IMU-centric LiDAR-visual-inertial estimator for challenging environments,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 8729–8736.

[3] P. Geneva, K. Eckenhoff, W. Lee, Y. Yang, and G. Huang, “OpenVINS: A research platform for visual-inertial estimation,” in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 4666–4672.

[4] E. Foxlin, “Pedestrian tracking with shoe-mounted inertial sensors,” *IEEE Comput. Graph. Appl.*, vol. 25, no. 6, pp. 38–46, Nov./Dec. 2005.

[5] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, “On-manifold preintegration for real-time visual-inertial odometry,” *IEEE Trans. Robot.*, vol. 33, no. 1, pp. 1–21, Feb. 2017.

[6] C. Chen, X. Lu, A. Markham, and N. Trigoni, “IoNet: Learning to cure the curse of drift in inertial odometry,” in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 6468–6476.

[7] 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.

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

[9] R. Buchanan, M. Camurri, F. Dellaert, and M. Fallon, “Learning inertial odometry for dynamic legged robot state estimation,” in *Proc. Conf. Robot Learn.*, 2022, vol. 164, pp. 1575–1584.

[10] M. Brossard, S. Bonnabel, and A. Barrau, “Denoising IMU gyroscopes with deep learning for open-loop attitude estimation,” *IEEE Robot. Automat. Lett.*, vol. 5, no. 3, pp. 4796–4803, Jul. 2020.

[11] M. Zhang, M. Zhang, Y. Chen, and M. Li, “IMU data processing for inertial aided navigation: A recurrent neural network based approach,” in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 3992–3998.

[12] J. Steinbrener, C. Brommer, T. Jantos, A. Fornasier, and S. Weiss, “Improved state propagation through AI-based pre-processing and downsampling of high-speed inertial data,” in *Proc. IEEE Int. Conf. Robot. Automat.*, 2022, pp. 6084–6090.

[13] D. Wisth, M. Camurri, and M. Fallon, “Vilens: Visual, inertial, lidar, and leg odometry for all-terrain legged robots,” *IEEE Trans. Robot.*, 2022, doi: 10.1109/TRO.2022.3193788.

[14] D. Titterton and J. Weston, *Strapdown Inertial Navigation Technology*. London, U.K.: Inst. Eng. Technol., 2004.

[15] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[16] A. Vaswani et al., “Attention is all you need,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6000–6010.

[17] L. Zhang, M. Camurri, D. Wisth, and M. Fallon, “Multi-camera lidar inertial extension to the newer college dataset,” 2021. [Online]. Available: https://arxiv.org/abs/2112.08854

[18] L. Zhang, D. Wisth, M. Camurri, and M. Fallon, “Balancing the budget: Feature selection and tracking for multi-camera visual-inertial odometry,” *IEEE Robot. Automat. Lett.*, vol. 7, no. 2, pp. 1182–1189, Apr. 2022.

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

[20] R. J. Williams and D. Zipser, “A learning algorithm for continually running fully recurrent neural networks,” *Neural Comput.*, vol. 1, no. 2, pp. 270–280, 1989.

[21] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, “A benchmark for the evaluation of RGB-D SLAM systems,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2012, pp. 573–580.

[22] M. Zhang, X. Zuo, Y. Chen, Y. Liu, and M. Li, “Pose estimation for ground robots: On manifold representation, integration, reparameterization, and optimization,” *IEEE Trans. Robot.*, vol. 37, no. 4, pp. 1081–1099, Aug. 2021.