2021 IEEE International Conference on Robotics and Automation (ICRA 2021)
May 31 - June 4, 2021, Xi'an, China

IMU Data Processing For Inertial Aided Navigation: A Recurrent Neural Network Based Approach

Ming Zhang Mingming Zhang Yiming Chen Mingyang Li

Abstract— **In this work, we propose a novel method for performing inertial aided navigation, by using deep neural networks (DNNs). To date, most DNN inertial navigation methods focus on the task of inertial odometry, by taking gyroscope and accelerometer readings as input and regressing for integrated IMU poses (i.e., position and orientation). While this design has been successfully applied on a number of applications, it is *not* of theoretical performance guarantee unless patterned motion is involved. This inevitably leads to significantly reduced accuracy and robustness in certain use cases. To solve this problem, we design a framework to compute *observable* IMU integration terms using DNNs, followed by the numerical pose integration and sensor fusion to achieve the performance gain. Specifically, we perform detailed analysis on the motion terms in IMU kinematic equations, propose a dedicated network design, loss functions, and training strategies for the IMU data processing, and conduct extensive experiments. The results show that our method is generally applicable and outperforms both traditional and DNN methods by wide margins.**

I. Introduction

A typical inertial measurement unit (IMU) consists of a three-axis gyroscope and a three-axis accelerometer, which measure rotational velocity and gravity-compensated linear acceleration (i.e. specific force) of a moving platform respectively. As the IMU captures motion information without relying on any external infrastructure, it has been extensively used in navigation systems and applications, e.g., mobile devices [1], [2], ground robots [3], miniature drones [4], autonomous vehicles [5], and so on. Up to the hardware design and manufacturing process, the price of an IMU varies can vary over a wide range, e.g. 1-10k dollars [6]. In this work, we focus on the widely-used low-cost IMUs, whose data quality is much lower than that of the high-end IMUs. This makes algorithms and systems of either improving the IMU data quality or enhancing the data processing pipeline of noteworthy significance in the research community.

To process the IMU data, most methods exploit highly hand-crafted models to approximate sensor characteristics and the underlying motion dynamics. A representative pipeline can be summarized by [7]: data filtering (e.g., denoising), intrinsic model compensation, data interpolation and extrapolation, pose integration, probabilistic sensor fusion or update. The first two steps are generally recognized as data pre-processing, whose objective is to obtain a 'cleaned' version of sensory input by reducing errors caused by mechanical, electronic, and signal processing imperfections. The next two steps are to calculate IMU's integrated orientation and position based on the underlying motion dynamics. To reduce the inevitable drift from the IMU integration, probabilistic update using data from other sensors [1], [4] or motion constraints [8], [9] is also implemented in a variety of applications.

The authors are with Alibaba Group, Hangzhou, China. {mingzhang | mingmingzhang | yimingcheng | mingyangli}@alibaba-inc.com.

Although the *state-of-the-art* IMU data processing pipeline is mature, the current design is still subject to a number of non-negligible error sources. One of the most critical components is the sensor modeling error introduced in the IMU data pre-processing steps, which has been extensively studied in the literature, especially the IMU raw data signal processing model [7], [10] and the intrinsic model [7], [11]–[13]. However, existing sensor models are heavily hand-crafted (e.g., number, order, and arrangement of linear and nonlinear functions in Euclidean and $SO(3)$ space), which lack of optimality guarantee and generality across different devices and practices.

In this work, inspired by recent effort of using DNN in inertial navigation algorithms [9], [14]–[17], we propose a novel data driven method for the inertial sensor modeling to achieve enhanced performance. In particular, we design a recurrent neural network (RNN) based method to process raw gyroscope and accelerometer measurements to reduce the modeling errors and allow improved sensor fusion. To ensure generality of our approach under different use cases, we train our network by regressing the observable IMU motion terms, instead of the relative IMU position and orientation that are adopted in most literature [14], [16]. Additionally, we propose a dedicated method on processing IMU measurements, including multiple loss functions, training strategies, and data preparation (data pre-processing, augmentation, and so on), which are critical in applying our method for high-precision inertial aided navigation. To validate the effectiveness of the proposed method, extensive real-world experiments are also conducted in this work, by utilizing the datasets collected from automotive vehicles [18], drones [19], and ground robots. The results show that the proposed method outperforms the various competing methods, including both traditional and DNN based ones [2], [3], [7], [17], [20], by wide margins.

II. Related Work

Low-cost IMUs are subject to multiple error sources, including the ones coming from mechanical, electronic, and signal processing imperfections. To date, most existing methods on reducing the noises in IMU readings and compensating for the intrinsic models are based on statistical approaches. Standard methods in filtering the IMU readings in inertial navigation are to eliminate or attenuate the high frequency components [10]. Auto regressing and moving average (ARMA) methods are also widely applied in this field [21]. In addition to noises, IMU readings are also affected by sensor biases, which are generally modeled as additive components in the measurement equations [11]. For high-end IMUs, modeling noises and biases might be enough for high-precision navigation [7]. However, to have high-quality performance, additional parameters must be considered for low-cost IMUs, including scale factors, misalignment parameters, G-sensitivity matrix, and other high order terms [12], [22]. Compensating for the thermal effects is another important factor that is widely studied and must be taken into consideration in real applications [23]. Once readings from IMUs are properly filtered and a closed-form sensor model equation is given, IMU poses can be straightforwardly computed by numerical integration following the Newton's law [1], [7].

However, due to the hardware complexity of low-cost IMUs, it becomes almost infeasible for researchers and engineers to formulate the *exact* mathematical equations to describe the sensor noise and intrinsic models. Therefore, using data driven methods instead of the model based ones in this domain becomes worthy exploring. Motivated by the recent success of DNN in a variety of tasks, researchers started to look into methods that represent IMU models with DNNs to improve the inertial navigation performance [2], [9], [14]–[17]. The majority of existing DNN methods are designed to perform the end-to-end IMU integration [9], [14], [16], by taking sensor data as inputs and integrating poses using neural networks. However, such design regresses both observable or un-observable system terms [12], so its robustness can only be guaranteed when patterned motions are involved. Alternatively, there are DNN based methods that target on refining sensor models [15], [17]. However, both [15], [17] can not be applied to general-case pose integration since they are either optimized for stationary periods or only able to handle rotations.

Since noises and biases in IMU measurements are unavoidable, even with proper sensor model, the accumulated drift in the integrated IMU poses will inevitably occur. To reduce the long-term pose drift, there are generally two families of algorithms: by fusing IMU readings with measurements from other sensors [1], [3], [7], [24] or relying on motion constraints of the platform where the IMU is installed on [8], [9], [25]. Representative sensors that are widely used in combination with IMU include global positioning system (GPS) [7], visual sensors [1], laser range finders [24], wheel odometers [3], and so on. On the other hand, if repetitive or special motion patterns are involved in the inertial navigation, they can also be probabilistically utilized to reduce the estimation errors. Typical motion patterns used in the navigation systems are zero velocity events [8], [9] or pedestrian patterns [25]. There are also algorithms that utilize both measurements from complementary sensors and motion constraints to reduce pose errors [4], [26]. For instance, Kottas et al. [26] proposed a method on visual-inertial odometry by integrating a module of hovering motion detection, and VIMO [4] designed a system that simultaneously estimates IMU poses and force using multi-sensory inputs.

In this work, we focus on proposing a DNN based method that is to model IMU measurements using data driven methodology. In addition, we also ensure that the filtered IMU readings can be fused with measurements from other sensors to conduct inertial navigation, with higher accuracy compared to competing methods. The rest of the paper is organized as follows. Section III provides mathematical details on IMU integration equations, discusses on generally learn-able terms, and presents our design choices. Section IV describes the details of our DNN design, including the network architecture, loss functions, training strategies, and data preparation. Finally, Section V presents our experimental results from multiple testing platforms which validate the advantages of our method.

III. IMU MODELS

A. IMU Measurement Equations

In this work, we assume an IMU, $\{I\}$, moves with respect to a global frame, $\{G\}$. The corresponding IMU measurement model can be described by:

$$[\boldsymbol{\omega}_{rm}^T, \boldsymbol{a}_{rm}^T]^T = f_\pi \left({}^I\boldsymbol{\omega}, {}^I\boldsymbol{a}, \boldsymbol{\pi}, \boldsymbol{n}\right) \tag{1}$$

where $\boldsymbol{\omega}_{rm}$ and $\boldsymbol{a}_{rm}$ are the **raw** gyroscope and accelerometer data, ${}^I\boldsymbol{\omega}$ the angular rate in the IMU frame $\{I\}$, f_π the IMU measurement model and $\boldsymbol{\pi}$ the model parameter vector, and $\boldsymbol{n}$ the noise terms. Additionally, ${}^I\boldsymbol{a} = {}^I_G\mathbf{R}({}^G\boldsymbol{a} - {}^G\boldsymbol{g})$, where ${}^G\boldsymbol{a}$ is the linear acceleration in global frame $\{G\}$, ${}^I_G\mathbf{R}$ the rotation from $\{G\}$ to $\{I\}$, and ${}^G\boldsymbol{g}$ the known gravity (e.g., ${}^G\boldsymbol{g} = [0, 0, -9.8m/s^2]^T$).

To design f_π, most inertial navigation algorithms rely on hand-crafted pre-processing steps for measurement denoising and intrinsic compensation:

$$[\boldsymbol{\omega}_m^T, \boldsymbol{a}_m^T]^T = g_\pi \left(\boldsymbol{\omega}_{rm}, \boldsymbol{a}_{rm}, \boldsymbol{\pi}\right) \tag{2}$$

where g_π is the pre-processing model, $\boldsymbol{\omega}_m$ and $\boldsymbol{a}_m$ the preprocessed measurements, commonly approximated by linear models, e.g., bias model: [1], [7], [11]:

$$\boldsymbol{\omega}_m = {}^I\boldsymbol{\omega} + \boldsymbol{b}_g + \boldsymbol{n}_g, \;\; \boldsymbol{a}_m = {}^I\boldsymbol{a} + \boldsymbol{b}_a + \boldsymbol{n}_a \tag{3}$$

where $\boldsymbol{b}_g$ and $\boldsymbol{b}_a$ the biases and $\boldsymbol{n}_g$ and $\boldsymbol{n}_a$ measurement noises. Our objective is to represent g_π and $\boldsymbol{\pi}$ via a data driven way to achieve better performance compared to existing hand-crafted methods.

B. IMU Based Pose Integration

To describe the detailed formulation of our method, we first present the IMU state vector and integration equations that are generally used in the inertial navigation. Following [1], [11], [27], the IMU state is defined as:

$$\boldsymbol{x} = \left[{}^G_I\bar{\boldsymbol{q}}^\mathsf{T}, \; \boldsymbol{b}_g^\mathsf{T}, \; {}^G\boldsymbol{v}_I{}^\mathsf{T}, \; \boldsymbol{b}_a^\mathsf{T}, \; {}^G\boldsymbol{p}_I{}^\mathsf{T} \; \right]^\mathsf{T} \in \mathbb{R}^{16} \tag{4}$$

where ${}^G_I\bar{\boldsymbol{q}}$ presents rotation from IMU frame to global frame (i.e. ${}^G_I\mathbf{R}$) in quaternion [11], and ${}^G\boldsymbol{p}_I$ and ${}^G\boldsymbol{v}_I$ stand for the IMU's position and velocity.

Denoting ${}^G_I\bar{\boldsymbol{q}}(t_k)$, ${}^G\boldsymbol{p}_I(t_k)$, and ${}^G\boldsymbol{v}_I(t_k)$ for IMU's rotation, position, and velocity at time t_k, the IMU integration kinematic equations are:

$$ {}^G_I\bar{\boldsymbol{q}}(t_{k+1}) = {}^G_I\bar{\boldsymbol{q}}(t_k)\Delta\bar{\boldsymbol{q}} \tag{5} $$

$$ \begin{aligned} {}^G\boldsymbol{v}_I(t_{k+1}) &= {}^G\boldsymbol{v}_I(t_k) + \int_{t_k}^{t_{k+1}} {}^G\boldsymbol{a}_I(\tau)d\tau \\ &= {}^G\boldsymbol{v}_I(t_k) + {}^G\boldsymbol{g}\Delta t + {}^G_I\boldsymbol{R}(t_k){}^{I(t_{k+1})}_{I(t_k)}\boldsymbol{\beta} \end{aligned} \tag{6} $$

$$ \begin{aligned} {}^G\boldsymbol{p}_I(t_{k+1}) &= {}^G\boldsymbol{p}_I(t_k) + {}^G\boldsymbol{v}_I(t_k)\Delta t + \int_{t_k}^{t_{k+1}}\int_{t_k}^{\tau} {}^G\boldsymbol{a}_I(s)dsd\tau \\ &= {}^G\boldsymbol{p}_I(t_k) + {}^G\boldsymbol{v}_I(t_k)\Delta t + \frac{1}{2}{}^G\boldsymbol{g}(\Delta t)^2 \\ &\quad + {}^G_I\boldsymbol{R}(t_k){}^{I(t_{k+1})}_{I(t_k)}\boldsymbol{\gamma} \end{aligned} \tag{7} $$

where , $\Delta t = t_{k+1} - t_k$, and

$$ {}^{I(t_{k+1})}_{I(t_k)}\boldsymbol{\beta} = \int_{t_k}^{t_{k+1}} {}^{I(t_k)}_{I(\tau)}\boldsymbol{R} \cdot {}^{I(\tau)}\boldsymbol{a}d\tau \tag{8} $$

$$ {}^{I(t_{k+1})}_{I(t_k)}\boldsymbol{\gamma} = \int_{t_k}^{t_{k+1}}\int_{t_k}^{\tau} {}^{I(t_k)}_{I(s)}\boldsymbol{R} \cdot {}^{I(s)}\boldsymbol{a}dsd\tau \tag{9} $$

We also define $\Delta\bar{\boldsymbol{q}} = {}^{I(t_{k+1})}_{I(t_k)}\bar{\boldsymbol{q}}$, $\Delta\boldsymbol{\beta} = {}^{I(t_{k+1})}_{I(t_k)}\boldsymbol{\beta}$, $\Delta\boldsymbol{\gamma} = {}^{I(t_{k+1})}_{I(t_k)}\boldsymbol{\gamma}$ as simplified notation for later usage in this work. We note that in the above equations $\Delta\bar{\boldsymbol{q}}$, $\Delta\boldsymbol{\beta}$, and $\Delta\boldsymbol{\gamma}$ are fully characterized by ${}^{I(t)}\boldsymbol{\omega}$ and ${}^{I(t)}\boldsymbol{a}$, $t_k \leq t \leq t_{k+1}$, and independent of all other variables. All intermediate rotation terms in Eqs. 8 and 9, i.e. ${}^{I(t_k)}_{I(\tau)}\boldsymbol{R}$ and ${}^{I(t_k)}_{I(s)}\boldsymbol{R}$, can be computed by integrating ${}^{I(t)}\boldsymbol{\omega}$.

C. Learn-able Terms

Existing DNN methods seek to directly regress the relative position and rotation terms [14], [16], [20]:

$$ \Delta\boldsymbol{q} = {}^G_I\bar{\boldsymbol{q}}(t_k)^{-1}{}^G_I\bar{\boldsymbol{q}}(t_{k+1}) \tag{10} $$

$$ \Delta\boldsymbol{p} = {}^G_I\boldsymbol{R}^T(t_k)\left({}^G\boldsymbol{p}_I(t_{k+1}) - {}^G\boldsymbol{p}_I(t_k)\right) \tag{11} $$

While computing $\Delta\bar{\boldsymbol{q}}$ is feasible via Eq. 5, obtaining the relative position term $\Delta\boldsymbol{p}$ requires ${}^{I(t_k)}\boldsymbol{v}_I(t_k)$ and ${}^{I(t_k)}\boldsymbol{g} = {}^G_I\boldsymbol{R}^T(t_k){}^G\boldsymbol{g}$ from Eq. 7. If a DNN method is able to compute $\Delta\boldsymbol{p}$ by using windowed IMU measurements, this network must also obtain ${}^{I(t_k)}\boldsymbol{v}_I(t_k)$ and ${}^{I(t_k)}\boldsymbol{g}$ as hidden variables. However, both terms are *not* necessarily observable using IMU measurements [7], unless patterned motion [14], zero velocity event [9], or other special cases are involved. In other words, it is always possible that under the same IMU measurements the corresponding position ($\Delta\boldsymbol{p}$) is different.

To avoid this problem and allow general usage, we propose to leverage RNN methods to compute the learnable terms $\Delta\boldsymbol{\gamma}$, $\Delta\boldsymbol{\beta}$, and $\Delta\boldsymbol{q}$ instead of the relative poses. As analyzed previously, all those terms are functions of the motion dynamic terms ${}^I\boldsymbol{\omega}$ and ${}^I\boldsymbol{a}$ only, and independent of prior information on local gravity direction, local velocity term, and so on. Since ${}^I\boldsymbol{\omega}$ and ${}^I\boldsymbol{a}$ are both measured by the IMU sensor, our method directly captures the intrinsic model of an IMU. By contrast, the network in previous methods [14], [16], [20] express both IMU models and motion patterns in the training datasets. Our design also aligns well with the closed-form solutions that use $\Delta\boldsymbol{\gamma}$, $\Delta\boldsymbol{\beta}$, and $\Delta\boldsymbol{q}$ for later sensor fusion [1], [28].

IV. Learning Method

A. Network Architecture

The overall architecture of our network is shown in Fig. 1, which consists of both standard RNN and closed-form IMU integration modules. Specifically, the RNN module is used to represent the IMU model (de-noise, intrinsic, and others), which takes raw measurements as input and calculates refined values. The differentiable integration module is used to generate $\Delta\boldsymbol{\gamma}, \Delta\boldsymbol{\beta}$ and $\Delta\boldsymbol{q}$ for loss functions. We note that, an alternative solution is to use RNN layers to directly regress $\Delta\boldsymbol{\gamma}, \Delta\boldsymbol{\beta}$ and $\Delta\boldsymbol{q}$ instead of adding the extra integration blocks. However, integration computation requires a combination of operations in Euclidean and $SO(3)$ space (especially the ones relate to rotation terms), which can not be well expressed by standard light-weighted RNN layers. In our method, we use stacked bidirectional LSTM connected by fully-connected (FC) layers for RNN, and utilize the parameter-free closed-form differential equations [1] for the integration module.

In the training stage, the output of RNN layer will be passed to the integration layer for calculating the training loss. In the reference stage, the RNN layer computes the learned IMU measurements, which will be directly used as the input for the probabilistic sensor fusion. We note that, our network architecture is relatively preliminary, which can be further largely optimized via a number of methods (e.g., using [29]). However, our results show that the current design already outperforms competing methods significantly, which strongly support our design purpose of using DNNs to process IMU data in inertial navigation.

B. Loss Function

For the task of learning high precision IMU models, the ideal method is to design a loss function to measure the error between the learned IMU measurements and the ground truth ones. However, such design is not practical since it is almost impossible to obtain the 'ground-truth' rotational velocities and linear accelerations of real IMU hardware devices. Therefore, we must utilize the indirect loss terms on the training stage. In this work, we define loss terms as:

$$ \mathcal{L}_q = |log(\Delta\boldsymbol{q}_s \otimes \Delta\hat{\boldsymbol{q}}^{-1})|_h \tag{12} $$

$$ \mathcal{L}_v = |\Delta\boldsymbol{\beta}_s - \Delta\hat{\boldsymbol{\beta}}|_h \tag{13} $$

$$ \mathcal{L}_p = |\Delta\boldsymbol{\gamma}_s - \Delta\hat{\boldsymbol{\gamma}}|_h \tag{14} $$

where $\boldsymbol{a}_s$ and $\hat{\boldsymbol{a}}$ represent ground-truth and DNN output value of the variable $\boldsymbol{a}$, $|\cdot|_h$ denotes the Huber loss function, $\otimes$ is the quaternion multiplication operator, and $log(\cdot)$ is the logarithm operator in $SO(3)$ space. It is important to note that, in this work, the ground-truth used in training are provided by either additional high-quality sensors or sensor fusion algorithms (see Sec. V for details). Therefore, the selected Huber loss operator is able to improve training robustness, to systematically handle possible errors and outliers in the training ground truth.

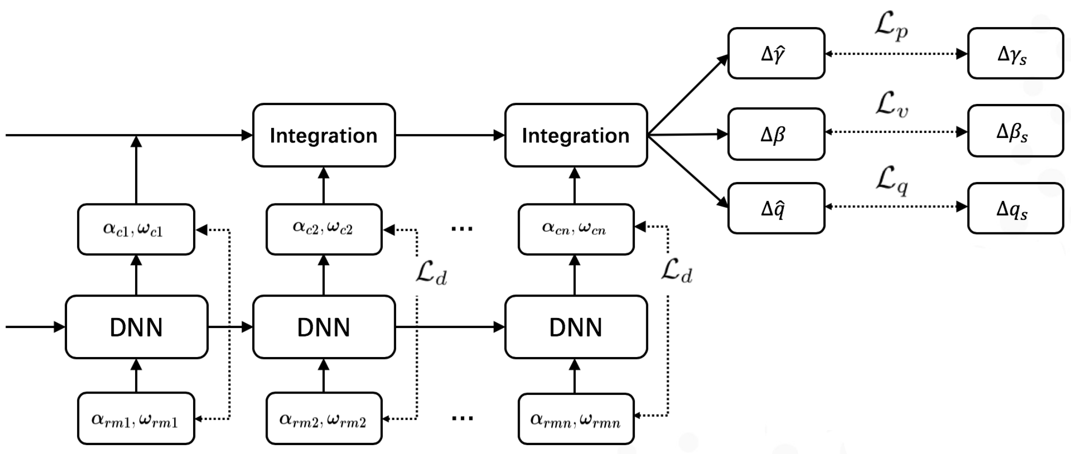

Fig. 1. The proposed network and learnable terms.

Additionally, we use a regularization loss

$$\mathcal{L}_d = max\left(|\boldsymbol{u}_m - \hat{\boldsymbol{u}}| - \lambda, 0\right) \tag{15}$$

where $\boldsymbol{u}_m$ is the raw IMU measurement, $\hat{\boldsymbol{u}}$ is the refined one, and λ is a control parameter. This term penalizes the refined IMU values only when they deviate from the raw measurements above a threshold, which is to ensure fast network convergence. The design motivation of this term is as follows. Although the IMU measurements are subject to multiple error sources, directly integrating IMU data will still lead to poses of certain quality. To train a network, we would like to start from that 'certain' quality, instead of starting from scratch. Additionally, this term also ensures that the IMU intrinsic models only incur a certain amount of changes in the corresponding measurements values and reject unlimited changes, which stem from the process of how the IMU measurements are generated electronically.

C. Training

The ground-truth training signals $\Delta\boldsymbol{q}_s$, $\Delta\boldsymbol{\beta}_s$, and $\Delta\boldsymbol{\gamma}_s$, as shown later in experiments, are derived through manipulating Eq. 6 and 7 with the ground-truth poses. The ground-truth poses are from either high-fidelity sensors or external infrastructures. Additionally, Fig. 1 shows that we compute the loss function by integrating all measurements within a time window. Furthermore, to enhance the training, integration is conducted by integrating 20%, 40%, 60%, 80%, and 100% data to formulate multiple loss terms. This is designed since sensor fusion algorithm using IMU measurements might need integration for different duration times and this is naturally the 'data augmentation' for better training results.

To enhance our network performance, we also propose dedicated data augmentation strategies. Specifically, we start with different indexes of IMU data to make windowed IMU input for training, and add manually generated noise into raw IMU measurements. Furthermore, since our refined measurement still contains bias [1], we also manually add random biases during training. The loss function thus becomes:

$$\mathcal{L}_q = |log(\Delta\boldsymbol{q}_s \otimes \Delta\hat{\boldsymbol{q}}^{-1} \otimes \boldsymbol{q}_b^{-1})|_h \tag{16}$$

$$\mathcal{L}_v = |\Delta\boldsymbol{\beta}_s - \Delta\hat{\boldsymbol{\beta}} - \hat{\boldsymbol{\beta}}_b|_h \tag{17}$$

$$\mathcal{L}_p = |\Delta\boldsymbol{\gamma}_s - \Delta\hat{\boldsymbol{\gamma}} - \hat{\boldsymbol{\gamma}}_b|_h$$

where $\hat{\boldsymbol{q}}_b$, $\hat{\boldsymbol{\beta}}_b$, and $\hat{\boldsymbol{\gamma}}_b$ are the integrated biases.

V. Experiments and Results

A. Setup

To demonstrate the performance of the proposed method, we conducted experiments on three platforms:

1) EuRoC [19]: the dataset contains image and inertial data from a drone. The measurements recorded by the image sensors and IMU were 20Hz and 200Hz respectively. Poses provided from a Vicon motion capture system are used as ground truth for training and also for computing RMS errors for testing.
2) KAIST [18]: collected from automotive vehicles in urban scenes. A sensor fusion method using RTK-GPS, an IMU, and 3D LiDAR is used to generate the ground truth poses. The measurements recorded by the image sensors and IMU were 10Hz and 100Hz respectively.
3) Ground robots: collected from a ground robot (Clearpath), with 10Hz images and 200Hz IMU data. Ground truth poses are computed by RTK-GPS assisted sensor fusion algorithm (RTK-VIO). Since the RTK-VIO algorithm outputs high precision poses as the image is received, the ground truth pose is not at the exact IMU measurement timestamp. We interpolate the ground truth pose to align with the nearest IMU timestamp.

For each dataset, we randomly chose half of the sequences for training and the others for testing. Our method was implemented on Tensorflow 2.2.0, in which ADAM optimizer was used with a learning rate initialized at 0.0001 and training epochs at 700. The training was carried out on a single NVIDIA GeForce GTX 1080 Ti GPU with a batch size of 32

[1]Using the trained model to de-bias completely at the inference stage is not practically feasible, since the bias is time-varying.

TABLE I

Positional RMSE (m) over the entire trajectories, using the five test sequences in the EuRoC dataset..

Sequence	Tr. [7]	VINS [30]	TLIO [2]	End-to-End learning [20]	Pr.
MH_02_easy	0.185	0.150	0.303	3.307	**0.150**
MH_04_difficult	0.151	0.320	0.823	5.199	**0.138**
V1_03_difficult	0.267	0.180	0.653	5.329	**0.147**
V2_02_medium	0.111	0.160	0.509	2.897	**0.104**
V1_01_easy	0.081	0.079	0.343	8.166	**0.066**
Average	0.159	0.179	0.526	4.980	**0.121**

TABLE II

RMSE in relative translation (m) and orientation (rad) over ten IMU frames for different methods, using the five test sequences in the EuRoC dataset. The reported numbers are in the format of (translation (m) / orientation (rad)).

Sequence	Tr. [7]	Gyro-Denoising [17]	TLIO [2]	End-to-End learning [20]	Pr.
MH_02_easy	0.0108 / 0.0093	- / 0.0060	0.0053 / 0.0045	0.0077 / 0.0072	**0.0028** / **0.0025**
MH_04_difficult	0.0237 / 0.0103	- / 0.0061	0.0078 / 0.0058	0.0092 / 0.0078	**0.0058** / **0.0037**
V1_03_difficult	0.0189 / 0.0168	- / **0.0051**	0.0151 / 0.0099	0.0130 / 0.0107	**0.0053** / 0.0067
V2_02_medium	0.0191 / 0.0130	- / 0.0118	0.0127 / 0.0083	0.0131 / 0.0122	**0.0083** / **0.0068**
V1_01_easy	0.0118 / 0.0234	- / 0.0168	0.0053 / 0.0181	0.0075 / 0.0219	**0.0030** / **0.0162**
Average	0.0169 / 0.0146	- / 0.0092	0.0092 / 0.0093	0.0101 / 0.0120	**0.0050** / **0.0072**

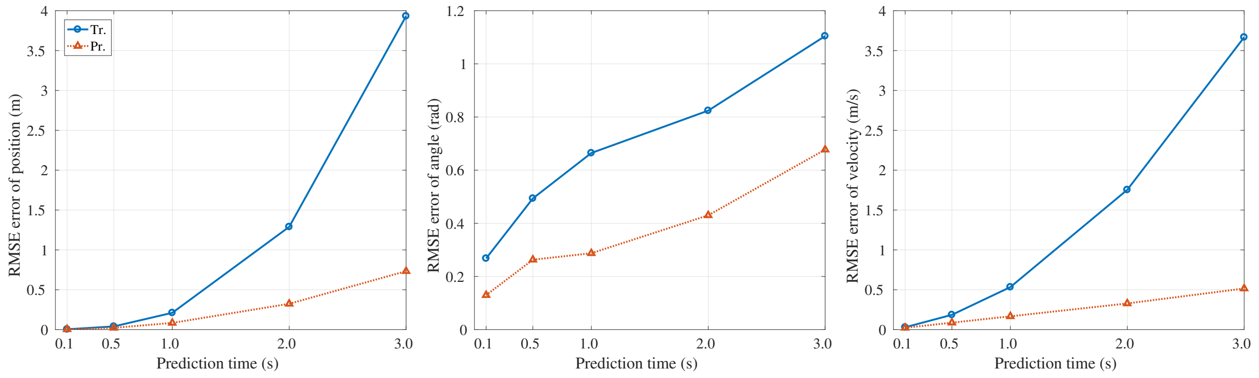

Fig. 2. IMU integration errors for varying time periods, using the traditional method and proposed one. Left: Errors in predicted position; Middle: Errors in predicted orientation; Right: Errors in predicted velocity.

samples. The model with the best validation loss throughout the training was chosen as the final one for the testing.

B. Competing Methods

We compared the proposed method (termed Pr.) against five competing algorithms:

1) Traditional method (termed Tr.): This method is to pre-process IMU data using the traditional approaches, i.e., a low-pass filter followed by the intrinsic modeling [7]. The results are used either standalone for integration or fused with the visual data for long term navigation [3].
2) VINS [30]: The second one is a representative state-of-the-art visual-inertial odometry method with open-source implementation, which also relies on analytical equations to process IMU measurements.
3) End-to-end learning (termed EL.) method [20]: The third one is a competing DNN method by learning the relative position and rotation directly, with our customized RNN implementation.
4) TLIO [2]: The fourth method is the EKF and deep learning combined method, which demonstrates high-quality performance in augmented reality applications.
5) Gyro de-noising [17]: The last one is a DNN based gyroscope de-noising method for attitude estimation.

We note that the formulation of the first two competing methods and the proposed one allow for sensor fusion with visual sensors, while other methods yield pose estimates only based on IMU measurements. Additionally, the gyro de-noising [17] method only computes rotational estimates, while all other methods compute both positional and rotational measurements. The motivation of also comparing [17] in experiments is due to the fact that the algorithm design in that work is similar to us by pre-filtering the sensory data.

C. Tests on EuRoC Dataset

Since EuRoC [19] contains high-quality ground truth values for both rotational and positional terms, more detailed tests are performed under the corresponding sequences.

Long-term Inertial Navigation Accuracy Test: The first test is to demonstrate the most important results: the pose estimation accuracy. In this test, the processed IMU measurements by both the proposed and the traditional methods are fused respectively with only visual sensor data using the approach of [3], for open-loop positional estimation. On the other hand, the EL. [20] and TLIO [2] methods only utilize IMU measurements. Note that, in this test, we sought to obtain the 'best possible' inertial navigation accuracy for all competing algorithms. By computing observable IMU terms, both Tr. and Pr. allow sensor fusion probabilistically. On the other hand, since EL. [20] and TLIO [2] directly regress for IMU's relative poses, the standard methods of fusing visual and inertial data become not applicable. Results of VINS are the reported ones in [30].

In this test, we split the training and testing sequences similar to the tests conducted in the original paper of EL. [20]. To show the results, we compared the root-mean-squared errors (RMSE) between the ground-truth IMU positions and the estimated ones by different algorithms. The results are shown in Table I. Those computed positional RMSE values clearly demonstrate that the proposed method outperforms competing ones by wide margins. Compared to traditional method and VINS [30], our method is able to better represent the IMU model. Compared to the DNN methods, our method regresses learn-able terms which are of better accuracy and make probabilistic sensor fusion possible.

Relative Pose Accuracy: In addition to knowing the long-term inertial navigation accuracy, it is also interesting to investigate the relative pose accuracy within short time windows. In this test, the RMSE for the relative position and orientation over 10 IMU readings are computed for different algorithms, and the results are reported in Table. II. We note that, compared to the previous test, we also added [17] into the competing algorithms, which is an algorithm that only seeks to regress for rotational estimates. We also note that visual measurements are not used in Tr. and Pr. and VINS [30] is not compared against here, due to the goal of looking into the short-term pose accuracy.

Results in Table. II show that the proposed method outperforms all competing methods clearly. The average RMSE of position and orientation of our method are 54% and 78% lower than the best competing method. This test clearly demonstrates the advantage of our method, including both loss function design and the training strategies.

Pose Integration Accuracy: The next test is to demonstrate the accuracy of IMU prediction between varying time periods for the proposed and traditional methods. This is due to the fact that, the design of those two methods allows for probabilistic sensor fusion with other sensors, and the frequency of other sensors might vary significantly (e.g., camera at 10 Hz or GPS at 1 Hz). Thus, IMU based prediction might be conducted for different time periods in real applications. In this test, we relied on the known IMU poses (position, rotation, and velocity), and used only IMU measurements to predict future poses. For both methods, we computed the RMSE for position, rotation, and velocity.

The results are shown in Fig. 2, which demonstrate that our method achieves significantly improved accuracy compared to the competing one. This validates our theoretical analysis and our design motivation that a data driven method can potentially provide better IMU models for forward short-time integration compared to the hand-crafted method. The results also indicate that our method is more preferable for probabilistic sensor fusion.

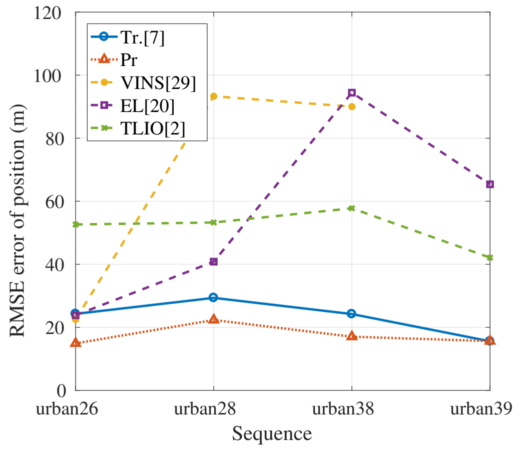

Fig. 3. Positional RMSE in KAIST dataset (VINS [30] fails on urban39).

TABLE III

AVERAGE POSITIONAL RMSE ERRORS (M) FOR COMPETING METHODS.

Dataset	KAIST	EuRoC	Ground robot
Tr. [7]	23.3482	0.1592	4.7007
Pr.	**17.4665**	**0.1210**	**3.7588**
VINS [30]	68.6	0.1778	8.2478
EL [20]	56.1167	2.0402	11.3905
TLIO [2]	51.4357	0.5262	9.8441

D. Tests on KAIST Dataset and Ground Robots

To demonstrate the generality of our algorithm, we also conducted tests on KAIST dataset and ground robots. Fig. 3 shows positional RMSE per sequence in KAIST dataset and Table. III shows average RMSE across different datasets. We emphasize that, for those testing cases, the proposed algorithm consistently outperforms all competing methods by wide margins. Those results clearly show that our method is *not* application specific and can be applied in different use cases for performance gain.

VI. CONCLUSION

To summarize, in this work we propose a framework with a data driven algorithm to obtain complex sensor model, which naturally outperforms hand-crafted ones and other deep learning based methods. In addition, our method regresses observable IMU kinematic terms, which is of theoretical guarantee and can be easily integrated for probabilistic sensor fusion. Both properties are not achieved by the competing DNNs. The experimental results on different testing platforms demonstrate that our algorithm is generally applicable and outperforms the competing state-of-the-art methods significantly.

References

[1] M. Li and A. I. Mourikis, "High-precision, consistent EKF-based visual-inertial odometry," *The International Journal of Robotics Research*, vol. 32, no. 6, pp. 690–711, 2013.

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

[3] M. Zhang, X. Zuo, Y. Chen, and M. Li, "Pose estimation for ground robots: On manifold representation, integration, re-parameterization, and optimization," *arXiv preprint arXiv:1909.03423*, 2019.

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

[5] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Z. Kolter, D. Langer, O. Pink, V. Pratt *et al.*, "Towards fully autonomous driving: Systems and algorithms," in *IEEE Intelligent Vehicles Symposium (IV)*, 2011, pp. 163–168.

[6] C. Goodall, S. Carmichael, and B. Scannell, "The battle between mems and fogs for precision guidance," Analog Devices Inc., Tech. Rep., 2019.

[7] J. Farrell, *Aided navigation: GPS with high rate sensors*. McGraw-Hill, Inc., 2008.

[8] K. Abdulrahim, T. Moore, C. Hide, and C. Hill, "Understanding the performance of zero velocity updates in mems-based pedestrian navigation," *International Journal of Advancements in Technology*, vol. 5, no. 2, 2014.

[9] B. Wagstaff and J. Kelly, "Lstm-based zero-velocity detection for robust inertial navigation," in *IEEE IPIN*, 2018.

[10] C. H. Kang, S. Y. Kim, and C. G. Park, "Improvement of a low cost mems inertial-gps integrated system using wavelet denoising techniques," *IJASS*, 2011.

[11] N. Trawny and S. I. Roumeliotis, "Indirect kalman filter for 3d pose estimation," *University of Minnesota, Dept. of Comp. Sci. & Eng., Tech. Rep*, vol. 2, 2005.

[12] M. Li, H. Yu, X. Zheng, and A. I. Mourikis, "High-fidelity sensor modeling and self-calibration in vision-aided inertial navigation," in *ICRA*, 2014.

[13] T. Schneider, M. Li, M. Burri, J. Nieto, R. Siegwart, and I. Gilitschenski, "Visual-inertial self-calibration on informative motion segments," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 6487–6494.

[14] C. Chen, X. Lu, A. Markham, and N. Trigoni, "Ionet: Learning to cure the curse of drift in inertial odometry," in *AAAI*, 2018.

[15] C. Jiang, S. Chen, Y. Chen, B. Zhang, Z. Feng, H. Zhou, and Y. Bo, "A mems imu de-noising method using long short term memory recurrent neural networks (lstm-rnn)," *Sensors*, vol. 18, no. 10, p. 3470, 2018.

[16] H. Yan, S. Herath, and Y. Furukawa, "Ronin: Robust neural inertial navigation in the wild: Benchmark, evaluations, and new methods," *arXiv:1905.12853*, 2019.

[17] M. Brossard, S. Bonnabel, and A. Barrau, "Denoising imu gyroscopes with deep learning for open-loop attitude estimation," *arXiv preprint arXiv:2002.10718*, 2020.

[18] Y. Choi, N. Kim, S. Hwang, K. Park, J. S. Yoon, K. An, and I. S. Kweon, "KAIST multi-spectral day/night data set for autonomous driving," *TITS*, 2018.

[19] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The euroc micro aerial vehicle datasets," *IJRR*, 2016.

[20] J. P. Silva do Monte Lima, H. Uchiyama, and R.-i. Taniguchi, "End-to-end learning framework for imu-based 6-dof odometry," *Sensors*, vol. 19, no. 17, p. 3777, 2019.

[21] A. G. Quinchia, G. Falco, E. Falletti, F. Dovis, and C. Ferrer, "A comparison between different error modeling of mems applied to gps/ins integrated systems," *Sensors*, vol. 13, no. 8, pp. 9549–9588, 2013.

[22] Y. Yang, P. Geneva, X. Zuo, and G. Huang, "Online imu intrinsic calibration: Is it necessary?" in *Robotics: Science and Systems*, 2020.

[23] X. Niu, Y. Li, H. Zhang, Q. Wang, and Y. Ban, "Fast thermal calibration of low-grade inertial sensors and inertial measurement units," *Sensors*, vol. 13, no. 9, pp. 12 192–12 217, 2013.

[24] P. Geneva, K. Eckenhoff, Y. Yang, and G. Huang, "LIPS: Lidar-inertial 3d plane slam," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018, pp. 123–130.

[25] A. Ahmed and S. Roumeliotis, "A visual-inertial approach to human gait estimation," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[26] D. G. Kottas, K. J. Wu, and S. I. Roumeliotis, "Detecting and dealing with hovering maneuvers in vision-aided inertial navigation systems," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 3172–3179.

[27] T. Schneider, M. Dymczyk, M. Fehr, K. Egger, S. Lynen, I. Gilitschenski, and R. Siegwart, "maplab: An open framework for research in visual-inertial mapping and localization," *IEEE RAL*, 2018.

[28] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration for real-time visual–inertial odometry," *IEEE TRO*, 2017.

[29] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *arXiv preprint arXiv:1706.03762*, 2017.

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