OriNet: Robust 3-D Orientation Estimation With a Single Particular IMU

Mahdi Abolfazli Esfahani, Han Wang, *Senior Member, IEEE*, Keyu Wu, and Shenghai Yuan

***Abstract*—Estimating the robot's heading is a crucial requirement in odometry systems which are attempting to estimate the movement trajectory of a robot. Small errors in the orientation estimation result in a significant difference between the estimated and real trajectory, and failure of the odometry system. The odometry problem becomes much more complicated for micro flying robots since they cannot carry massive sensors. In this manner, they should benefit from the small size and low-cost sensors, such as IMU, to solve the odometry problem, and industries always look for such solutions. However, IMU suffers from bias and measurement noise, which makes the problem of position and orientation estimation challenging to be solved by a single IMU. While there are numerous studies on the fusion of IMU with other sensors, this study illustrates the power of the first deep learning framework for estimating the full 3D orientation of the flying robots (as yaw, pitch, and roll in quaternion coordinates) accurately with the presence of a single IMU. A particular IMU should be utilized during the training and testing of the proposed system. Besides, a method based on the Genetic Algorithm is introduced to measure the IMU bias in each execution. The results show that the proposed method improved the flying robots' ability to estimate their orientation displacement by approximately 80% with the presence of a single particular IMU. The proposed approach also outperforms existing solutions that utilize a monocular camera and IMU simultaneously by approximately 30%.**

***Index Terms*—Localization, SLAM, deep learning in robotics and automation, autonomous vehicle navigation.**

I. Introduction

INTELLIGENT robots require to know their position and orientation in the environment in order to navigate and make decisions. Various techniques based on different sensor combinations have been proposed to estimate robot movement trajectory, called odometry, in Global Positioning System (GPS) denied environments. Among widely used sensors, the Inertial Measurement Unit (IMU) is a small size and low-cost sensor which is in focus of industries. However, this sensor suffers from measurement noise and bias, which makes the problem of odometry much more complicated to be solved by a single IMU. While recent researches [1] have achieved satisfactory results by fusing IMU accelerometer and gyroscope information with other sensors, solving the problem of odometry with single IMU requires more focus in order to find the optimal - robust and accurate - solution.

IMU measures linear acceleration, with an accelerometer, and angular velocity, with a gyroscope, in the IMU body frame at each time step, which can be used to estimate robot orientation and scale of movement in the world (robot) frame. In the early stage, IMU was fused with GPS in order to implement an accurate odometry system [2], and recently the monocular Visual-Inertial Navigation Systems (VINS) are in the focus of researchers [1]. The existing Inertial Odometry (IO) systems can be classified into traditional pre-integration methods based on the IMU Strapdown Inertial Navigation System (SINS)[3], and learning-based methods [4], [5] which train a network to extract rich feature representation from IMU measurements.

SINS workflow follows the mathematical and physical model of IMU, in which the orientation in the IMU body frame can be measured by taking integral from the gyroscope information over time, and the scale of movement can be measured by taking the double integral from accelerometer measurements after removing the effect of gravity. However, IMU suffers from bias and measurement noises, and so getting integral from measurements, increases the impact of noise and results in wrong IMU orientation and position estimation. In order to reduce the effect of bias and noise, recent works are benefiting from the Extended Kalman Filter (EKF) to model a series of noisy observations. Since EKF requires error evaluation (feedback), methods based on EKF are fusing IMU information with other sensors such as GPS and magnetometer [6]. However, the zero-velocity update is another approach that can be used in some applications, such as pedestrian movement estimation, to implement an EKF [7], [8]. In such a method, IMU stationary periods are detected, and based on the measurements in that period, the noise and bias of the IMU are estimated; such an approach cannot be used on autonomous robots, especially flying robots, which are not stationary for an extended period. In addition to the mentioned approaches, Madgwick *et al.* [9] have allowed IMU data to be used in an analytically derived and optimized gradient descent algorithm, and Valenti *et al.* [10] have introduced a complementary filter to fuse various IMU measurements and achieve a robust estimation.

Deep learning, on the other hand, has achieved satisfactory results in solving various problems, especially in the area of robotics. Clark *et al.* [5] proposed a monocular Visual-Inertial Odometry (VIO) system called *VINET*. The inertial module of their system has a two-layered Long Short Term Memory

Manuscript received September 9, 2019; accepted December 6, 2019. Date of publication December 12, 2019; date of current version January 2, 2020. This letter was recommended for publication by Associate Editor R. Triebel and Editor T. Asfour upon evaluation of the reviewers' comments. *(Corresponding author: Han Wang.)*

The authors are with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, 639798 (e-mail: mahdi001@ntu.edu.sg; hw@ntu.edu.sg; wukeyu@ntu.edu.sg; syuan003@ntu.edu.sg).

Digital Object Identifier 10.1109/LRA.2019.2959507

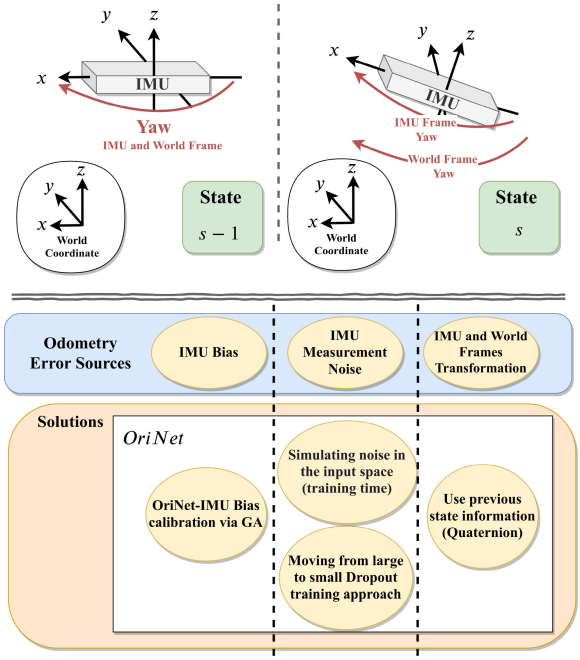


Fig. 1. Challenges and proposed solutions for estimating 3D orientation with a single particular IMU.

(LSTM) architecture to learn and extract rich feature representations from IMU measurements, and they combined these features with features extracted from the camera images to solve the problem of VIO. Afterward, Chen *et al.* [11] proposed the first deep learning-based IO system, called *IONet*, for mobile phones in the application of pedestrian movement estimation. They have used two connected LSTM, each with 96 hidden cells, and estimated the displacement of IMU translation and rotation, for each time window of IMU readings. They have further improved their method, to perform well in different scenarios, by extracting consistent invariant feature representations [12]. We, in our previous work [4], have studied the problem of IO for flying robots in detail and proposed the *AbolDeepIO* network architecture, which models the SINS workflow by Deep Neural Networks. *AbolDeepIO* learns feature representations from the accelerometer, gyroscope, and time interval between consecutive IMU readings in separate channels, and fuses them to achieve the final output. Benefiting from the time interval between consecutive IMU readings as input makes it robust to change or drop in the IMU frequency. While *AbolDeepIO* can outperform existing solutions, similar to others, it measures the length of IMU orientation and position displacement.

In this work, for the first time to our knowledge, an end-to-end deep network architecture is proposed to estimate the full 3D orientation of flying robots with a single particular IMU in the quaternion coordinate. The contributions of this letter, as illustrated in Fig. 1, are:

- proposing the $OriNet$ deep network architecture to estimate the 3D orientation of a robot (world frame) over time, with the presence of a single particular IMU.
- defining terms of *state* and *local time window* space, which helps the network to transfer IMU measurements between the world frame and IMU body frame.
- sliding a local time window over nearby IMU measurements, and outputting various estimation for each time step by considering a different combination of adjacent IMU measurements. Fusing such estimations helps to model uncertainty and achieve a robust solution.
- proposing a training approach which moves the network from exploration of the robust weights to exploitation of the weights near the explored robust weights during training.
- proposing a novel method based on the Genetic Algorithm (GA) to calibrate a trained model with the bias of IMU in each execution.

The proposed network architecture is evaluated on the EuRoC Micro Aerial Vehicle (MAV) dataset [13], and the results show that it not only outperforms existing IO methods but also achieves a result better than VIO methods. For an in-depth review of the history of IO and existing approaches, readers can refer to our previous work [4].

II. Preliminaries

IMU measures 3-axis linear acceleration, $\hat{a}_t$, and 3-axis angular velocity, $\hat{w}_t$, in the IMU body frame at each time step (t) which can be adapted to compute orientation, velocity, and position of the robot. These measurements are influenced by bias and measurement noise. The relation between IMU measurements and the exact and actual linear acceleration (a_t) and angular velocity (w_t) can be modeled as

$$\hat{a}_t = a_t + b_{a_t} + R^t_w g^w + n_{a_t} \tag{1}$$

$$\hat{w}_t = w_t + b_{w_t} + n_{w_t} \tag{2}$$

where b_{a_t} and b_{w_t} are accelerometer and gyroscope bias, and n_{a_t} and n_{w_t} are additive zero-mean Gaussian noises with variances σ^2_a and σ^2_w. The bias of IMU is different in each run and changes very slowly over time. The effect of gravity on accelerometer models by g^w, which is the gravity vector in the world frame. g^w rotates to the IMU body frame with R^t_w in order to influence a_t. By considering the kinematic of IMU and assuming that a and w are constant in-between times $[t, t + \Delta T]$, where ΔT is the smallest time interval between two consecutive IMU readings, the orientation, velocity, and position of the IMU can be measured by

$$R_{t+\Delta T} = R_t Exp(w_t \Delta T) \tag{3}$$

$$V_{t+\Delta T} = V_t + a_t \Delta T \tag{4}$$

$$P_{t+\Delta T} = P_t + v_t \Delta T + \frac{1}{2} a_t \Delta T^2 \tag{5}$$

respectively [14]. However, the exact value of a and w are not available, and noise and bias would change them dramatically. Hence, in order to get the best orientation and position estimation from the IMU measurements, the bias and measurement noise needs to be modeled perfectly, and the noisy measurements $(\hat{a}_t, \hat{w}_t)$ have to transfer to real measurements (a_t, w_t), which is a difficult task. In this work, a deep network architecture is proposed, which aims to output orientation robustly with the presence of noisy observations.

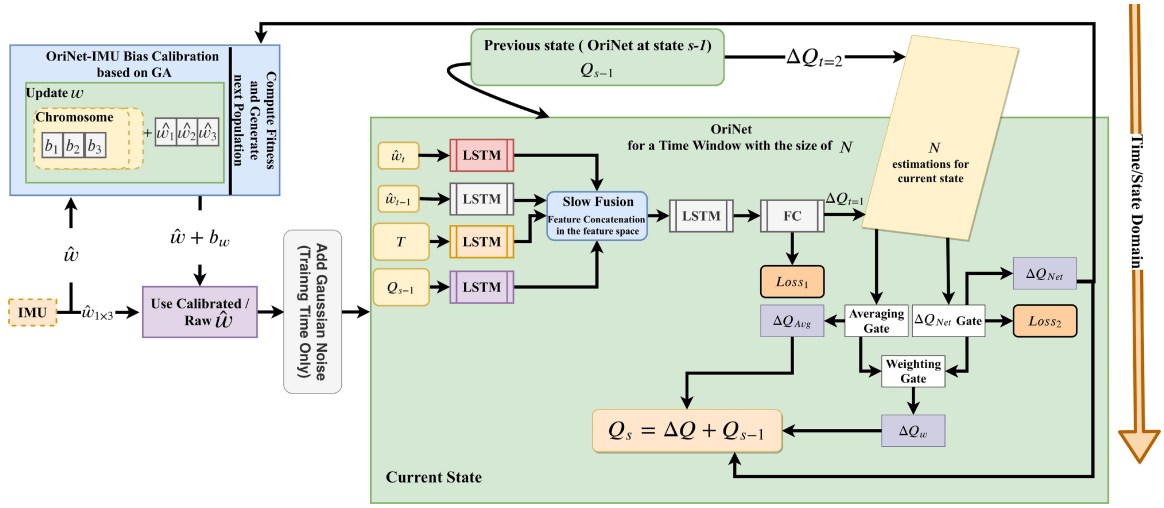


Fig. 2. The procedure of the proposed $OriNet$ module (b_1, b_2 and b_3 are the gyroscope bias, b_w).

III. Proposed Method

A. Network Architecture

This section presents a novel network architecture to estimate the full 3D orientation (yaw, pitch, and roll) in the world frame, with the presence of a single particular IMU. The frame that the orientation of the robot or particular object with connected IMU measures in that is called the world frame; for instance, if an IMU connects to a MAV and the orientation of MAV is required, MAV orientation creates the world frame. It is also important to mention that the estimated orientation is in the quaternion coordinates (as Q_4 where 4 stands for the size of the output, which is correspondent to the estimation of q_w, q_x, q_y and q_z sub-coordinates)

IMU measurements are considered in two spaces in this letter, which are *state space*, and *local time window space*. State space is the global state of the IMU over time, and once quaternion for the time of a particular IMU measurement is computed, a new (quaternion) state is created. In order to create a state space, the LSTM unit is applied on a local time window space with a size of N, which contains N continues IMU measurements.

In the beginning, the network is required to know the exact quaternion of the IMU in the initial state. The local time window starts from the IMU measurement number two, and the exact quaternion in the initial state creates the first state ($Q_{s=1}$). After processing the first local time window, it is shifted by one IMU measurement (starts from measurement number 3) and quaternion regarding the time step of the second IMU measurement is computed based on the outputs of the local time window before shifting, and the second quaternion state ($Q_{s=2}$) is created. From now onwards, index t refers to a time step in a local time window space ($1 \leq t \leq N$), and s demonstrates a state in the state space ($s \geq 1$).

OriNet consists of two parts, and the first part includes four LSTM channels where each get $\hat{w}$ of the current and previous time step, sampling time (time interval between two consecutive IMU readings), T, at each time step in a local time window, and prior state quaternion (Q_{s-1}), as input. Feature representations learned in these channels are fused, by using the concatenation of features in the slow fusion gate. Then, the fused feature representation moves to another LSTM and Fully Connected (FC) gates to output the displacement of quaternion sub-coordinates at any time step, in the local window, relative to its previous time step ($\Delta Q_t = Q_t - Q_{t-1}$). Fig. 2 shows the proposed framework, which is discussed in this section.

It is essential to mention that the accelerometer measurements are not considered as input in our system; Because, based on our experiments, it reduces the performance of the network. While an accelerometer can help to reduce noise and extract robust orientation when IMU is stationary, it can reduce the overall performance. Designing a zero-velocity detection system and benefiting from accelerometer information when IMU is in a stationary state is a good approach which we will focus on in the future.

Similar to our previous work [4], an additive zero-mean gaussian noise layer with the variance of σ_g^2 is applied in the input layer. This layer adds noise to the input in the training time, and so makes the network robust to measurement noise. The value of σ_g^2 is measured by the Probability Density Function (PDF) analysis and is usually available in the IMU datasheets.

Since the gyroscope measures angular velocity relative to the world in the IMU body frame, knowing the latest state quaternion, in the world frame, is essential in order to estimate the next state quaternion (As IMU rotates the IMU body frame changes, and the angular velocity effects differently on the world frame quaternion). In recent deep learning-based VIO frameworks, the impact of previous state quaternion on the gyroscope and accelerometer measurements is ignored, and only the gyroscope and accelerometer measurements at each time step are passed to the LSTM units. One of the main issues that made scientists ignore the previous state quaternion is the structure of LSTM, which does not have a model to accept the latest state info as input; It is due to the lack of state definition which results in considering a local window of data as the only inputs

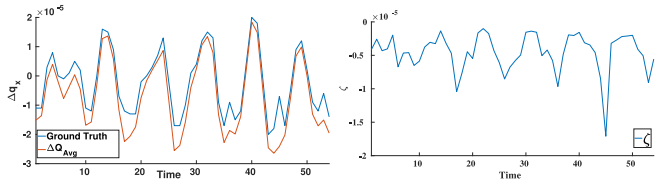
of an LSTM unit. The previous state's output is essential for estimation of the current state, and so, in this work, we benefit from the definition of state and local time window to solve this issue, and previous state quaternion (Q_{s-1}) is considered as the input of the network. In this manner, it is assumed that the initial IMU quaternion ($Q_{s=1}$) is available at the beginning of each test execution. Hence, one limitation of our work is its requirement of the IMU quaternion in the world frame at the first time step in test time, which is not challenging, and is equivalent of starting algorithm from a static quaternion at all the time; for instance you should always put your drone (world frame) which contains IMU on the flat ground looking North, and run the algorithm.

$OriNet$ is applied on a local time window with N measurements. Each time window outputs the displacement of quaternion sub-coordinates (Δq_w, Δq_x, Δq_y, and Δq_z) for each of the time steps, among N, relative to its previous time step. As the time window shifts by one, the computed ΔQ, regarding the first IMU measurement ($t = 1$) in the previous window (state $s - 1$), which is not in the new window (state s), sums by its prior state quaternion (Q_{s-2}), to measure and create the new quaternion state (Q_{s-1}), which is going to be the input for the current state (s) time window. The new quaternion state always can be achieved by following similar steps and summing the displacement of the quaternion, for the first measurement dropped from local time window when window shifts, with its prior state quaternion ($Q_{s-1} = \Delta Q_{s-1} + Q_{s-2}$). In the training time, the ground truth quaternions are used for training, and in the test time, only the first quaternion is required to be known. It is also important to note that, since in the quaternion coordinate $Q = -Q$, in this work, the network is trained by considering both Q and $-Q$, and outputting the smooth Q over time.

Since the local time window shifts one by one, there will be N estimations for each time step. For instance, if $N = 10$, there exists one ΔQ estimation for measurement number 11 when the window begins from measurement number two, and we have another one when we shift the window by one to compute the first estimation for the measurement number 12. It is possible to benefit from all of these estimations and probabilistically measure the final output at each time step. The actual output of the network should be the averaged value of all estimations plus or minus an error term ζ, which can be modeled as

$$\Delta Q_s = \frac{1}{N}\sum_{i=1}^{N} \Delta Q_{s_i} \pm \zeta \tag{6}$$

where ΔQ_s is the quaternion displacement at state s relative to its previous state, and ΔQ_{s_i} is the existing estimations for that state. Fig. 3 is showing the average of all estimations, and ζ for sub-quaternion of q_x in a short period. As it shows, finding the value of ζ is an important requirement for an accurate and robust estimation. In this manner, to control the effect of ζ, a two-layered Multi-Layer Perceptron (MLP) is designed, which gets ΔQ_{s_i} as input and attempts to regress them to the desired ΔQ_s. We define the first term of equation 6, which performs averaging on the N estimations, as the averaging gate with the outcome of ΔQ_{Avg}, and the network which helps to handle ζ is ΔQ_{Net} (with the output of ΔQ_{Net}).



(a) Average of N estimtations (b) ζ

Fig. 3. Importance of ζ which should be handled by ΔQ_{Net}.

The final output of the network, called ΔQ_w, is computed by weighting gate which is defined as

$$\Delta Q_w = \frac{w_1 \Delta Q_{Avg} + w_2 \Delta Q_{Net}}{w_1 + w_2} \tag{7}$$

where w_1 and w_2 are weights which obtain by tunning the ΔQ_w on the training data. The loss function regarding the first part of the network, $Loss_1$ shown in Fig. 2 , is defined as the Mean Square Error (MSE) of the estimated and Ground Truth values regarding the displacement of quaternion sub-coordinates for all measurements that exist in a local time window. Besides, the loss function of ΔQ_{Net}, $Loss_2$ in Fig. 2, is the MSE of the Ground Truth and the estimated values regarding the displacement of quaternion sub-coordinates for a particular time.

B. From Large to Small Dropout Training Approach

Numerous studies have been tried to find the best way of training deep neural networks to avoid overfitting, find robust weights, and solve regression problems [15]. In [4], we have shown that using dropout can help inertial networks to become robust to noise and perform well. In this letter, a new methodology is applied, and the network starts training with a dropout of 0.25 in LSTM layers at the beginning. As time passes, and the loss gets into a stationary state for a short period (i.e. 400 epochs), dropout is reduced by 20%, and the training procedure continues until reaching a new stationary state again. This approach resumes until dropout reaches zero value.

The proposed approach helps the network to learn the robust weights when the dropout is hight, which makes network robust to noise. The first stages of training can be referred to as exploration steps, in which the network tries to find the best robust weights without overfitting. As time passes and dropout reduces, the network moves to the exploitation steps and tries to find the best weights nearby the previously computed robust weights, which keeps the network robust. The results show that this approach helps the network to be robust to noise and also achieve better performance.

C. OriNet-IMU Bias Calibration

IMU suffers from bias and measurement noise. The network gets robust to the noise by benefiting from the input noise layer and the proposed training approach. In order to handle bias, we ignored the small changes of bias over time and assumed it as a fixed value, which decides at the beginning of each execution. We utilized the Genetic Algorithm (GA) [16] to optimize a

TABLE I
TRAIN AND TEST SPLITS OF ASL EUROC MAV DATASET

#	Training Data (Number of IMU Measurements)	Testing Data (Number of IMU Measurements)
1	Machine Hall01 - Easy (36383)	Machine Hall02 - Easy (29993)
2	Machine Hall03 - Medium (26302)	Machine Hall04 - Difficult (19753)
3	Machine Hall05 - Difficult (22212)	Vicon Room 1 03 - Difficult (21500)
4	Vicon Room 1 02 - Medium (17100)	Vicon Room 2 02 - Medium (23490)
5	Vicon Room 2 01 - Easy (22800)	Vicon Room 1 01 - Easy (29120)
6	Vicon Room 2 03 - Difficult (23370)	-

TABLE II
NUMBER OF UNITS IN EACH LAYER OF THE *ORINET*

Layer			
LSTM before SLOW FUSION	**LSTM after SLOW FUSION**	**Fully Connected (FC)**	ΔQ_{Net}
100 (Bidirectional Layer)	250-100 (Two Bidirectional Layers)	20-4 (Two Time Distributed Dense Layers)	150-4 (Two Dense Layers)

TABLE III
CONSIDERED VARIANTS OF THE PROPOSED METHOD

#	Bias Calibration	Average Gate	ΔQ_{Net} Gate	Weighting Gate	Proposed Training Approach
OriNet1	✓	✗	✓	✗	✓
OriNet2	✗	✗	✓	✗	✓
OriNet3	✗	✓	✗	✗	✓
OriNet4	✗	✗	✓	✗	✗
OriNet5	✓	✓	✓	✓	✓

fitness function at the beginning of each test, which computes the bias of IMU. In this manner, it is assumed that IMU is going to be stationary in the first 5 seconds (can have small rotations withing a fixed trajectory for better calibration). The IMU measurements in the stationary period are used to calibrate the IMU with the deep network and compute the bias.

Considering *OriNet* as F, which gets angular velocity $\hat{w}$ and sampling time (T) for all time steps in a local time window, and previous state quaternion (Q_{s-1}) as input, and outputs the Q_s, the fitness function to be optimized is the MSE of the actual quaternion (Q'_s) and estimated quaternion in duration of 5 seconds, at the beginning of the execution. The problem can be formulated as

$$\min_{b_w}\{(F(\hat{w}_t + b_w, \hat{w}_{t-1} + b_w, T, Q_{s-1}) - Q'_s)^2\} \qquad (8)$$

where b_w is the bias parameter that needs to be investigated by the GA. Hence, a population of 20 chromosomes, with the dimensionality of 1×3 are randomly initialized in the range of $[-0.5, 0.5]$; Each chromosome has three values corresponding to the 3 values measured from the gyroscope. During the bias estimation process, the ΔQ_{Net} gate is only considered for computing the quaternion as the final output, and other outputs (ΔQ_w and ΔQ_{Avg}) are ignored. In this manner, the ΔQ_{Net} bias is also considered, and the bias of the gyroscope is computed based on all networks.

IV. EXPERIMENTS AND RESULTS

A. Testing Environment

This section evaluates the proposed method on the EuRoC MAV dataset [13]. This dataset captured images with stereo cameras at 20 frames per second (FPS) and recorded IMU measurements at 200 Hz, from the time MAV is in a stationary state until it takes off, flys and lands on its initial position. They have used a particular IMU in all scenarios. This dataset is a challenging dataset for the focused task because the MAV has lots of sudden motions during its flight, which makes it difficult to estimate MAV orientation at each state. Similar to [4], the dataset is split into training and testing sets which are shown in Table I, and the number of neurons in each layer of the proposed *OriNet* architecture is shown in Table II. $OriNet$ can evaluate a local time window in 3 milliseconds, by benefitting from cuDNN implementation of LSTM, which is a significant improvement compared to 6–9 milliseconds execution time for other deep learning-based approaches. The evaluation goes into detail to understand the effect of each of the proposed modules, which is extremely important. The size of the time window, N, is considered as 11 in all experiments. It is also important to mention that in all experiments, similar to [9], [10] and the single state alignment approach in [17], the frame of the estimated quaternions is aligned with the Ground Truth based on the estimated and Ground Truth values at the first state. After aligning the first state quaternion, based on the quaternion displacement between next time steps and the aligned quaternion (transform in the first state), the new aligned states are created.

In order to analyze the impact of the proposed modules, five different variants of the proposed framework are considered, trained, and tested. The summary of the considered networks is shown in Table III. The performance of each variant is compared with other variants and existing deep learning-based IO frameworks, as well as the traditional SINS framework. Existing deep IO solutions are not estimating the 3D orientation with the presence of single IMU, and are most capable of measuring the length of quaternion displacement in a local time window; Hence, in this work, similar to [4], the length of quaternion displacement estimation in a local time window is computed from the output of our method and compared with others in terms of Mean Absolute Error (MAE) and Root Mean Square Error (RMSE). Besides, the performance of the proposed method for estimating the full quaternion is compared with the traditional inertial odometry techniques which are capable of tracking the orientation along the path. Furthermore, the performance is also compared with the VINS-MONO, VIO framework, which is diagnosed as the top Monocular VIO framework in [18].

B. Comparing IO Methods in Terms of Quaternion Displacement Estimation and Analyzing Modules

This section analyzes the proposed modules by comparing the performance of the considered variants in terms of length of quaternion displacement estimation in a local time window. Length of quaternion displacement for a local time window is defined as

$$|\Delta Q| = \sum_{i=1}^{N} \sqrt{\Delta q_{w_i}^2 + \Delta q_{x_i}^2 + \Delta q_{y_i}^2 + \Delta q_{z_i}^2} \qquad (9)$$

TABLE IV
COMPARING VARIANTS OF THE PROPOSED METHOD WITH OTHERS IN TERMS OF MAE/RMSE OF $|\Delta Q|$ ESTIMATION

	Methods								
Test Dataset#	**SINS**	**IONet**	**VINet**	**AbolDeepIO**	**OriNet1**	**OriNet2**	**OriNet3**	**OriNet4**	**OriNet5**
1	0.0039/0.0052	0.0019/0.0025	0.0058/0.0063	0.0015/0.0020	5.98e-05/8.39e-05	6.15e-05/8.70e-05	5.42e-05/7.92e-05	0.0013/0.0026	**3.94e-05/0.000061**
2	0.0065/0.0635	0.0046/0.0637	0.0073/0.0639	0.0036/0.0636	**3.85e-05/7.87e-05**	5.25e-05/9.71e-05	4.28e-05/9.27e-05	0.00014/0.00024	5.1e-05/8.69e-05
3	0.0175/0.1445	0.0160/0.1450	0.0200/0.1458	0.0153/0.1447	0.00016/0.00024	0.00014/0.00023	**0.00014/0.00023**	0.00016/0.00028	0.00015/0.00024
4	0.0133/0.0945	0.0095/0.0948	0.0127/0.0951	0.0086/0.0943	0.00014/0.00024	0.00015/0.00025	0.00018/0.00029	**0.00014**/0.00028	0.00014/**0.00023**
5	0.0174/0.1723	0.0142/0.1725	0.0167/0.1725	0.0134/0.1725	8.29e-05/0.00010	9.60e-05/0.00012	7.40e-05/0.000099	0.00012/0.00015	**7.07e-05/9.33e-05**
Overall	0.0116/0.1136	0.0090/0.1138	0.0123/0.1141	0.0084/0.1136	9.38e-05/0.000159	9.79e-05/0.000165	9.49e-05/0.000172	0.00042/0.00130	**8.61e-05/0.000154**

where Δq_{w_i}, Δq_{x_i}, Δq_{y_i}, and Δq_{z_i} are sub-quaternions displacement in between two consecutive IMU readings. The MAE and RMSE of different networks on the test split are shown in Table IV; as the table shows, the proposed method can outperform the existing IO techniques, as well as inertial part of VINET, significantly. The analysis of the results and proposed modules come in the following.

1) Impact of Using the Previous State Quaternion: OriNet4 is trained from scratch without dropout, and its main differences with *AbolDeepIO*[4] are: 1) it benefits from the previous state quaternion (Q_{s-1}), 2) it outputs the ΔQ_{Net} which is an aggregation of various estimations for a particular time step. Comparing the results shows that, benefiting from the quaternion and aggregating the estimations with ΔQ_{Net} improves the performance of the network by 95%. This significant improvement is mainly because IMU measurements are in the IMU body frame, and passing the quaternion state, in the world frame, as another input to the network is critical. Having Q_{s-1} at the beginning of each local time window helps the network to transfer IMU measurements between world and IMU body frame, and correctly propagate measurements in order to estimate the quaternion in the world frame. It is essential to transfer IMU measurements between the world and IMU frames because, for instance, the IMU measurements regarding the yaw angle of the world frame are obviously different when IMU is at 90° and 0° (it is also different in between). Besides, similar to an EKF, the network can learn to perform an analysis and do the update process based on the previous state measurements and current values, by giving more weights to the previous states in some situations and learning to apply some filters. These facts are forgotten, eighter by deep learning-based IO or VIO methods like VINet [5], and the proposed approach can improve the performance of deep learning-based VIO systems significantly.

2) Impact of the Proposed Training Approach: OriNet4 is trained from scratch, without using the proposed training approach and dropout; on the other hand, *OriNet2* has the same configuration as *OriNet4*, but it is trained with the proposed training approach. It can be seen that the proposed training approach results in a 76% improvement in MAE. When the dropout is high, it is hard to reach the optimum loss and overfit; Hence, learning weights in the presence of high dropout at the first stages, results in a robust model which is not overfitted. However, having a high dropout value may avoid the network to reach its optimum result on validation and test set. Benefiting from the proposed method and reducing the dropout value once the model achieved a stationary state helps to avoid overfitting, and at the same time, it helps in reaching the optimum model, which is robust and gives the best output on the validation and test sets.

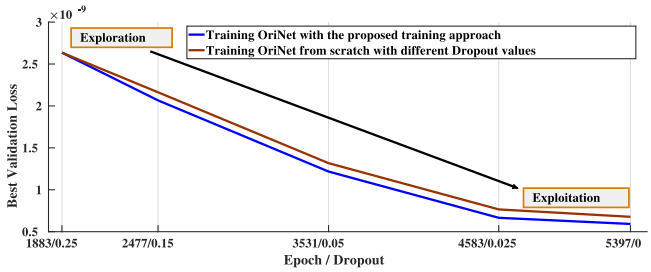


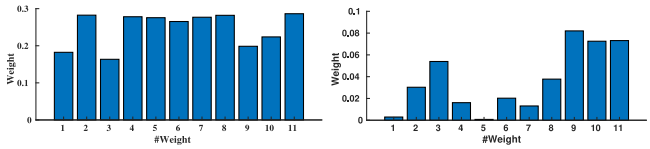
Fig. 4. Brown line shows the best validation loss obtained by training the network from scratch and using the dropout value indicated on the horizontal axis. The blue line shows the best validation loss achieved by training the network with the proposed training approach; The proposed training method reduces the dropout during training, and so the utilized dropout at each training epoch is indicated on the horizontal axis.

Fig. 4 is illustrating the best validation loss, during training the network with and without using the proposed training approach, and verifies our discussion on the effect of the proposed training approach. As can be seen, the proposed training approach cannot reach the optimum validation loss in the early stages, and it moves from exploration of the robust weights in the early stages to the exploitation of the weights in the later ones. Besides, comparing the validation loss of the proposed method, by the models trained from scratch with different dropout values, shows that the proposed method can achieve a better validation loss and distance between their validation loss increases by getting into the exploitation steps.

3) Importance of the ΔQ_{Net}, Average, and Weighting Gates: In order to analyze the effect of the three gates, *OriNet2* and *OriNet3* need to be compared in the beginning to see the impact of each of the ΔQ_{Net} and averaging gates. Comparing the overall result shows that using the averaging gate, as the final output, results in better performance, in terms of MAE. However, the RMSE of the ΔQ_{Net} is better. RMSE gives higher weight to the significant errors during averaging, and so the ΔQ_{Net} has less significant errors compared to average. As illustrated in Fig. 5, the most active neuron in ΔQ_{Net} gives high weight to almost all quaternion displacement estimations, which results in the near average estimate of all outputs, and other neurons are considering to perform a probabilistic analysis based on various estimations. Considering other neurons shows that the ΔQ_{Net} trusts on the estimations 9–11 more; such estimations are the ones estimated when there is a small distance between the start of the local time window and state estimation. For instance,

TABLE V
COMPARING ORINET FULL QUATERNION ESTIMATION CAPABILITY WITH IO AND VIO FRAMEWORKS. q_w, q_x, q_y, AND q_z COLUMNS REFER TO THE MAE IN EACH COORDINATE, $|Q|$ IS THE MEAN LENGTH OF THE ERROR, AND $\bar{\theta_D}$ IS THE MEAN DISTANCE BETWEEN THE ESTIMATED AND GROUND-TRUTH QUATERNIONS IN DEGREES

	VINS-MONO [1]						OriNet						Complementary Filter [10]						Madgwick et al. [9]					
Test Dataset #	q_w	q_x	q_y	q_z	$\|Q\|$	$\bar{\theta_D}$	q_w	q_x	q_y	q_z	$\|Q\|$	$\bar{\theta_D}$	q_w	q_x	q_y	q_z	$\|Q\|$	$\bar{\theta_D}$	q_w	q_x	q_y	q_z	$\|Q\|$	$\bar{\theta_D}$
1	**0.0092**	0.0817	0.0135	0.0647	0.1068	12.2640	0.0121	**0.0233**	**0.0073**	0.0339	**0.0447**	**5.1242**	0.0152	0.0333	0.0177	**0.0331**	0.0564	6.5376	0.2122	0.3234	0.3155	0.2209	0.5459	70.4467
2	0.0175	**0.0141**	**0.0270**	0.0174	**0.0438**	**5.0389**	**0.0083**	0.0167	0.0604	**0.0167**	0.0678	7.7748	0.0514	0.0715	0.0427	0.0530	0.1186	13.7461	0.2410	0.115	0.3537	0.0757	0.4611	54.0868
3	0.0304	0.0360	**0.0497**	0.0234	**0.0802**	9.2405	0.0381	0.0567	0.0632	0.0424	0.1152	13.2222	**0.0294**	**0.0345**	0.0472	**0.0230**	0.0955	**9.028**	0.2801	0.7115	0.6526	0.2119	1.0791	98.2662
4	**0.01435**	0.1460	**0.0180**	0.1567	0.2191	25.2119	0.0163	**0.0178**	0.0439	**0.0585**	**0.0836**	**9.5920**	0.2222	0.2586	0.3407	0.1772	0.5670	66.532	0.1398	0.2006	0.2071	0.2167	0.4404	51.1945
5	**0.0054**	**0.0083**	**0.0103**	**0.0141**	**0.0223**	**2.5728**	0.0123	0.0128	0.0215	0.0258	0.0436	5.0076	0.1677	0.1445	0.2322	0.1216	0.3653	42.3221	0.2493	0.2275	0.3371	0.1541	0.5296	62.7062
Overall	**0.0141**	0.0574	**0.0218**	0.0549	0.0926	10.6616	0.0166	**0.0243**	0.0353	**0.0350**	**0.0670**	**7.6961**	0.0970	0.1069	0.1359	0.0817	0.2386	27.4691	0.2237	0.3089	0.3632	0.1791	0.5977	67.0723



(a) The most active neuron (b) Second most avtive neuron

Fig. 5. Weights of most active neurons in ΔQ_{Net} input space.

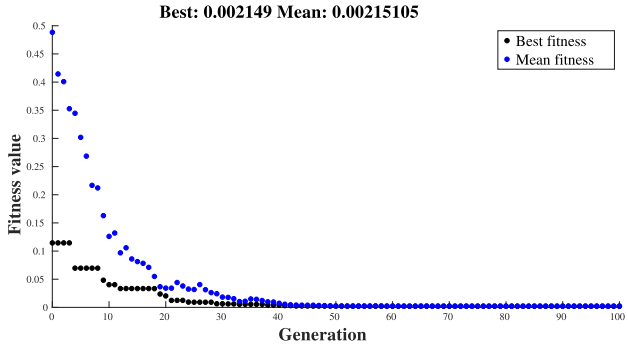


Fig. 6. Best and mean fitness value for the test dataset #4.

estimation 11 is for the time that the window starts right at the time of measurement ($\Delta Q_{t=1}$ considers from the window), and for the estimation 10, the window starts from one measurement behind the state we are considering ($\Delta Q_{t=2}$ considers from the window).

Weighting gate helps to benefit from the output of the Average and ΔQ_{Net} gates simultaneously and reach less error with the small effect of significant errors. Comparing the results of *OriNet5* with *OriNet1* shows the considerable impact of the weighting gate, compared to using ΔQ_{Net}, which results in 8% improvement in MAE. Besides, the network has the least RMSE. It is essential to notice that, when the weighting gate is on, the ΔQ_{Net} and averaging gates also need to be on, because the weighing gate uses the output of these two gates; but at the end, the output of the weighting gate only considers as the final output.

4) Importaance of OriNet-IMU Bias Calibration: OriNet1 and *OriNet2* have a same configuration, but the *OriNet1* performs bias calibration at the beginning of each test set. Comparing the results of *OriNet1* and *OriNet2* demonstrates a 4% improvement, by performing bias calibration, which shows the importance of this module for estimating the bias of IMU in the test time. Fig. 6 illustrates the fitness value achieved in different generations, and as it can be seen, the optimum value is almost achieved at $50th$ generation. The results show that the biases are in the range of $[-0.0019, 0.0027]$ in all test sets, and so it is possible to change the search domain to $[-0.005, 0.005]$ and find biases faster (within 20 generations).

C. Comparing Full Quaternion Estimation Capability of OriNet With IO and VIO Frameworks

This section compares the performance of the proposed method with the VINS-MONO [1] VIO method and two other IO techniques (Madgwick *et al.* [9] and Complementary Filter [10]), which are capable of extracting the full 3D quaternion along time. Estimations at times that image is captured and VINS-MONO outputs the orientation are only considered. The MAE of estimating orientation in each of the quaternion subspaces, along the moving path, is shown in Table V. As it shows, the proposed method can outperform IO methods significantly. Besides, considering $|Q|$, which demonstrates the averaged length of error, shows that the proposed method can approximately outperform VINS-MONO by 30%. Furthermore, the quaternion distance (rotation required to get from one quaternion to another) between two quaternions p and q is also defined as [19]

$$\theta_D = 2\arccos(|< p, q >|) \tag{10}$$

where

$$< p, q >= p_1 * q_1 + p_2 * q_2 + p_3 * q_3 + p_4 * q_4 \tag{11}$$

and $|.|$ is the modulus function. The averaged quaternion distance between estimated and Ground Truth quaternion over each test dataset ($\bar{\theta_D}$) is computed and reported. As shown in Table V, the proposed method has achieved the least quaternion distance compared to others, and it beats the VINS-MONO by approximately 28%. The Estimated and Ground Truth quaternion of the test dataset #4 is shown in Fig. 7. As it can be seen, $OriNet$ can perform robustly during high motions on q_x and q_z estimations, while the VINS-MONO over-fits and under-fits the estimations.

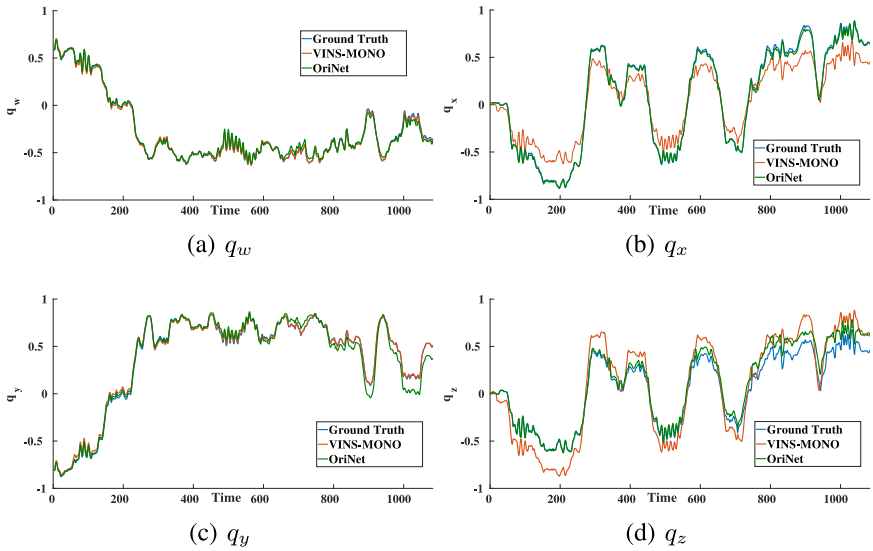


Fig. 7. Estimated and exact quaternion of test dataset #4.

V. Conclusions

This letter proposed a deep learning-based framework for estimating the 3D orientation of flying robots in quaternion coordinates. The proposed method is compared with existing visual-inertial navigation systems, as well as inertial methods. The results show the notable power of the proposed method compared to existing approaches. The proposed framework benefits from the previous state information and aggregates various estimations achieved for each state, while shifting the local time window, to find the output robustly. Besides, a novel training approach is used, which forces the network to do the exploration of the best robust weights, which are not overfitted, in the first training stages, and move to the exploitation of the weights in later stages. A Genetic Algorithm based approach is also proposed to calibrate trained network with the IMU bias, which can be improved and utilized in other applications. The proposed method can significantly improve the performance of the existing deep learning-based visual-inertial navigation systems. Since deep learning shows its promising behavior in robust orientation estimation, with a single IMU, in the future, the best model for utilizing IMU for position estimation needs to be investigated, to see deep learning capability in retrieving position from a single particular IMU. Moreover, the generalization capability of the proposed framework when a new IMU utilizes, and the IMU frequency and noise model changes, should be investigated, and an approach which makes a trained network capable of being used with other IMUs have to be studied.

References

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

[2] S. Hecker, D. Dai, and L. Van Gool, "End-to-end learning of driving models with surround-view cameras and route planners," in *Proc. Eur. Conf. Comput. Vision*, 2018, pp. 435–453.

[3] P. G. Savage, "Strapdown inertial navigation integration algorithm design part 1: Attitude algorithms," *J. Guid., Control, Dyn.*, vol. 21, no. 1, pp. 19–28, 1998.

[4] M. A. Esfahani, H. Wang, K. Wu, and S. Yuan, "AbolDeepIO: A novel deep inertial odometry network for autonomous vehicles," *IEEE Trans. Intell. Transp. Syst.*, doi: 10.1109/TITS.2019.2909064.

[5] R. Clark, S. Wang, H. Wen, A. Markham, and N. Trigoni, "Vinet: Visual-inertial odometry as a sequence-to-sequence learning problem," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017.

[6] M. G. Petovello, "Real-time integration of a tactical-grade IMU and GPS for high-accuracy positioning and navigation," Ph.D. dissertation, University of Calgary, Calgary, AB, Canada, 2003.

[7] I. Skog, J.-O. Nilsson, and P. Händel, "Evaluation of zero-velocity detectors for foot-mounted inertial navigation systems," in *Proc. IEEE Int. Conf. Indoor Positioning Indoor Navigat.*, 2010, pp. 1–6.

[8] A. Solin, S. Cortes, E. Rahtu, and J. Kannala, "Inertial odometry on handheld smartphones," in *Proc. IEEE 21st Int. Conf. Inf. Fusion*, 2018, pp. 1–5.

[9] S. O. Madgwick, A. J. Harrison, and R. Vaidyanathan, "Estimation of IMU and MARG orientation using a gradient descent algorithm," in *Proc. IEEE Int. Conf. Rehabil. Robot.*, 2011, pp. 1–7.

[10] R. Valenti, I. Dryanovski, and J. Xiao, "Keeping a good attitude: A quaternion-based orientation filter for IMUS and MARGs," *Sensors*, vol. 15, no. 8, pp. 19 302–19 330, 2015.

[11] C. Chen, X. Lu, A. Markham, and N. Trigoni, "IONet: Learning to cure the curse of drift in inertial odometry," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018.

[12] C. Chen *et al.*, "Motiontransformer: Transferring neural inertial tracking between domains," in *Proc. AAAI Conf. Artif. Intell.*, 2019, vol. 33, pp. 8009–8016.

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

[14] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "IMU preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation," Georgia Inst. Technol., Atlanta, GA, USA, Rep. GT-IRIM-CP&R-2015-001, 2015.

[15] M. Abolfazli Esfahani, K. Wu, S. Yuan, and H. Wang, "From local understanding to global regression in monocular visual odometry," *Int. J. Pattern Recognit. Artif. Intell.*, p. 2055002, doi: 10.1142/S0218001420550022.

[16] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[17] 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.

[18] J. Delmerico and D. Scaramuzza, "A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2018, pp. 2502–2509.

[19] D. Q. Huynh, "Metrics for 3D rotations: Comparison and analysis," *J. Math. Imag. Vision*, vol. 35, no. 2, pp. 155–164, 2009.