IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received February 4, 2021, accepted February 17, 2021, date of publication March 1, 2021, date of current version March 9, 2021.

*Digital Object Identifier 10.1109/ACCESS.2021.3062817*

# Understanding the Behavior of Data-Driven Inertial Odometry With Kinematics-Mimicking Deep Neural Network

**QUENTIN ARNAUD DUGNE–HENNEQUIN[1], HIDEAKI UCHIYAMA[1], (Member, IEEE), AND JOÃO PAULO SILVA DO MONTE LIMA[2], (Member, IEEE)**

[1]Central Library, Kyushu University, Fukuoka 819-0395, Japan
[2]Departamento de Computação, Universidade Federal Rural de Pernambuco, Recife 52171-900, Brazil

Corresponding author: Hideaki Uchiyama (uchiyama@limu.ait.kyushu-u.ac.jp)

This work was supported in part by JSPS KAKENHI Grant Number JP20K11891 and JP18H04125.

**ABSTRACT** In navigation, deep learning for inertial odometry (IO) has recently been investigated using data from a low-cost IMU only. The measurement of noise, bias, and some errors from which IO suffers is estimated with a deep neural network (DNN) to achieve more accurate pose estimation. While numerous studies on the subject highlighted the performances of their approach, the behavior of data-driven IO with DNN has not been clarified. Therefore, this paper presents a quantitative analysis of kinematics-mimicking DNN-based IO from various aspects. First, the new network architecture is designed to mimic the kinematics and ensure comprehensive analyses. Next, the hyper-parameters of neural networks that are highly correlated to IO are identified. Besides, their role in the performances is investigated. In the evaluation, the analyses were conducted with publicly-available IO datasets for vehicles and drones. The results are introduced to highlight the remaining problems in IO and are considered a guideline to promote further research.

**INDEX TERMS** Inertial odometry, dead reckoning, deep neural network, kinematics, navigation, IMU.

## I. INTRODUCTION

Odometry, which is the process to compute positional displacements between two sequential moments, is a fundamental functionality for any applications associated with motion. For instance, an autonomous robot needs to be localized in a target environment to achieve a specific task while avoiding obstacles [1]. Also, consumer locations in a shopping mall are tracked for location-aware services and analysis of consumer behaviors [2]. For those applications, an odometry technique is utilized to estimate the position by accumulating relative movements continuously. In navigation, odometry is referred to as dead reckoning (DR) [3]. Especially, pedestrian dead reckoning (PDR) is the technique specialized for tracking pedestrians by using the walking specificities [4], [5]. For a robot and a vehicle, the amount of wheel rotations coming from sensor reading is accumulated with a known radius to compute the movement [1]. As a generalized approach for any movable target in three-dimensional (3D) space, the odometry can be computed by only using cameras attached to the target, as referred to as visual odometry (VO) [6]. Since the localization based on an odometry technique suffers from errors accumulation, another supplemental equipment such as GNSS, Wi-Fi, and landmarks providing global locations are used to suppress the error [7]–[9].

An inertial measurement unit (IMU), which measures acceleration and angular velocity at high rates, is the device utilized for odometry techniques, as referred to as inertial odometry (IO) [10]. IMU-based micro electro mechanical systems (MEMS) is cheap and is installed on various devices such as a smartphone [11]. Compared with other devices, external equipment such as a GNSS satellite and a Wi-Fi base station are not required. In other words, IMU is useful for a stand-alone odometry system without any preparation. Besides, sensor reading and its processing are not affected by its surrounding conditions, such as lighting changes and dynamic environments, which generally degrade the stability of VO. Traditionally, IO has been referred to as inertial navigation system (INS) [12]. The theory is based on kinematics. Acceleration is integrated to compute velocity, and

The associate editor coordinating the review of this manuscript and approving it for publication was Muhammad Asif.

then the velocity is integrated to compute distance. Since this odometry technique is sensitive to sensor noise and bias in practice, one of the research issues has been error suppression by designing reasonable constraints [13]. For instance, foot-mounted IMU-based PDR typically uses the velocity constraint of walking because the speed of a foot becomes zero when touching the ground [14]. This constraint is specifically referred to as zero velocity update (ZUPT) [15]. For a vehicle, the constraint is designed such that the vehicle does not jump [16].

In computer vision and natural language processing, data-driven machine learning techniques with neural network (NN) have shown superior performances for many tasks, compared with classical techniques such as support vector machine (SVM) [17]. The performances are achieved using the functions representing highly-nonlinear phenomena with numerous parameters, which are optimized with massive training data, namely deep NN (DNN). Since it is not easy to model such phenomena heuristically, the model is generated from the data with some optimization techniques in DNN. In navigation, DNN-based IO has recently been investigated. For instance, step detection, step length estimation and heading estimation are performed by using long short-term memory (LSTM) and its variants in PDR [18]–[23]. For vehicle odometry, velocity and heading are estimated with DNN [24], [25]. Generally, the primary purpose of those papers was to outperform other classical methods in terms of accuracy with some specific datasets. However, the reported results in those papers generally show only some aspects of DNN-based approaches. As discussed in other tasks [26]–[29], it is essential to investigate the behaviors of DNN-based IO in terms of both research issues and network architectures.

This paper presents a quantitative analysis to clarify the behavior of DNN-based IO. Contrary to other works, the purpose of this paper is to investigate in detail the correlation between DNN hyper-parameters and their role in the performances rather than outperforming existing literature. To the extent of our knowledge, such investigation is the first one in the literature of DNN-based IO. In order to do that, we propose an end-to-end network architecture that follows basic kinematic models. The network is specially designed with LSTM layers to estimate the orientation and velocity from sequential inputs of acceleration and angular velocity. As a preliminary evaluation, we investigate the impact of data distribution and several kinematic approaches to determine the baseline network to be investigated. Next, a list of hyper-parameters to be considered when designing DNN-based IO is proposed. A unique qualitative and quantitative evaluation is performed for each hyper-parameter that is highly correlated to IO problems. The performances were mainly evaluated with KITTI dataset as vehicle odometry. Finally, new perspectives for the future are opened.

In summary, our contributions are described as follows.

- A new DNN architecture that mimics the kinematics is introduced (Section III);
- Preliminary and comprehensive analyses of the data distribution and the variants of our kinematic approach are proposed (Section IV, V);
- A list of DNN hyper-parameters to be investigated is established (Section VI). Their influence on the performances is investigated and discussed (Section VII, VIII).

The results of our analyses were informative to clarify the remaining problems to be solved in DNN-based IO approach. Besides, they will support the design and the evaluation of future architecture as a guideline.

## II. RELATED WORK

The mechanical frameworks of IMU are divided into a stable platform and a strap-down one [30]. The stable platform was first developed based on gimbals. The motion of the platform is determined by computing the difference between the inner gimbal and the outer one. The strap-down platform, which uses Sagnac effect or Coriolis force to compute angular velocity, was then developed because its equipment configuration can be smaller than the stable platform one. Several architectures of the strap-down platform have been proposed, such as ring laser gyro (RLG), fiber optic gyro (FOG), and MEMS [31], [32]. Since MEMS is the most compact and cheapest of all, it has been installed on various devices, such as smartphones, cameras, and vehicles, to compute device orientation. However, the sensor signal from MEMS generally contains noise and bias caused by many factors such as small vibrations and temperatures. Therefore, it is required to investigate the processing techniques to suppress such undesirable components [33]. Recently, additional researches on fluid-based inertial sensors and Hemispherical resonator gyroscope achieved a minor bias instability and low-cost manufacturing, respectively [34], [35].

The approach based on double-integrating linear acceleration is the most straightforward to achieve IO [33], [36]. However, the most common and inexpensive MEMS suffers from sensor noise and biases in sensor reading, making odometry estimation drifts over time. The statistical formulation of the error phenomenon had been investigated to solve this issue [37]. Also, other sensors such as Magnetometer and GPS are fused with extended Kalman filtering (EKF) as observation or measurement [38]–[40]. In recent years, methods relying on cameras [41]–[43] show accurate tracking in highly-textured environments owing to the detection stability of visual features, as referred to as visual-inertial odometry. However, camera-based approaches generally face energy-consuming and processing-demanding problems. Also, they cannot be utilized in a dark environment because nothing can be observed. When IMU is the only sensor in the odometry computation, some motion and trajectory constraints, such as ZUPT, stationary state, and loop closure, are used in the filtering based approaches [44]–[46].

Classically, machine learning techniques were used for IO and PDR. For instance, phone velocity attached to a pedestrian was estimated using a support vector regression after classifying the location of the phone attachment with SVM [47]. With advances in theories and computational resources, DNN-based methods have recently renewed IO. Overall, they are classified into two approaches. The first approach relies on existing filtering-based approaches where a DNN is used to compute pseudo-measurements or motion constraints. For instance, recurrent NN (RNN) was leveraged to classify the motion profile of wheeled robots [48], [49]. It was later extended to dynamically estimate the covariance of pseudo-measurement by using convolutional NN (CNN) [16]. In another work, CNN was used to constrain the velocity in the Kalman filtering for PDR [50]. TLIO used residual network (ResNet) to compute displacements and their uncertainty and then integrated them into EKF [51]. The second approach relies on DNN to compute the odometry in an end-to-end manner without filtering-based approaches. IONet used RNN to predict the movements from sequential IMU data [52], [53]. The Degree of Freedom (DoF) of motion was restricted to 2D because it focused only on 2D planar trajectories. Similarly, several DNN architectures were investigated to predict the velocity and orientation of a pedestrian, such as 1D ResNet [54], LSTM, and temporal convolutional network (TCN) in [55]. The magnitude of relative translation and rotation in the 3D space was computed in AbolDeepIO [25] where orientation prediction was later extended with OriNet [56]. Six-DoF odometry was achieved by using the combination of convolution and LSTM with a weighting scheme to ensure the balance of multiple metrics [57]. Nevertheless, these approaches were not designed from the existing knowledge of inertial navigation that is introduced by kinematics. To the best of our knowledge, there is no kinematic approach of DNN-based IO. This proposed approach is used as a support to provide comprehensive analyses of the data-driven DNN-based IO rather than demonstrating performances, conversely to other works.

## III. PRELIMINARY

### A. KINEMATIC MODEL

To explain the issues to be tackled in this paper, the kinematic model used in IO is first reviewed [33]. This subsection is the introduction to understand the relationship between the standard kinematic model and our DNN architecture.

The angular velocity and acceleration provided by IMU are subjected to bias and noise based on some sensor properties. Assuming that Coriolis acceleration and earth rotation effect are negligible, the model of IMU measurement can be written as follows:

$$\begin{cases} \mathrm{w}_t^{IMU} = \mathrm{w}_t + \boldsymbol{\delta}_t^w + \mathrm{n}_t^w \\ \mathrm{a}_t^{IMU} = \mathrm{a}_t + \boldsymbol{\delta}_t^a + \mathrm{n}_t^a \end{cases} \tag{1}$$

where $\boldsymbol{\delta}_t^w$, $\boldsymbol{\delta}_t^a$ are time-varying bias, and $\mathrm{n}_t^w$, $\mathrm{n}_t^a$ are noises for the angular velocity $\mathrm{w}_t$ and the acceleration $\mathrm{a}_t$, assuming that they follow a zero-mean Gaussian distribution. The biases are then modeled as a random walk given by

$$\begin{cases} \boldsymbol{\delta}_t^w = \boldsymbol{\delta}_{t-1}^w + \mathrm{n}_{t-1}^{\delta w} \\ \boldsymbol{\delta}_t^a = \boldsymbol{\delta}_{t-1}^a + \mathrm{n}_{t-1}^{\delta a} \end{cases} \tag{2}$$

where $\mathrm{n}_t^{\delta w}$, $\mathrm{n}_t^{\delta a}$ are zero-mean Gaussian noises.

Finally, the following kinematic model is adopted to compute the odometry by using relative orientation $\Delta\mathrm{q}_t$ and relative velocity $\Delta\mathrm{v}_t$ between two discrete instants $dt$:

$$\begin{cases} \Delta\mathrm{q}_t = exp(\frac{dt}{2}\mathrm{w}_{t-1}) & \text{(3a)} \\ \Delta\mathrm{v}_t = (\mathrm{R}(\mathrm{q}_{t-1}^{IMU})\mathrm{a}_{t-1} + \mathrm{g})dt & \text{(3b)} \\ \mathrm{q}_t^{IMU} = \mathrm{q}_{t-1}^{IMU} \otimes \Delta\mathrm{q}_t & \text{(3c)} \\ \mathrm{v}_t^{IMU} = \mathrm{v}_{t-1}^{IMU} + \Delta\mathrm{v}_t & \text{(3d)} \\ \mathrm{p}_t^{IMU} = \mathrm{p}_{t-1}^{IMU} + \mathrm{v}_{t-1}^{IMU}\,dt & \text{(3e)} \end{cases}$$

where $\mathrm{q}_t^{IMU}$ is the quaternion representing the orientation in the world frame. The world frame is generally defined such that one axis is parallel to the gravity g and two other axes are determined according to the initial orientation of the IMU frame. $\mathrm{v}_t^{IMU}$ is the velocity, and $\mathrm{p}_t^{IMU}$ is the position at time $t$ in the world frame, respectively. $\otimes$ characterizes the Hamilton product between two quaternions. R is the rotation matrix computed from q. *exp* refers to the approximation expressed below:

$$exp(\boldsymbol{\eta}) = \begin{pmatrix} 1 \\ \boldsymbol{\eta} \end{pmatrix} \tag{4}$$

with $\boldsymbol{\eta} \in \mathbb{R}^3$ under the assumption that $\boldsymbol{\eta}$ is small.

In IO, the bias and noise represented in (1) and (2) are harmful in the double integration process (3), leading to the accumulative drift of the error over time [58]. For the solutions, some DNN-based approaches have been proposed to simultaneously estimate these factors and eliminate their impacts.

### B. ARCHITECTURE OF OUR BASELINE NETWORK

The purpose of this paper is to understand the behavior of data-driven DNN-based IO. Especially, our end-to-end DNN architecture is newly designed to mimic standard IO represented in (3) in terms of the data flow and the modules in the architecture. This architecture is the closest to the kinematic definition. Therefore, it should offer a better comprehension of the IO problem.

Fig. 1 illustrates the architecture of our baseline network for absolute kinematic regression. Our network is based on RNN that is suitable for modeling the temporal and dynamic behavior of IO sequences, as introduced in [55]. The inputs of our network are $N$ sequential past accelerations, angular velocities, previous velocities, previous orientations, and timestamp intervals between sampled data. Instead of using the previous data at a specific moment, we use $N$ sequential past data because they can help to improve the performances of DNN. This part is slightly different from the mathematical

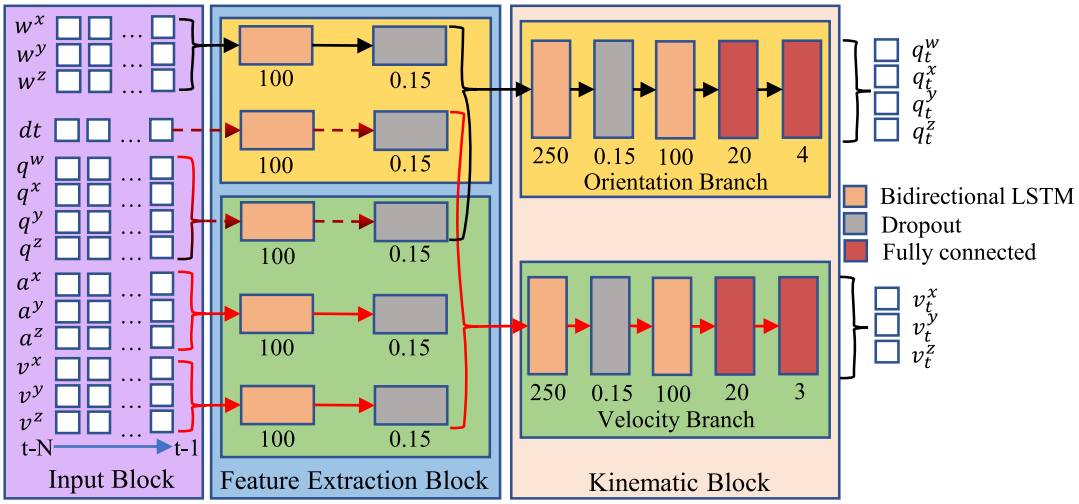



**FIGURE 1.** Network architecture for absolute kinematic regression. This architecture outputs three-axis velocity v and unit quaternion q at time $t$. From each input, the feature is separately extracted in the feature extraction block. Then, each feature is concatenated and fed into two parallel networks respecting the kinematic model: the orientation branch and the velocity branch. Features used for velocity regression are connected with a red arrow, whereas features used for orientation regression are connected with black arrows. When the features are used for both regression problems, a dark-red and dotted arrow is used. The number of units and dropout rate are shown below each layer.

kinematic model in (3). Then, the network architecture is divided into two main blocks. One is referred to as the feature extraction block that extracts some latent features from each input data, individually. In this block, it is expected that noise and bias represented in (1) are implicitly modeled, or the impact of other undesirable phenomena is suppressed. The features are further fed into the second block, defined as the kinematic block. This block comprises two parallel networks: the orientation branch and the velocity branch, that compute absolute quaternion and velocity, respectively. Compared with other DNN architectures [55]–[57], our motivation is to compute the odometry from IMU data sequence in a similar approach with a kinematic model presented in (3), where orientation and velocity are related to each other but are computed separately. Since the data flow in the network is designed according to the kinematic model, this can be considered as a baseline network that mimics the absolute kinematic model.

The details of the architecture are designed as follows. In the feature extraction block, each input is separately processed in a bidirectional LSTM layer with dropout to extract the feature specific to each input data. Then, the features are concatenated into two unique vectors that are fed into the orientation and velocity branches in the kinematic block, respectively. This block follows the kinematic model defined in (3) to compute the orientation and velocity individually. In the orientation branch, the features coming from three-axis angular velocity $\mathrm{w} = (w^x, w^y, w^z)$, timestamp interval $dt$, and previous four-dimensional quaternion $\mathrm{q} = (q^w, q^x, q^y, q^z)$ are considered. In the velocity branch, the features coming from $dt$ and q are concatenated with those of acceleration $\mathrm{a} = (a^x, a^y, a^z)$ and previous three-axis velocity $\mathrm{v} = (v^x, v^y, v^z)$. Each branch is made of two-stacked bidirectional LSTM layers with dropout. The last LSTM of each branch is designed in a many-to-one manner. Thus, the feature vector from the last cell state is extracted and fed into two fully connected layers in charge of regressing the output at time $t$.

Since our architecture is motivated by [56], we mainly follow the parameters of their architecture as follows. Each dropout rate is set as 15 percent. In the feature extraction block, the LSTM layer is made of 100 units. The two LSTM layers from the orientation and velocity branches are made of 250 and 100 units, respectively, whereas the first linear layer contains 20 neurons and the second one contains four or three neurons according to the regression problem. After experimenting, this configuration provided satisfying results for the purpose of this paper. Nonetheless, the proximity with parameters introduced in [56] aims to have a comparative architecture. From an optimization point of view, additional works are required according to the data used, the training process adopted and other elements.

### C. MOTIVATION OF THE ARCHITECTURE

Next, the motivation of the architecture in Fig. 1 is explained. In Karpathy's work [59], three different connectivity patterns in the network architecture have been introduced: early, late and slow fusion. Translated to the regression problem for IO, these definitions slightly differ.

In the early fusion stage, the input features are extracted from a common layer. For 1D convolution and LSTM, features are extracted from the inputs and are then summed up into a single and unique vector. Therefore, the features coming from IMU, quaternion, timestamp interval, and velocity are merged into the same and unique value for the regression. This approach was adopted by [16], [50], [52], [55]

where IMU data is simultaneously fed into a single and same layer.

In the late fusion stage, each input is first separated into parallel layers. Then, every feature is concatenated into a common vector before getting through other layers. The same approach was adopted in [56], [57] where inputs were separated into LSTM and CNN layers, respectively, before merging the features into one vector and fed into LSTM layers.

The slow fusion uses a more comprehensive combination of inputs by gradually fusing extracted features to learn higher-level ones. Although better performances have been obtained with this pattern in the micro-expression recognition field, this process remains slow and memory-consuming.

The early, late, and slow fusions are the three main conventions for features fusion when dealing with the structure of DNN. From these definitions and our understanding of kinematics in (3), our new architecture is proposed for IO. The kinematics is considered during the process of features fusion, as illustrated in Fig. 1. Similarly to the late fusion, each input data is first processed into a singular LSTM to extract low-level features. Then, each feature is concatenated according to the comprehension of the kinematic model and fed into their respective branch to output either orientation or velocity. In other words, the fusion process of orientation and velocity branches is differently designed according to the kinematic model. Our new design for DNN-based IO can offer new possibilities for future perspectives where each parallel network can be replaced with a fine-tuned model that has been pre-trained for a single output of relative kinematics.

### D. DATASET

The experiments in this paper were performed by using two public datasets: KITTI dataset [60] for vehicle odometry and EuRoC [61] for drone odometry. The main difference between these datasets is the DoF of the motion. Since a vehicle runs on the ground, overall motions can be represented with three-DoF containing 2D displacement and 1D orientation. The motion of a drone moving arbitrarily in 3D space is represented with six-DoF. To start the analysis with a simpler motion, we mainly used the KITTI dataset.

The details of the sequences used in our analysis are as follows. Similarly to [16], 38 sequences at 100 Hz frequencies in the KITTI dataset were selected as described in Table. 1. The recording duration of each sequence varies from 27 seconds to 8.4 minutes. In total, the duration time of the dataset is close to one hour. One important processing for the dataset is as follows. In the dataset, it was observed that some values were duplicated with the same timestamp, causing a time jump in the sequence for a short amount of time such as drive 36 in Table. 1. This kind of error in the dataset is harmful to data-driven approaches because they should not be used as ground truth. To solve this issue, duplicated values were simply deleted and replaced by a linear approximation for acceleration, angular velocity, velocity, position, and Euler angles. Nevertheless, the sequence was cut in half, and the erroneous data part was not chosen for the training and validation process when the time jump was more than 100 milliseconds. Sometimes, such error occurred several times along a single and same track, leading to a detachment in several distinct parts. Therefore, 72 sequences were generated from the 38 original ones. Then, 46 ones were used for the training process whereas the 26 remaining ones were used for the validation process, as described in Table. 1. This split was designed according to the data distribution through a preliminary evaluation, as explained in Section IV.

**TABLE 1. Training and validation split for KITTI dataset. Due to the timestamp issue, some sequences have been cut into several parts. The bold numbers refer to the number of splits the sequence had. Each part was either used for training or validation dataset. For example, drive 27 (03/10) was separated into seven parts. Five of them were used for training and the remaining ones for validation.**

| Training Data | Training/Validation | Validation Data |
|---|---|---|
| **1** - drive 14 (26/09) | **1/1** - drive 9 (26/09) | **1** - drive 15 (26/09) |
| **1** - drive 18 (26/09) | **3/1** - drive 28 (26/09) | **1** - drive 22 (26/09) |
| **1** - drive 19 (26/09) | **1/1** - drive 29 (26/09) | **1** - drive 56 (26/09) |
| **1** - drive 23 (26/09) | **1/2** - drive 64 (26/09) | **1** - drive 59 (26/09) |
| **1** - drive 32 (26/09) | **1/1** - drive 84 (26/09) | **2** - drive 101 (26/09) |
| **3** - drive 36 (26/09) | **2/3** - drive 86 (26/09) | **1** - drive 4 (29/09) |
| **1** - drive 39 (26/09) | **1/1** - drive 91 (26/09) | **1** - drive 20 (30/09) |
| **1** - drive 51 (26/09) | **1/1** - drive 96 (26/09) | |
| **1** - drive 57 (26/09) | **2/2** - drive 104 (26/09) | |
| **2** - drive 95 (26/09) | **1/2** - drive 117 (26/09) | |
| **3** - drive 71 (29/09) | **5/2** - drive 27 (03/10) | |
| **1** - drive 16 (30/09) | **2/1** - drive 34 (03/10) | |
| **2** - drive 18 (30/09) | | |
| **1** - drive 27 (30/09) | | |
| **1** - drive 28 (30/09) | | |
| **1** - drive 33 (30/09) | | |
| **1** - drive 34 (30/09) | | |
| **1** - drive 42 (03/10) | | |
| **1** - drive 47 (03/10) | | |

To extend our first analysis, we also used the EuRoC dataset. The dataset is made of 11 sequences at 200 Hz for a total recording time of approximately 23 minutes. Similarly to [56], [57], the dataset was split into training and validation datasets. The training dataset was made of six sequences, and the validation one contained five sequences, as described in Table. 2. We followed the split way used in [56], [57]. As aforementioned, our analysis was mainly performed with the KITTI dataset due to its simpler motion. Therefore, the analysis with the EuRoC dataset was performed only for some sections in this paper to provide some perspectives for six-DoF DNN-based IO.

Through the analysis in this paper, we use the terminology of a learning process to train and validate a DNN model. In the learning process, we use the training dataset, whereas the validation dataset is used to regulate the training process to avoid overfitting. Generally, the test dataset is used to compare the performances of a new DNN with others as a third independent dataset. Since the purpose of our paper is to analyze the potential use of a DNN-based approach itself for the IO problems, the analysis was performed by using

**TABLE 2. Training and validation splits for EuRoC dataset. We followed the split way in [56], [57].**

| Training Data | Validation data |
|---|---|
| MH_01_easy | MH_02_easy |
| MH_03_medium | MH_04_difficult |
| MH_05_difficult | V1_03_difficult |
| V1_02_medium | V2_02_medium |
| V2_01_easy | V1_01_easy |
| V2_03_difficult | |

both training and validation datasets after a DNN model was trained. We refer to this as the testing process. The analysis with the validation dataset is equivalent to the one with the test dataset in general.

## IV. PRELIMINARY EVALUATION ON IMPACT OF DATA DISTRIBUTION

First, the impact of data distribution is investigated. Since it affects all of the other experiments, this was independently performed as a preliminary evaluation.

This evaluation was designed due to the hypothesis that a velocity range cannot be predicted if the data in such range was not presented in the learning process. The performances of our network were analyzed to confirm the hypothesis through two different learning approaches, detailed as follows. In the first approach, we trained and validated the network in Fig. 1 by using all the data as referred in Table. 1. We refer to this approach as normal learning. In the second approach, the same network architecture was trained and validated without using samples with the norm of a ground truth velocity within the range of $[4, 7[$ in both training and validation datasets. Thus, only samples with a velocity norm within the range of $[0, 4[\cup[7, \infty[$ were selected for the second approach. We refer to this approach as missing-range learning. In this evaluation, the behavior of our network for the IO problem was investigated when an output range of values is missing.

For the two approaches, the testing process with the all-range data was performed to investigate the performances of missing-range learning. In particular, two sequences from the validation dataset such as Track 1 and Track 2 were used to analyze the performances of each learning approach, as illustrated in Fig. 2. In both approaches, the prediction was globally similar to the ground truth within the range of $[4, 7]$ even though one of them did not use this specific range of values in the learning process. However, the result of the missing-range learning was sometimes low, as illustrated in Fig. 2b.

To compare the overall performances between the two approaches, we computed the Root Mean Square Error (RMSE) between velocity norm and its ground truth, for both ranges. The results of four different tracks are referred in Table. 3. Along this paper, we will refer to testing tracks as introduced in Table. 4. For Track 1, Track 2 and Track 4, the best accuracy was obtained with the normal learning approach. Especially, the difference in accuracy within the missing-range was higher, proving the importance of data distribution in the learning process. On the other hand, the observation differs for Track 3 where the error for the missing-range learning is lower than the error for the normal one. Since Track 3 is included in the training dataset, this phenomenon can be considered as overfitting for the non-missing range where the network learned a model according to the distribution of learning data, leading to the deterioration of generalization capability.

**TABLE 3. Comparison between normal and missing-range learning. The RMSE between the velocity norm and its ground truth was computed for two different learning approaches: normal learning and missing-range one. In both approaches, the dataset was split as described in Table. 1. The best results are indicated with bold type in green cells. Note that Track 3 did not have a velocity norm within the missing-range.**

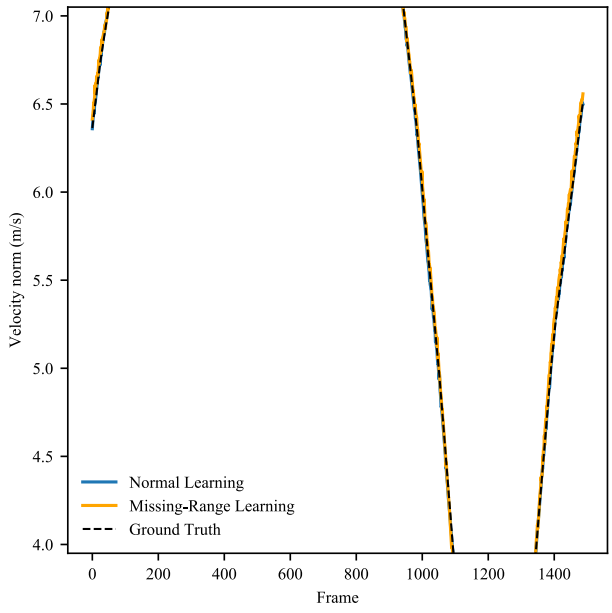
| | $[4, 7[$ | | $[0, 4[\cup[7, \infty[$ | |
|---|---|---|---|---|
| Learning / Tracks | Normal | Missing | Normal | Missing |
| Track 1 | **2.71e-2** | 5.22e-2 | **1.75e-2** | 2.72e-2 |
| Track 2 | **1.89e-2** | 4.79e-2 | **2.01e-2** | 2.40e-2 |
| Track 3 | X | X | 4.43e-2 | **3.78e-2** |
| Track 4 | **2.19e-2** | 5.39e-2 | **1.99e-2** | 2.21e-2 |

**TABLE 4. Testing tracks of the Kitti Dataset.**

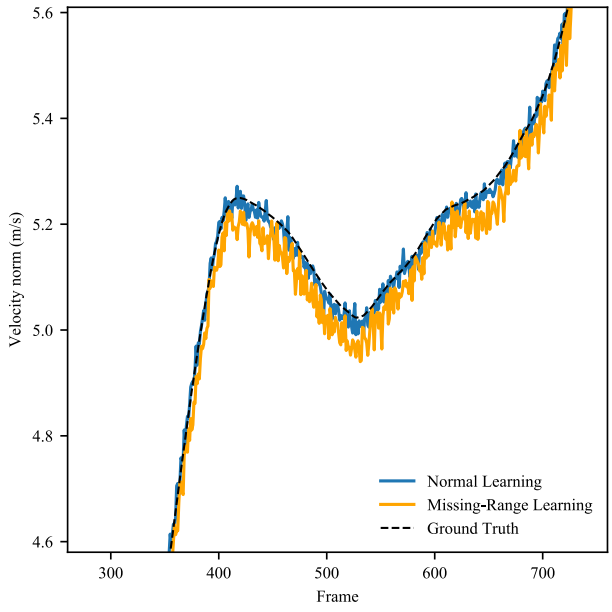
| Tracks | Sequence | Range | Dataset |
|---|---|---|---|
| Track 1 | drive 27 (03/10) | [21309,22807] | Validation |
| Track 2 | drive 64 (26/09) | [0,2254] | Validation |
| Track 3 | drive 32 (26/09) | [0,4094] | Training |
| Track 4 | drive 18 (30/09) | [0,15660] | Training |
| Track 5 | drive 71 (29/09) | [3257,6653] | Training |

Through this analysis, the importance of the data distribution in the training and validation datasets was clarified. The conclusion is reasonable but straightforward such that the accuracy is deteriorating if a range of values is missing in the distribution. However, this was not discussed in existing DNN-based IO [16], [55]–[57] even though this affected the performances. Therefore, the data distribution is considered to eliminate the impact when splitting the data into training and validation ones in the future experiments. In this paper, the separation was performed according to the velocity norm, while orientation distribution was ignored. Other separation ways for six-DoF data would be investigated in future work. We empirically divided the KITTI dataset into training and validation datasets, respecting three principles: a similar distribution, a wide velocity norm representation in the training dataset and an approximate ratio of 70/30 percent, as illustrated in Fig. 3.

For the EuRoC dataset, the distribution of the velocity norm for training and validation is illustrated in Fig. 4. In this case, the data split for training and validation datasets follows other papers [56], [57].




(a) Track 1




(b) Track 2

**FIGURE 2.** Comparison between normal and missing-range learning with Track 1 and Track 2. Two approaches for the learning process were performed: normal learning and missing-range one. In both approaches, the training data were selected, as described in Table. 1.

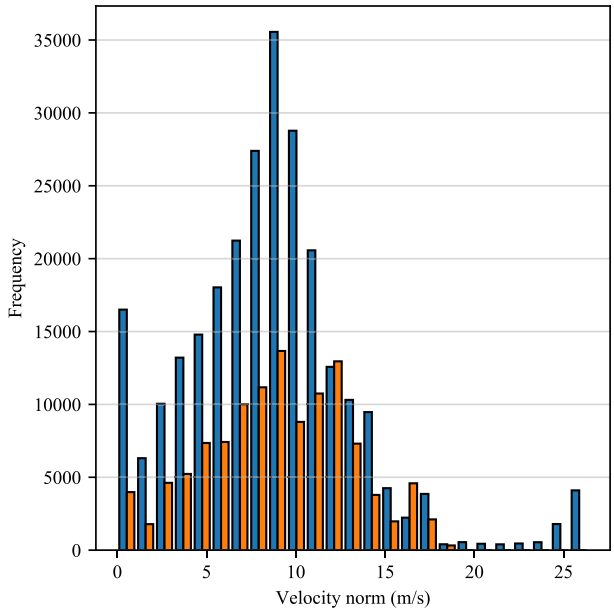



**FIGURE 3.** The distribution of the velocity norm in KITTI dataset. The histograms of the velocity norm in the training and validation dataset are illustrated in blue and orange, respectively.

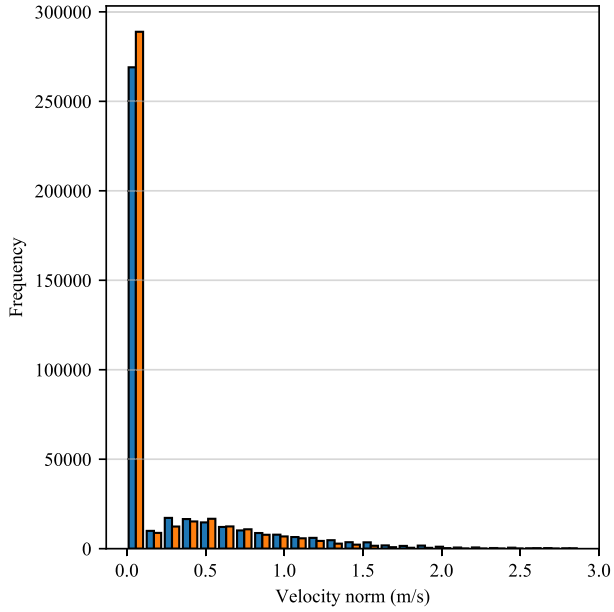



**FIGURE 4.** The distribution of the velocity norm in EuRoC dataset. The histograms of the velocity norm in the training and validation dataset are illustrated in blue and orange, respectively. The data split for training and validation follows other papers [56], [57].

## V. PRELIMINARY EVALUATION ON FOUR KINEMATIC APPROACHES

### A. CLASSIFICATION OF APPROACHES

As a second preliminary evaluation, the problem settings on the kinematic regression are discussed. Respecting the kinematic model in (3), orientation and velocity regression approaches can be classified into two categories. One is the relative kinematics represented by (3a) and (3b), and the other one is the absolute kinematics represented by (3c) and (3d). Depending on the output, the inputs fed into our neural network can be selected, as illustrated in Fig. 1 and Fig. 5. Conversely to absolute kinematics, the network for relative kinematics relies only on timestamp interval and angular velocity to predict orientation. The velocity branch of the relative kinematics uses the features of acceleration, previous quaternion, and timestamp interval, whereas the absolute one also required previous velocity and quaternion.

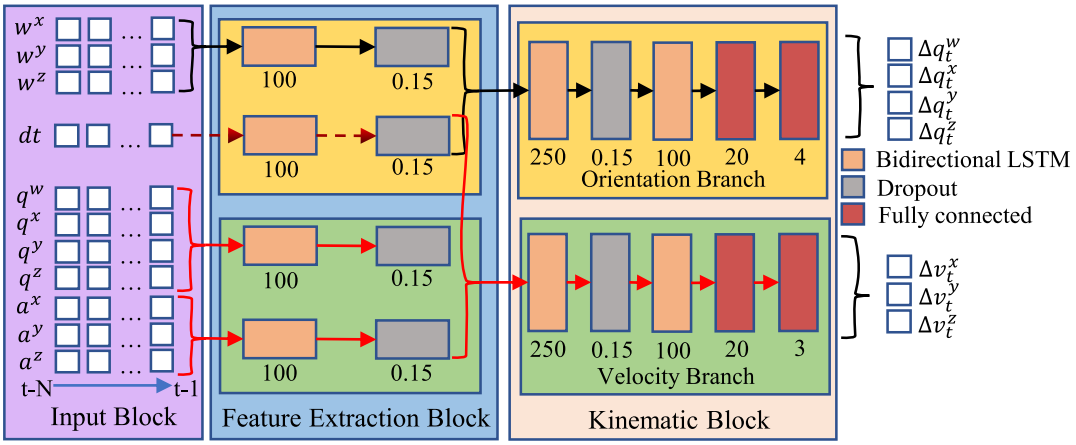



**FIGURE 5.** Network architecture for relative kinematic regression. This architecture outputs three-axis relative velocity $\Delta$v and four-dimensional quaternion $\Delta$q at time $t$. Each feature is separately extracted for each input in the Feature Extraction Block. Respecting the kinematic model in (3), features are concatenated and fed into two parallel networks: the orientation branch and the velocity branch. Features used for velocity regression are connected with red arrows, whereas features used for orientation regression are connected with black arrows. When the features are used for both regression problems, dark-red and dotted arrows are used. The number of units and dropout rate are shown below each layer.

**TABLE 5.** Classification of approaches on kinematic regressions. This table shows the different inputs fed into orientation and velocity branches of our neural network architecture according to the kinematic approaches: normal and constrained relative kinematics, and normal and over absolute kinematics.

| | | Relative kinematics | | Absolute kinematics | |
|---|---|---|---|---|---|
| | **Branch** | Normal (NRK) | Constrained (CRK) | Normal (NAK) | Over (OAK) |
| **Inputs** | Orientation | $\mathbf{w}_{t-1}^{IMU}/dt$ | $\mathbf{w}_{t-1}^{IMU}/dt/\mathbf{q}_{t-1}^{IMU}$ | $\mathbf{w}_{t-1}^{IMU}/dt/\mathbf{q}_{t-1}^{IMU}$ | $\mathbf{w}_{t-1}^{IMU}/dt$ |
| | Velocity | $\mathbf{a}_{t-1}^{IMU}/dt/\mathbf{q}_{t-1}^{IMU}$ | $\mathbf{a}_{t-1}^{IMU}/dt/\mathbf{q}_{t-1}^{IMU}/\mathbf{v}_{t-1}^{IMU}$ | $\mathbf{a}_{t-1}^{IMU}/dt/\mathbf{q}_{t-1}^{IMU}/\mathbf{v}_{t-1}^{IMU}$ | $\mathbf{a}_{t-1}^{IMU}/dt/\mathbf{q}_{t-1}^{IMU}$ |

Furthermore, the network can be designed in both well-conditioned and ill-conditioned manners. For instance, positional displacement is estimated from angular velocity and acceleration only whereas the previous velocity and orientation are not considered in [57]. This ill-conditioned case is referred to as over kinematics in this paper. On the other hand, additional data, which is not necessary for the mathematical kinematic model, can be used as constraints. This condition is referred to as constrained kinematics.

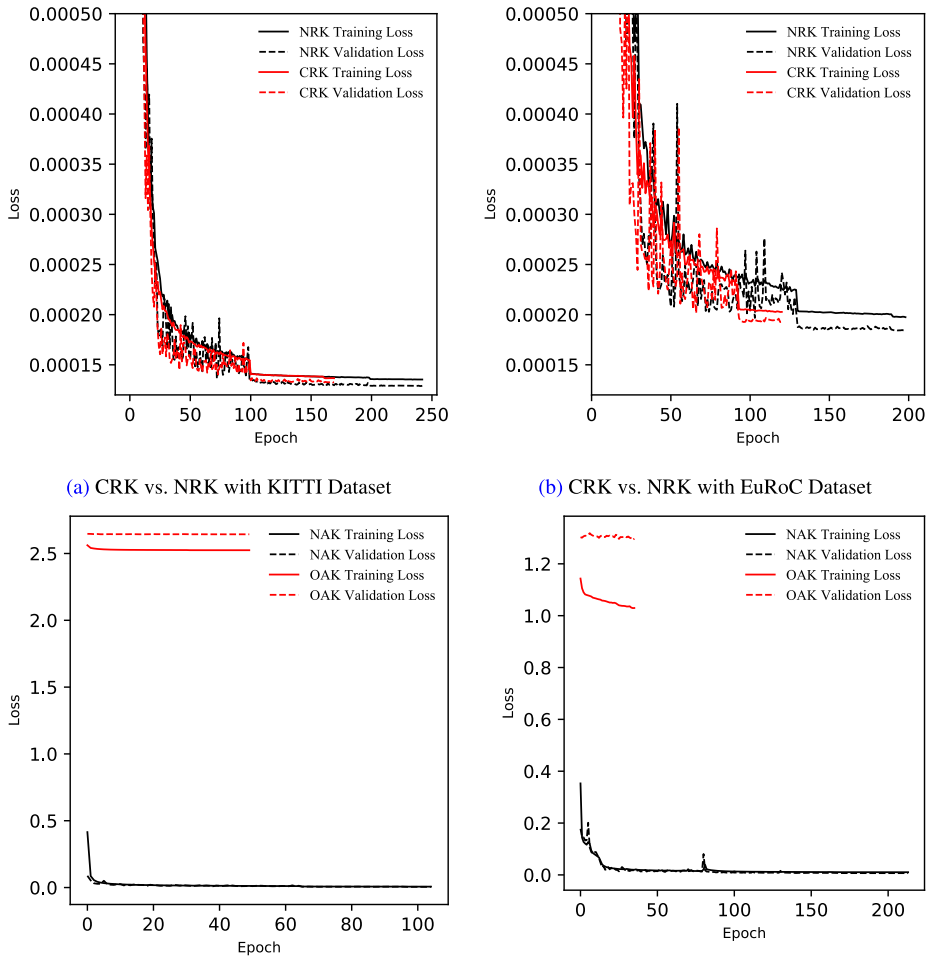
In the well-conditioned case, where the exact definition of the kinematic is considered, the regression of the absolute and relative kinematics are achieved as defined in Figs. 1, 5. We refer to these models as normal absolute kinematics (NAK) and normal relative kinematics (NRK).

In this paper, we define two other approaches: constrained relative kinematics (CRK) and over absolute kinematics (OAK). CRK predicts relative velocity and quaternion by extracting additional features coming from previous velocity and previous quaternion, respectively. Besides, OAK predicts absolute kinematics without using the previous state. In CRK, we investigate the influence of additional features over the regression accuracy, whereas we investigate the feasibility of the prediction where essential features are missing in OAK.

A summary of the kinematic approaches and their associated inputs are described in Table. 5. As a second preliminary evaluation, we investigate the difficulty of training each kinematic approach in this section. Generally, it is excepted that the difficulty of training OAK is higher than others because it is not theoretically feasible to predict absolute kinematic without previous state information. On the other hand, CRK should be the most accurate because additional inputs can provide useful features for the regression if the information is relevant. The purpose of this section is to consolidate these hypotheses and determine one kinematic approach for our future experiments.

### B. ANALYSIS SETTINGS

The details of the architecture for this analysis are as follows. The parameters in the architecture were similarly defined for all of the approaches to make the analysis fair. The LSTM weights were randomly initialized, respecting a normal distribution. A window size of $N = 11$ was fed into the neural network after every input was normalized between $[0, 1]$ except timestamp interval. For the output, only absolute velocity was normalized to ensure a balanced weight update process. During the learning process, we set the starting learning rate to 1e-3. Then, it was reduced by a factor of 0.1 every time a stationary state was faced for 25 sequential epochs in a row. The stationary state refers to the decline or the stagnation of the validation loss. After 35 epochs of the stationary state, the training was stopped. The dataset used for training and validation was the same for each




(a) CRK vs. NRK with KITTI Dataset

(b) CRK vs. NRK with EuRoC Dataset

(c) OAK vs. NAK with KITTI Dataset

(d) OAK vs. NAK with EuRoC Dataset

**FIGURE 6. Comparison of four kinematic approaches. For both KITTI and EuRoC datasets, the losses for training and validation are respectively illustrated with a full line and a dotted line. NRK and NAK are represented in black whereas red curves are used for CRK and OAK.**

approach, such that it was similarly shuffled into a batch size of 256.

The loss function used for back-propagation was similar to the one used in [57]. For the absolute and relative velocity and quaternion, the loss is expressed as follows.

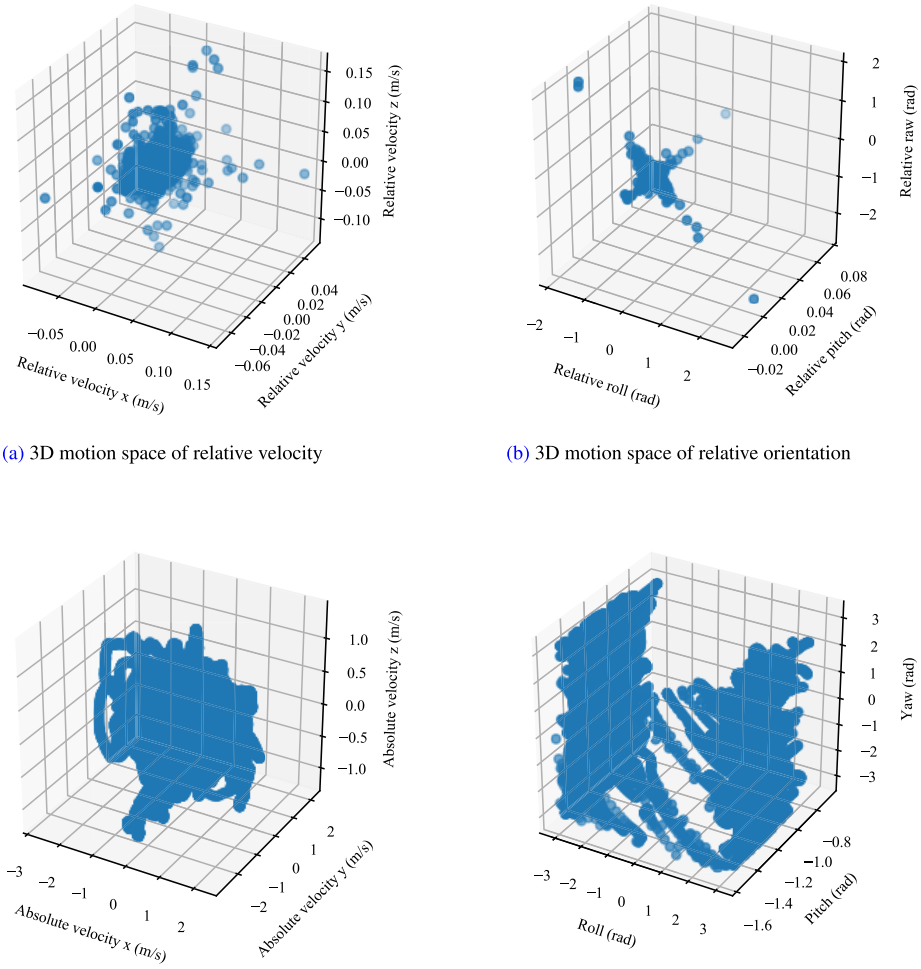
$$\begin{cases} \mathcal{L}_1 = \|\hat{\mathrm{v}} - \mathrm{v}\|_2^2 & \text{(5a)} \\ \mathcal{L}_2 = 2 \cdot \|imag(\hat{\mathrm{q}} \otimes \mathrm{q}^*) - \log(\|\hat{\mathrm{q}}\|)\|_1 & \text{(5b)} \\ \mathcal{L}_{Final} = \sum_{i=1}^{2} \exp(-\log(\sigma_i^2))\mathcal{L}_i + \log(\sigma_i^2) & \text{(5c)} \end{cases}$$

The loss function for velocity was defined in (5a) as the square of the L2 norm between the ground truth v and the predicted value $\hat{\mathrm{v}}$. The orientation error metric in (5b) was based on the Hamilton product between the predicted quaternion $\hat{\mathrm{q}}$ and the ground truth conjugate quaternion $\mathrm{q}^*$. It is worth noting that both quaternions were normalized. An additional logarithm term was added to the orientation metric to prevent the neural network from outputting zero values. Both loss functions shared a different scale and different nature that could harm the learning process. Therefore, a multi-loss layer was defined to ensure a proper balance between orientation and velocity metrics. For each metric, the multi-loss function introduced a trainable parameter $\log(\sigma_i^2)$, where $\sigma_1^2$ and $\sigma_2^2$ are the variance of the metric for velocity and orientation regressions, respectively. Each parameter was initially set to zero and then was automatically adjusted during the learning process. The final loss was defined as (5c).

### C. RESULT

For each kinematic approach, one example of the learning curve for both training and validation was analyzed in Fig. 6. The learning process was performed by using the data distribution in Table. 1 and Table. 2.

In Fig. 6a and Fig. 6b, we compared NRK with CRK for both KITTI and EuRoC datasets. In both cases, the neural network learned a model to output relative orientation and




(a) 3D motion space of relative velocity

(b) 3D motion space of relative orientation

(c) 3D motion space of absolute velocity

(d) 3D motion space of absolute orientation

**FIGURE 7. The distribution of both orientation and velocity in the EuRoC dataset.**

relative velocity. The training and validation loss converge towards a similar accuracy for both kinematic approaches. Contrary to our expectation, the use of additional features did not clearly end up with greater performances. In this evaluation, only one network architecture was analyzed even though a multitude of CRK representations may exist. From this assessment, further analysis of CRK was performed in the future experiments.

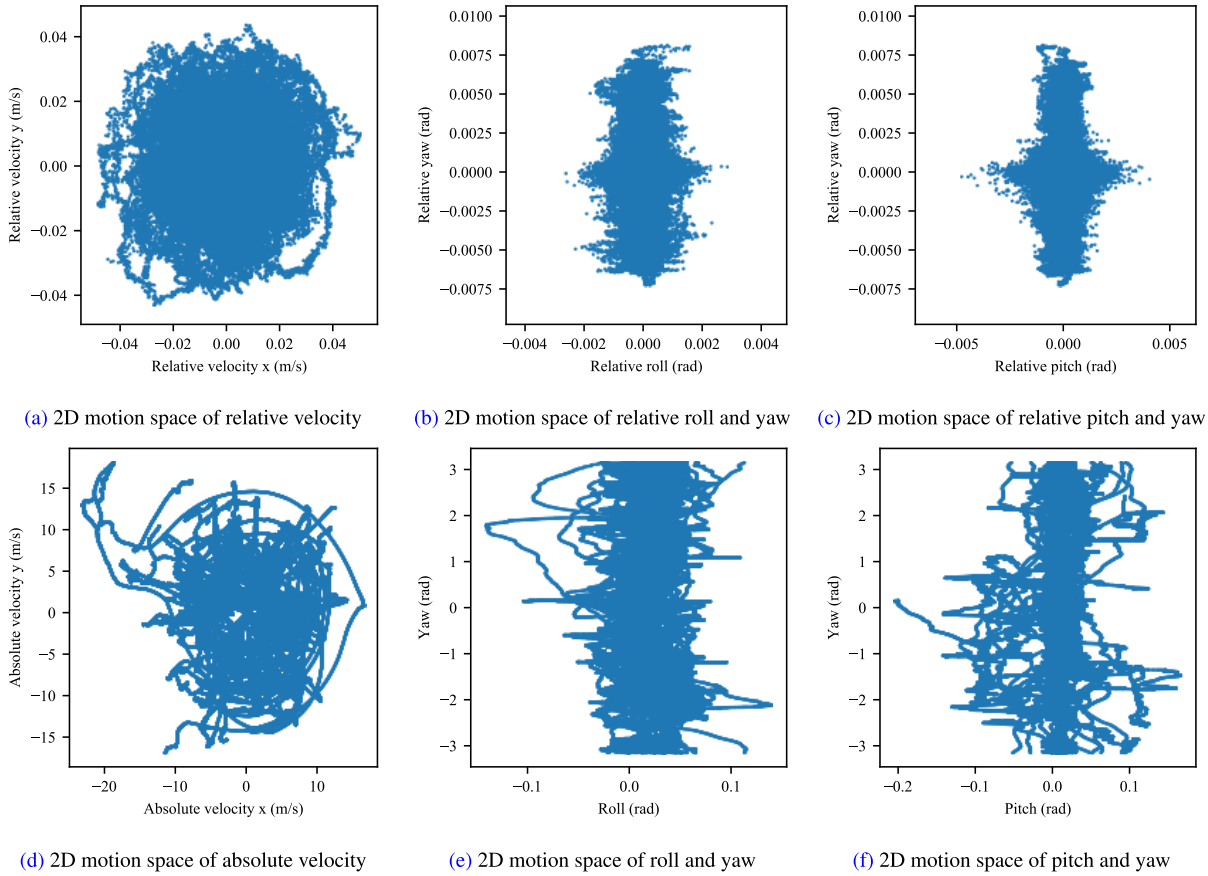
In Fig. 6c and Fig. 6d, NAK was compared with OAK for both KITTI and EuRoC datasets. As expected, the neural network could not predict absolute orientation and absolute velocity using OAK due to the lack of information. The losses for the training and validation were higher than those of NAK and converged only after a few epochs. Nevertheless, the absolute kinematic was predictable when the definition of the kinematic model was followed in the design of our architecture.

### D. DISCUSSION ON MOTION SPACE

The purpose of this analysis was to compare the performances of each approach in Table. 5 and establish if our kinematics-mimicking DNN was accurately trainable or not. Both absolute and relative kinematics were predictable if sufficient information was provided. In the future sections of this paper, we will focus our experiments either on relative or absolute kinematics to investigate one of them deeply. In this subsection, the motion space of both representations was analyzed to make our decision.

In Fig. 7, the velocity and orientation in the EuRoC dataset are represented in 3D, whereas the motion space of the KITTI dataset is analyzed from a 2D point of view in Fig. 8. In our analysis, the z-axis velocity of a car is not considered due to its constancy. Euler angles were preferred to quaternion due to better visualization of the motion space. For both datasets, the density of the relative velocity and the relative orientation was higher than that of absolute ones.

When driving a car, users usually keep a constant speed and hold a similar orientation in many situations. At some points, the car accelerates or decelerates, turns left or right, making the motion space bigger. Based on the absolute pose of the car, each velocity and each orientation pose represent a specific case for IO. Furthermore, a neural network may have more




(a) 2D motion space of relative velocity (b) 2D motion space of relative roll and yaw (c) 2D motion space of relative pitch and yaw

(d) 2D motion space of absolute velocity (e) 2D motion space of roll and yaw (f) 2D motion space of pitch and yaw

**FIGURE 8.** The distribution of both orientation and velocity in KITTI dataset.

difficulties to predict a specific velocity if the neural network was not trained with the value beforehand, as proved in Fig. 2 and Table. 3. The same assessment can be made for the relative kinematics. Nevertheless, the relative orientation and the relative velocity are defined between two specific frames, making the broad-spectrum easier to represent during data acquisition. Thus, the motion space for relative kinematics is denser than the absolute one and is more easily trainable by DNN with the given datasets. From this observation, our future experiments will focus on relative kinematic regression such as NRK and CRK.

## VI. DEEP ANALYSIS ON KINEMATICS-MIMICKING DNN-BASED IO

In the previous sections, we investigated both the data distribution for training and validation and four possible architectures of kinematics as preliminary evaluations. These were necessary to determine the choices to be investigated in the analysis of DNN-based IO. From this section, we review significant points associated with DNN we consider essential when dealing with IO.

Hyper-parameters are considered matters to examine in DNN. Deep learning is a wide area in which a great deal of adjustments is needed to boost the prediction accuracy. However, only a few of them are directly associated with IO. Previous works on IO used DNN in many different ways, as described in Table. 6. In those papers, some hyper-parameters such as window size were established as the most optimal one heuristically. Nonetheless, recent studies were capable of achieving great performances with different criteria. In that way, the purpose of this paper is to openly discuss which hyper-parameters are widely correlated to the performances and suggest trails to be considered when designing DNN. Through our analysis, we deal with possible inputs related to INS that characterize the performance. Next, we discuss the window size of the input and its frequency representation. Then, we appointed a section to discuss the loss function and the output representation we used. Furthermore, we come up to discuss the usage of future data and data pre-processing. To the best of our knowledge, such an investigation was not performed in the literature.

In the list below, we summarized hyper-parameters to be investigated while designing DNN. The sections where they are discussed in this paper are also described.

- Hyper-parameters related to IO
  - Inputs for DNN-based IO (Section VII-A)
  - Window size of input (Section VII-B)
  - Data frequency (Section VII-C)

- ☐ Loss function (Section VII-D)
- ☐ Relative quaternion representation (Section VII-E)
- ☐ Use of future data (Section VII-F)
- ☐ Data pre-processing
  - ∗ Data normalization (Section VII-G)
  - ∗ Data balance (Section VII-H)

■ Hyper-parameters related to DNN (Section VIII-A)

- ☐ Learning rate
- ☐ Activation function
- ☐ Optimizer
- ☐ Adaptive learning rate
- ☐ Batch size
- ☐ Number of layers and their hidden units
- ☐ Weight initialization
- ☐ Regularization
- ☐ Batch normalization
- ☐ Early stopping
- ☐ Data filtering

The motivation of these selected hyper-parameters is explained in each paragraph as follows.

First, the choice of inputs for DNN-based IO is discussed. Based on the kinematic definition in (3), different variables intervene in the double integration process, such as IMU measurements, latest quaternion, previous velocity, and timestamp interval. However, most of the existing DNN-based IO only used angular velocity and acceleration as input, whatever the desired output was, and also ignored the impact of previous knowledge on the prediction, except for [56]. When the absolute movement in the world frame is estimated, the preceding state is a crucial source of information. In (3b), previous quaternion is directly used to predict relative velocity. Besides, orientation may be a factor responsible for noise in measurements, whereas the timestamp interval is a piece of valuable information for IO. Since the latter has a critical role in estimating sensor movements, a simple variation in the sampling delay could be relevant for the noise analysis. Hence, it is hardly imaginable to pass by such useful features, merely using IMU data. Furthermore, CRK was defined in Section V-A to outperform NRK using additional information as input. Nevertheless, CRK achieved similar accuracy to NRK. From this observation, we decided to design experiments on additional inputs related to IO in Section VII-A.

Next, the window size of the input is approached as a hyper-parameter to reaffirm whether this hyper-parameter can be established heuristically from a work to another or not. This is because the choice of window size is not well-investigated. This discussion is handled in Section VII-B.

Then, the correlation between the frequency and the performances of our neural networks is investigated. Typically, we use the frequency of IMU data given in the dataset without modifying it. However, it is not clear that the frequency is sufficient or insufficient for modeling IO problems. The required frequency may depend on the complexity of the motion. If the data at low frequency provides enough accuracy, a reduction of processing can be achieved. This is discussed in Section VII-C.

In IO and more general problems in deep learning, the most standard loss function is the Mean Square Error (MSE), as shown in Table. 6. Nevertheless, a multitude of variation exists to compute the error between the predicted output and its ground truth. In Section VII-D, we investigated other possibilities to represent the loss function introduced in (5). This analysis is then extended to the orientation loss function, where we investigated two representations of the relative quaternion in Section VII-E.

The use of future data with DNN-based IO is recurrent in other works, as described in Table. 6. From the inputs and outputs choices, the use of convolution or bidirectional LSTM, or else from the computation methodology, future features can be extracted for the regression problem. In other words, a delay prediction is generated that leads to the improvement of the performances. Nevertheless, the distribution between past data and future data was not introduced in previous works and could be an important hyper-parameter to be investigated. This is discussed in Section VII-F.

Another important notion associated with deep learning is data pre-processing. It regroups several key aspects such as data normalization, outlier detection and removal, data smoothing and data balancing. They are mainly responsible for accuracy improvement. Even though this subject has been studied in other fields, it is essential to keep it in mind while dealing with IO. Therefore, we first discussed different techniques to normalize inputs correlated to the IO problem in Section VII-G. Then, the notion of data balancing is discussed in Section VII-H.

Other hyper-parameters are not investigated in this paper because we empirically selected highly-correlated ones to the IO problems only. However, their role is crucial for the performances. For this reason, a discussion was devoted to the general hyper-parameters of DNN in Section VIII-A.

## VII. HYPER-PARAMETERS RELATED TO IO

### A. INPUTS FOR DNN-BASED IO

#### 1) ANALYSIS SETTINGS

Even though we follow the kinematic model defined in (3), we found that we could achieve similar or better accuracy when using additional inputs, as described in Section V-A. Thus, the purpose of this section is to analyze the performances of our neural network with different inputs related to IO. From such an analysis, we tried to identify the variable that could improve the prediction accuracy of the relative velocity and orientation.

In this analysis, we extended the architecture defined in Fig. 5 with many-to-many LSTM to output $N$ sets of relative quaternions and relative velocities covering the range $[t - N + 1, t]$ instead of outputting one pair of the result at time $t$, as similar to [56]. The new architecture is illustrated in Fig. 9. Among these $N$ predictions, we investigated the output at time $t$ where only past features were used for the

**TABLE 6. Summary of the parameters used in related work. Important parameters related to IO such as DNN layers, inputs and their associated window size are referred. The loss functions used for the backpropagation process and the use of future features are also indicated.**

| Parameter / Paper | Layer | Inputs | Window Size | Output | Loss | Future features |
|---|---|---|---|---|---|---|
| RIDI [47] | SVM/SVR | IMU | 200 | $v \in \mathbb{R}^3$ | $\epsilon$-insensitive | No |
| IONET [52] | bi-LSTM | IMU | 200 (50,100,400) | $\Delta L$ / $\Delta\theta$ | Weighted($\|\Delta L\|_2^2$, $\|\Delta\theta\|_2^2$) | Yes |
| IONET2 [53] | bi-LSTM | IMU | 200 (50,100,400) | $\Delta L$ / $\Delta\theta$ | Weighted($\|\Delta L\|_2^2$, $\|\Delta\theta\|_2^2$) | Yes |
| Ronin [55] | uni-LSTM | IMU | 400 | $\Delta\mathbf{v} \in \mathbb{R}^2$ | $\|\Delta v_{[1,400]}\|_2$ | No |
| Ronin [55] | ResNet | IMU | 200 | $\Delta\mathbf{p} \in \mathbb{R}^2$ | MSE($\Delta\mathbf{p}_{[1,200]}$) | No |
| Ronin [55] | TCN | IMU | 253 | $\Delta\mathbf{v} \in \mathbb{R}^2$ | $\|\Delta\mathbf{v}_{[1,253]}\|_2$ | No |
| Ronin [55] | uni-LSTM | IMU | - | $\sin\theta$/ $\cos\theta$ | MSE($\sin\theta$,$\cos\theta$) + Normalization | No |
| Lima et al. [57] | Convolution/ bi-LSTM | IMU | 200 | $\Delta p \in \mathbb{R}^3$ / $\Delta q$ | Weighted ($\|\Delta p_{[95,105]}\|_1$,$d(\Delta q_{[95,105]})$) | Yes |
| AbolDeepio [25] | bi-LSTM | IMU / dt | 20 (10) | $\Delta p$ / $\Delta q$ | Weighted RMSE($\|\Delta p\|_2$, $\|\Delta q\|_2$) | both |
| OriNet [56] | bi-LSTM | IMU / dt / $\mathbf{q}_{t-1}$ | 11 | $\Delta q$ | MSE($\Delta q$) | Yes |
| AI-IMU [16] | Convolution | IMU | 200 | $[n^{lat}, n^{lat}] \in \mathbb{R}^2$ | Relative translation error $\mathbf{t}_{rel}$ | Yes |
| Cortés et al. [50] | Convolution | IMU | 200 | $\|v\| \in \mathbb{R}$ | MSE($\|v_{[1,200]}\|$) | No |

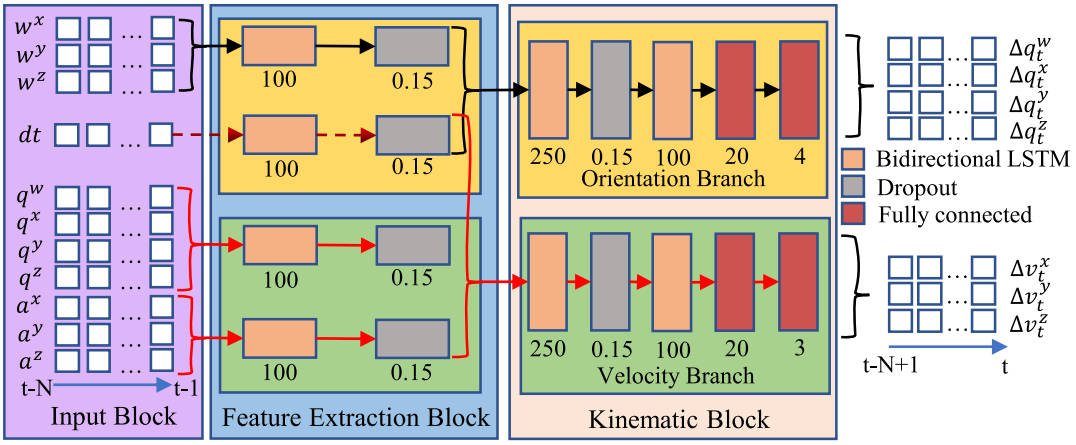



**FIGURE 9. Network architecture for *N* predictions of relative orientation and relative velocity. Compare to Fig. 5, this architecture outputs *N* predictions of three-axis relative velocity $\Delta\mathbf{v}$ and unit quaternion $\Delta\mathbf{q}$ from time $t - N + 1$ to time $t$.**

regression. The output at time $t - (N - 1)/2$, where both past and future features were extracted from bidirectional LSTM, was also analyzed. It is worth noting that different inputs may require a new configuration of hyper-parameters. However, they were fixed from an evaluation to another for the sake of the analysis. For this analysis, $N$ was set to 11.

We designed 18 experiments from E-0 to E-17 by changing the combination of inputs for CRK, as described in Table. 7. E-0 is considered as the initial experiment that represents NRK in Table. 5. The purpose of each experiment is given as follows.

- **E-1, E-2, E-3**
  In NRK, the features of angular velocity were not used to predict the relative velocity, and those of acceleration were not used to regress relative orientation. In E-1, E-2, and E-3, we changed the combination of the features of acceleration and angular velocity.
- **E-4, E-5, E-6**
  In these experiments, the combination of the inputs converges step by step towards the CRK defined in Table. 5. First, the features of the previous velocity were added to the velocity branch. Then, the previous quaternion was also used for orientation prediction. Finally, one more step was defined to the initial definition of CRK where the previous velocity was fed into both orientation and velocity branches.
- **E-7**
  *dt* was removed to observe its influence.
- **E-8**
  Quaternion representation in E-4 was replaced with a nine-dimensional vector that represents a $3 \times 3$ rotation matrix.
- **E-9**
  The multiplication of two variables is introduced as a new input in this experiment. In (3), acceleration and

angular velocity are multiplied with timestamp interval. DNN can learn such correlation. Nevertheless, the approximation of the model can break down if the training data does not represent the entirety of possibilities, as we observed in Section IV. Furthermore, it is hard to determine whether DNN highlights the correlation between features or not. From these observations, we supported the feature extraction by explicitly expressing the multiplication between variables. One way is to achieve a multiplication between extracted features inside the neural network to represent the same kinematic behavior implicitly. Nevertheless, the resulting product may have a wide variance and bring about significant information loss. Therefore, we investigated the multiplication between IMU data and timestamp interval as input for our architecture. In this experiment, acceleration, angular velocity and $dt$ of E-5 were replaced with $\mathrm{a}_{t-1}^{IMU}\,dt$ and $\mathrm{w}_{t-1}^{IMU}\,dt$.

- **E-10**
  From the same premise established for E-9, acceleration was multiplied to the rotation matrix as described in (3b). Then, the resulting multiplication was fed into the network to substitute the acceleration and the rotation matrix of E-4.
- **E-11, E-12, E-13**
  In the previous experiments, the inputs were carefully selected and concatenated respecting (3). Next, we analyzed the behavior of our neural network when every input was fed into both branches. In other words, we extended E-4, E-8 and E-9 into E-11, E-12, E-13.
- **E-14**
  In E-11 and E-13, the original and the multiplied inputs were tested separately. As an extension of these two experiments, both input representations were used to predict relative quaternion and relative velocity in E-14.
- **E-15, E-16**
  Based on E-11, E-15 and E-16 were experiments designed to investigate respectively the influence of the rotation matrix and the resulting multiplication between the acceleration and the rotation matrix.
- **E-17**
  As a final experiment, we used every possible input introduced in previous experiments to regress relative kinematics. The inputs were fed into both orientation and velocity branches.

### 2) EVALUATION

To compare experiments, two criteria were computed at time $t$ and time $t-(N-1)/2$. One is for the velocity prediction, and the other one is for the orientation prediction, as described in Table. 8. The metric of the relative velocity error is represented with the RMSE. For the metric of the orientation error, we used (5b) without the logarithm term. Through the testing process, the errors were computed from the 72 sequences in Table. 1 and the average was considered.

Each learning process is generally subjected to randomness effect from dropout, weight initialization, optimizer, and so on. Moreover, the errors between the 18 experiments slightly differ, making the comparison difficult to achieve from only one learning process. Besides, the behavior observed for a specific criterion such as the velocity error at time $t-(N-1)/2$ may be different for another criterion such as the velocity error at time $t$. Nevertheless, the purpose of this section is to clarify important aspects rather than to state the best combination of inputs. Therefore, a global analysis is performed in this section. The main points of this analysis are discussed as follows.

- **Does CRK outperform NRK?**
  First, adding the features of angular velocity to the velocity branch deteriorated relative velocity prediction and barely improved the orientation prediction in E-1. On the other hand, adding the features of acceleration to the orientation branch in E-2 slightly improved the velocity at time $t-(N-1)/2$ and the orientation prediction at time $t$, but slightly deteriorated orientation at time $t-(N-1)/2$. In E-3, both features were used for each regression task. A minor improvement for the orientation prediction was achieved contrary to velocity prediction, where the gap between training and validation error was higher.

  Compared to NRK, CRKs defined in E-4, E-5, E-6 obtained better performances for the relative velocity prediction, contrary to the orientation, which got worse. Furthermore, adding the previous quaternion to the orientation branch in E-5 slightly improved orientation prediction at time $t$ compared to E-4. However, a significant improvement was expected from using the previous quaternion as the input of the orientation branch, as discussed in [56].

  Similar to the result observed in Section V-C, the designed CRKs from E-1 to E-6 did not outperform NRK as we would have expected. Therefore, using additional information does not necessarily end up with better performances. Furthermore, observations established from a specific network may not be applicable to another architecture. Nonetheless, additional inputs were used from E-7 to E-17, and some of them were able to excel in both regressions of the relative kinematics, reaffirming the idea of using additional information for IO problems.
- **Is it relevant to use multiplied inputs?**
  In E-9, E-10, E-13, acceleration, angular velocity, timestamp interval, and previous quaternion were replaced by an alternative input representing the multiplication between two variables that intervenes during the double integration process. In comparison with E-5, E-4, and E-11 where the original representation of the input was used, better performances were observed for E-10, contrary to E-9 and E-13 where performances are mixed.
  In (1), the noise and bias generated during the data acquisition are associated with the acceleration and the angular velocity only, not their multiplicative representation. However, replacing the original inputs with our

**TABLE 7.** Inputs for DNN-based IO. In each experiment, different inputs were used to output relative orientation and relative velocity respecting the network in Fig. 9. The details of the experiments are indicated with colored and labelled cells. Blue cells annotated with 'v' indicate that the input was used for the velocity branch. Red cells annotated with 'o' indicate that the input was used for the orientation branch. Finally, the input was used for both branches when the cell is green with 'v/o'. Each input data is correlated with the kinematic model in (3).

| Input / No. | $\mathbf{a}_{t-1}^{IMU}$ | $\mathbf{w}_{t-1}^{IMU}$ | $dt$ | $\mathbf{q}_{t-1}^{IMU}$ | $\mathbf{v}_{t-1}^{IMU}$ | $\mathbf{a}_{t-1}^{IMU}dt$ | $\mathbf{w}_{t-1}^{IMU}dt$ | $Rot(\mathbf{q}_{t-1}^{IMU})$ | $Rot(\mathbf{q}_{t-1}^{IMU})\mathbf{a}_{t-1}^{IMU}$ |
|---|---|---|---|---|---|---|---|---|---|
| E-0 | v | o | v/o | v | | | | | |
| E-1 | v | v/o | v/o | v | | | | | |
| E-2 | v/o | o | v/o | v | | | | | |
| E-3 | v/o | v/o | v/o | v | | | | | |
| E-4 | v | o | v/o | v | v | | | | |
| E-5 | v | o | v/o | v/o | v | | | | |
| E-6 | v | o | v/o | v/o | v/o | | | | |
| E-7 | v | o | | v/o | v | | | | |
| E-8 | v | o | v/o | | v | | | v | |
| E-9 | | | | v/o | v | v | o | | |
| E-10 | | o | v/o | | v | | | | v |
| E-11 | v/o | v/o | v/o | v/o | v/o | | | | |
| E-12 | v/o | v/o | v/o | | v/o | | | v/o | |
| E-13 | | | | v/o | v/o | v/o | v/o | | |
| E-14 | v/o | v/o | v/o | v/o | v/o | v/o | v/o | | |
| E-15 | v/o | v/o | v/o | v/o | v/o | | | v/o | |
| E-16 | v/o | v/o | v/o | v/o | v/o | | | | v/o |
| E-17 | v/o | v/o | v/o | v/o | v/o | v/o | v/o | v/o | v/o |

new input representation may prevent the neural network to estimate the noise and the bias of each IMU sensor. Therefore, both input representations were fed into the network in E-14, E-16, E-17. Each experiment outperformed the case where only one input representation was used for the regression of the velocity. On the other hand, the performances of the orientation prediction were different. Compared to E-11, E-14 achieved a better orientation prediction at time $t-(N-1)/2$, conversely to the accuracy of the prediction at time $t$ that is a bit lower. Besides, the orientation prediction of E-16 was worse than E-11, whereas E-17 outperformed it. Both regression tasks outperformed the cases where only one input representation was used. From this observation, we reaffirmed the relevance of using multiplied inputs as additional information for IO problems with our architecture.

- **Is it relevant to use timestamp interval?**
  From E-5 and E-7, the removal of $dt$ from the inputs deteriorated the accuracy for both relative quaternion and velocity. However, this observation might not be enough to conclude about the relevance of timestamp interval for IO problems. Apart from the randomness effect of the learning process, our architecture may not have been able to extract sufficient features in E-7. Therefore, the observed result may differ according to the architecture.
- **Is it relevant to represent the orientation with a rotation matrix ?**
  E-8, E-12 used rotation matrix to represent the previous orientation as input. Both experiments achieved greater performances than E-4, E-11 for both relative kinematics. Furthermore, the gap between training loss and validation was less important, suggesting the rotation matrix offered a better weight regularization than quaternion inputs for our architecture. Such an explanation could justify the fact that orientation regression was better for E-8 whereas the rotation matrix was not directly influencing the orientation branch.
- **Is it relevant to use all inputs?**
  From E-11, E-12, we realized using the extracted features for both branches resulted in the accuracy improvement. Additional inputs were used in E-14, E-15, and E-16, leading to better accuracy and stability of the learning process. E-17 concluded our analysis using every possible input for both branches and outperformed every other experiment for the prediction of relative velocity. On the other hand, the prediction of the relative quaternion was not the most optimal one we could achieve but its performances were praiseworthy. The higher the number of inputs was, the higher the number of parameters inside the DNN was. However, it does not necessarily result in better prediction. On the contrary, DNN can be more subjected to overfitting. Nonetheless, from all these inputs, our architecture managed to extract key features to output relative quaternion and relative velocity.

From this analysis, one overall conclusion can be drawn. We may need to conclude that there is no optimal answer to the question of which input suits the best for IO problems. According to the neural network architecture and the combination of input data, the optimal accuracy or the worst one can be obtained. As aforementioned, the purpose of this section is to offer an overview of inputs and combination possibilities with quantitative analysis. Even though everything has not been tested, several trails such as input multiplication and orientation representation may offer new perspectives for future work.

**TABLE 8.** Performances of the experiments in Table. 7. The prediction errors of the relative kinematics at time $t-(N-1)/2$ and time $t$ over 72 sequences from KITTI dataset are described in this table. The data distribution for the learning process is referred in Table. 1. Among the 18 experiments achieved, the worse predictions are highlighted with a red cell whereas the best ones are shown in green. The second best predictions are written in bold type.

(a) Average RMSE of the relative velocity

| Loss / No. | At time $t-(N-1)/2$ | | At time $t$ | |
|---|---|---|---|---|
| | Training | Validation | Training | Validation |
| E-0 | 3.486e-4 | 1.411e-3 | 8.414e-4 | 1.676e-3 |
| E-1 | 1.746e-4 | 2.504e-3 | 5.284e-4 | 2.824e-3 |
| E-2 | 2.051e-4 | 1.370e-3 | 6.439e-4 | 1.679e-3 |
| E-3 | 2.052e-4 | 2.167e-3 | 5.801e-4 | 2.576e-3 |
| E-4 | 1.637e-4 | 8.918e-4 | 5.464e-4 | 1.106e-3 |
| E-5 | 1.896e-4 | 7.438e-4 | 5.656e-4 | 9.513e-4 |
| E-6 | 1.689e-4 | 9.181e-4 | 5.659e-4 | 1.092e-3 |
| E-7 | 2.316e-4 | 8.914e-4 | 6.171e-4 | 1.073e-3 |
| E-8 | 1.634e-4 | 3.685e-4 | 5.549e-4 | 6.782e-4 |
| E-9 | 2.534e-4 | 9.801e-4 | 7.163e-4 | 1.240e-3 |
| E-10 | 2.351e-4 | 3.430e-4 | 6.423e-4 | 7.018e-4 |
| E-11 | 1.746e-4 | 8.803e-4 | 5.201e-4 | 1.072e-3 |
| E-12 | 1.681e-4 | 3.905e-4 | 5.103e-4 | 6.522e-4 |
| E-13 | 2.235e-4 | 9.618e-4 | 6.120e-4 | 1.173e-3 |
| E-14 | **1.520e-4** | 7.930e-4 | 4.757e-4 | 9.696e-4 |
| E-15 | 1.785e-4 | 4.025e-4 | 5.286e-4 | 6.747e-4 |
| E-16 | 1.627e-4 | **2.736e-4** | 5.320e-4 | **5.938e-4** |
| E-17 | 1.394e-4 | 2.625e-4 | **4.816e-4** | 5.509-e4 |

(b) Average error of the relative quaternion

| Loss / No. | At time $t-(N-1)/2$ | | At time $t$ | |
|---|---|---|---|---|
| | Training | Validation | Training | Validation |
| E-0 | 1.788e-5 | 1.937e-5 | 1.270e-4 | 1.276-4 |
| E-1 | 1.671e-5 | 1.957e-5 | 1.238e-4 | 1.269e-4 |
| E-2 | 1.854e-5 | 2.033e-5 | 8.864e-5 | **9.010e-5** |
| E-3 | 1.759e-5 | 1.934e-5 | 8.650e-5 | 8.823e-5 |
| E-4 | 2.129e-5 | 2.367e-5 | 1.342e-4 | 1.372e-4 |
| E-5 | 2.006e-5 | 2.376e-5 | 1.268e-4 | 1.339e-4 |
| E-6 | 1.705e-5 | 2.573e-5 | 1.291e-4 | 1.425e-4 |
| E-7 | 2.431e-5 | 3.154e-5 | 1.323e-4 | 1.376e-4 |
| E-8 | 1.613e-5 | **1.861e-5** | 1.337e-4 | 1.353e-4 |
| E-9 | 1.860e-5 | 2.510e-5 | 1.335e-4 | 1.398e-4 |
| E-10 | 1.580e-5 | 1.764e-5 | 1.260e-4 | 1.263e-4 |
| E-11 | 1.805e-5 | 2.550e-5 | 8.306e-5 | 9.289e-5 |
| E-12 | 1.710e-5 | 2.330e-5 | 8.031e-5 | 9.039e-5 |
| E-13 | 1.721e-5 | 3.284e-5 | 9.358e-5 | 1.196e-4 |
| E-14 | **1.423e-5** | 2.374e-5 | 7.620e-5 | 9.420e-5 |
| E-15 | 2.184e-5 | 2.717e-5 | 8.551e-5 | 9.526e-5 |
| E-16 | 1.719e-5 | 2.749e-5 | 8.732e-5 | 1.033e-4 |
| E-17 | 1.380e-5 | 2.191e-5 | **7.720e-5** | 9.141e-5 |

Note that additional data from air pressure and temperature sensors were not considered in this work. They may be responsible for causing noise in acceleration and angular velocity and may contain useful features to be considered while dealing with IO.

### B. WINDOW SIZE OF INPUT

#### 1) ANALYSIS SETTINGS

In the previous works, the window size of input tends to be equal to 200, as described in Table. 6. Even though this hyper-parameter might achieve good performances, the purpose of this section is to show that such a hyper-parameter needs to be investigated whenever building up DNN.

It is easy to think that the feature extraction will be more relevant when the dimension of the input data is higher. However, the curse of dimensionality phenomenon is a significant obstacle, causing poor performances in general regression problems. The computation time proportionally increases as the window size gets larger and does not always result in better prediction.

In this experiment, we analyzed the performances of our neural network defined in Fig. 9 by using a window of $N$ inputs to output $N$ sets of relative quaternions and velocities between $[t-N+1, t]$. From this architecture choice, the window size hyper-parameter can be investigated for two different distributions of features, as mentioned in Section VII-A. Six different window sizes were considered, such as 11, 25, 50, 75, 100, 200. The learning process was performed with KITTI dataset following the data distribution, as described in Table. 1. Through the testing process, five tracks from the KITTI dataset were selected to compute the RMSE of the relative velocity and the same orientation metric defined in Section VII-A. The error metrics are shown in Table. 9. The tracks used during the testing process of this experiment are referred in Table. 4.

#### 2) EVALUATION

From the five tracks tested, the best result was obtained three times for the relative velocity at time $t-(N-1)/2$ for the window size of 200. Most of the time, our network showed its best accuracy when a middle-size window of inputs was used. Among the six possibilities of the window size, the size of 50, 75, 100 showed better performances for both relative kinematics predictions. Nonetheless, the window size of 11 showed interesting results for orientation prediction, as reported in [56]. However, this result was only observable at time $t$, and the accuracy for the velocity prediction was the worst among all the cases tested.

Through this experiment, the difficulties in generalizing the hyper-parameter for the window size was clarified. The difference in accuracy observed between $t-(N-1)/2$ and $t$ suggests that the input dimension may be related to the distribution of future and past data. Such correlation is investigated in Section VII-F. As a new perspective for future work, another trail can be considered to make this section complete. Some motions such as turning right, left, or going straight forward should require more or less information to be identifiable. Therefore, having an adaptive window size can be worth considering as a new perspective for future work to improve the performances.

### C. DATA FREQUENCY

#### 1) ANALYSIS SETTINGS

The measurement frequency of IMU varies according to devices. Generally, the given frequency is used for processing. However, a higher frequency signal may not be required

**TABLE 9.** Performances of NRK for different window sizes. The prediction errors at time $t-(N-1)/2$ and time $t$ for different window sizes are described for the testing tracks, as introduced in Table. 4. Best predictions are indicated with green cells, whereas the worse accuracies are indicated with a red cell. The second best predictions are written in bold type.

(a) RMSE of the relative velocity

| Track \ Size | At time $t-(N-1)/2$ | | | | | | At time $t$ | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 11 | 25 | 50 | 75 | 100 | 200 | 11 | 25 | 50 | 75 | 100 | 200 |
| Track 1 | 6.18e-4 | 6.41e-4 | **3.33e-4** | 3.40e-4 | 2.97e-4 | 3.39e-4 | 1.30e-3 | 1.06e-3 | 8.39e-4 | **8.95e-4** | 1.01e-3 | 9.72e-4 |
| Track 2 | 6.43e-4 | 4.08e-4 | **2.79e-4** | 4.05e-4 | 3.79e-4 | 2.39e-4 | 9.84e-4 | 7.53e-4 | 5.59e-4 | 6.34e-4 | 6.89e-4 | **6.12e-4** |
| Track 3 | 1.91e-4 | 8.98e-5 | 9.45e-5 | **7.37e-5** | 7.09e-5 | 7.40e-5 | 6.79e-4 | 5.77e-4 | 5.03e-4 | **5.19e-4** | 6.02e-4 | 5.48e-4 |
| Track 4 | 2.52e-4 | 1.17e-4 | 1.12e-4 | **9.05e-5** | 9.10e-5 | 8.18e-5 | 6.84e-4 | 5.13e-4 | 4.42e-4 | **4.63e-4** | 5.37e-4 | 4.77e-4 |
| Track 5 | 2.51e-4 | 2.16e-4 | 2.17e-4 | **2.03e-4** | 2.11e-4 | 1.32e-4 | 4.75e-4 | 3.52e-4 | 3.21e-4 | **3.22e-4** | 3.83e-4 | 3.25e-4 |

(b) Error of the relative quaternion

| Track \ Size | Loss at time $t-(N-1)/2$ | | | | | | Loss at time $t$ | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 11 | 25 | 50 | 75 | 100 | 200 | 11 | 25 | 50 | 75 | 100 | 200 |
| Track 1 | 2.14e-5 | 1.60e-5 | **1.46e-5** | 1.50e-5 | 1.40e-5 | 1.73e-5 | 1.79e-4 | 1.92e-4 | **1.81e-4** | 1.94e-4 | 1.91e-4 | 2.16e-4 |
| Track 2 | 1.93e-5 | 1.61e-5 | 1.43e-5 | **1.25e-5** | 1.24e-5 | 1.64e-5 | **1.24e-4** | 1.37e-4 | 1.24e-4 | 1.35e-4 | 1.30e-4 | 1.46e-4 |
| Track 3 | 1.16e-5 | 6.54e-6 | 5.31e-6 | **5.21e-6** | 4.55e-6 | 6.91e-6 | 1.18e-4 | 1.23e-4 | **1.20e-4** | 1.26e-4 | 1.24e-4 | 1.31e-4 |
| Track 4 | 1.67e-5 | 1.20e-5 | 1.00e-5 | **9.17e-6** | 8.61e-6 | 1.09e-5 | **1.15e-4** | 1.26e-4 | 1.15e-4 | 1.21e-4 | 1.22e-4 | 1.39e-4 |
| Track 5 | 1.28e-5 | 9.13e-6 | 8.09e-6 | **7.59e-6** | 7.58e-6 | 9.25e-6 | 5.67e-5 | 6.62e-5 | **5.84e-5** | 6.13e-5 | 6.12e-5 | 6.98e-5 |

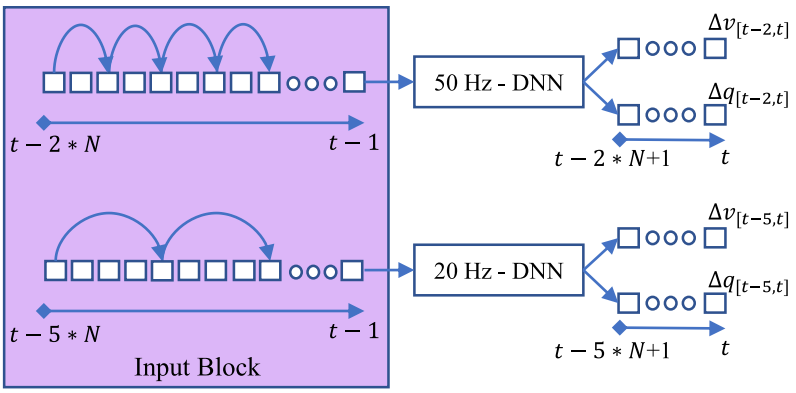



**FIGURE 10.** Neural network architecture for frequency analysis. For better visualization of the experiment realized in Section VII-C, the inputs and the outputs of the 20 Hz and 50 Hz-DNN are detailed.

to achieve outstanding performances. We assume the existence of a minimal frequency where sufficient accuracy can be achieved. Such frequency may depend on the complexity of the motion. Therefore, we investigate the relationship between the frequency and the accuracy for vehicle motion.

The details of the analysis are as follows. The neural network illustrated in Fig. 9 was trained to output $N$ sets of relative velocities and quaternions. The learning process was performed with the KITTI dataset following the data distribution referred in Table. 1. For time-computation reasons, the size of the window was set to 11 in this experiment. In total, four different learning processes were achieved by using the data at 100, 50, 20, and 10 Hz. A window of length $N$ at 100 Hz within the range $[t-N, t-1]$ was fed into the network to output $N$ sets of relative kinematics within the range $[t-N+1, t]$. In other cases, for a frequency $F$, a window of length $N$ covering the range $[t-\frac{100}{F}*N, t-1]$ was fed into the network to output $N$ sets of relative kinematics within the range $[t-\frac{100}{F}*N+1, t]$ at a frequency $F$. For a better understanding, the neural network for both 20 Hz and 50 Hz are illustrated in Fig. 10. In this section, we will refer to the networks as $F$ Hz-DNN.

To compare performances, we computed the prediction errors for both relative velocity and orientation at 10 Hz using (3). Therefore, to obtain relative kinematics at 10 Hz, ten predictions were required from features at 100 Hz, whereas five, two, and one were necessary for features at frequencies 50 Hz, 20 Hz, and 10 Hz. From this premise, five tracks from the KITTI dataset were selected to compute the RMSE between the relative velocity and its ground truth. The orientation metric defined in (5b) without the logarithm term was also considered in this experiment. Both error metrics are shown in Table. 10. Through the testing process, we considered only the prediction at time $t-(N-1)/2$, where features coming from both past and future data were extracted. The tracks used for the testing process of this experiment are

**TABLE 10. Frequency analysis with KITTI dataset. From Fig. 9, four networks with different frequencies such as 10 Hz, 20 Hz, 50 Hz, 100 Hz were trained. The prediction errors of the relative kinematics at 10 Hz are shown in the following table. The best results are indicated with green cells and bold type whereas the worse ones are indicated with red cells. For this analysis, tracks refer to the same drive sequences introduced in Table. 4.**

(a) Average RMSE of the relative velocity

| Track \ Freq. | 10 | 20 | 50 | 100 |
|---|---|---|---|---|
| Track 1 | 2.16e-2 | 1.9e-2 | **4.81e-3** | 5.19e-3 |
| Track 2 | 1.11e-2 | 7.18e-3 | 4.30e-3 | **4.24e-3** |
| Track 3 | 6.61e-3 | 4.31e-3 | 1.38e-3 | **1.03e-3** |
| Track 4 | 6.86e-3 | 4.37e-3 | 2.17e-3 | **1.61e-3** |
| Track 5 | 3.67e-3 | 2.75e-3 | 2.49e-3 | **2.25e-3** |

(b) Average error of the relative quaternion

| Track \ Freq. | 10 | 20 | 50 | 100 |
|---|---|---|---|---|
| Track 1 | 8.34e-4 | 4.79e-4 | 2.06e-4 | **1.81e-4** |
| Track 2 | 6.55e-4 | 3.69e-4 | **2.00e-4** | 2.22e-4 |
| Track 3 | 5.96e-4 | 3.38e-4 | 1.24e-4 | **9.45e-5** |
| Track 4 | 6.25e-4 | 3.45e-4 | **1.76e-4** | 1.81e-4 |
| Track 5 | 3.48e-4 | 2.06e-4 | 1.65e-4 | **1.31e-4** |

**TABLE 11. Frequency analysis with EuRoC dataset. From Fig. 9, three networks with different frequencies such as 50 Hz, 100 Hz, 200 Hz were trained. The prediction errors of the relative kinematics at 10 Hz are shown in the following table. The best results are indicated with green cells and bold type whereas the worse ones are indicated with red cells. For this analysis, tracks refer to the same drive sequences introduced in Table. 4.**

(a) Average RMSE of the relative velocity

| Track \ Freq. | 50 | 100 | 200 |
|---|---|---|---|
| MH_03_medium | 1.02e-2 | 9.06e-3 | **6.66e-3** |
| V2_01_easy | 9.80e-3 | 8.45e-3 | **4.21e-3** |
| V2_03_difficult | 9.15e-3 | 8.38e-3 | **5.14e-3** |
| MH_02_easy | 1.72e-2 | 1.47e-2 | **5.81e-3** |
| MH_04_difficult | 2.30-e2 | 1.96e-2 | **7.71e-3** |
| V2_02_medium | 4.21e-2 | 3.55e-2 | **1.36e-2** |

(b) Average error of the relative quaternion

| Track \ Freq. | 50 | 100 | 200 |
|---|---|---|---|
| MH_03_medium | 1.67e-3 | 1.48e-3 | **2.53e-4** |
| V2_01_easy | 2.19e-3 | 2.14e-3 | **6.77e-4** |
| V2_03_difficult | 2.65e-3 | 2.63e-3 | **1.89e-3** |
| MH_02_easy | 1.39e-3 | 1.23e-3 | **1.64e-4** |
| MH_04_difficult | 1.78e-3 | 1.56e-3 | **2.70e-4** |
| V2_02_medium | 3.27e-3 | 2.98e-3 | **1.04e-3** |

referred in Table. 4. It is worth noting that each learning process was achieved three times, and the average error was considered in our analysis.

### 2) EVALUATION

From the five testing tracks of the KITTI dataset, the worse results were obtained for the 10 Hz and 20 Hz-DNN. As expected for these two cases, the information lost resulted in a deterioration of the relative velocity and quaternion prediction, compared to the original signal at 100 Hz. On the other hand, the prediction accuracy for the 50 Hz-DNN

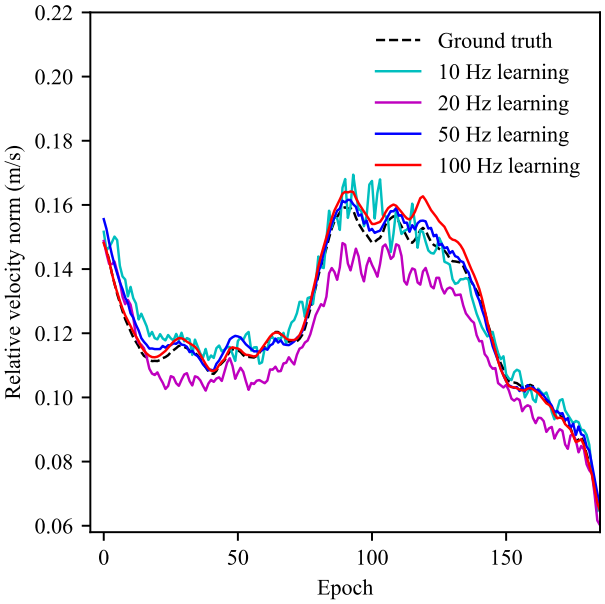



**FIGURE 11. Norm of the predicted relative velocity for different frequencies - Track 1. The prediction of a small part of Track 1 is illustrated to visualize the differences between the four cases.**

could compete with the 100 Hz one. Nevertheless, using data at 50 Hz must have been responsible for a loss of information. Therefore, such information should have been compensated differently. Three hypotheses are established. First, the information held by the signal at 50 Hz may be sufficient to predict the motion of a car with high accuracy. On the contrary, the IO model was too difficult to predict from data at 10 Hz and 20 Hz, as shown in Fig. 11. Another hypothesis concerns the number of predictions required to output a 10 Hz relative kinematics. For the 100 Hz-DNN, ten predictions were necessary, whereas only five values of relative kinematic were used for the data at 50 Hz. Therefore, fewer prediction errors may have occurred in the second case. Furthermore, the distribution of the data in motion space differs for each frequency. Such difference can be correlated to the prediction accuracy, as shown in Section V-D. Finally, for both cases, the time range considered for the input was different. When outputting relative kinematics at 50 Hz, past and future features were considered from a broader range of time. All these hypotheses can justify the resemblance of both 50 Hz and 100 Hz-DNN performances. Nevertheless, the predictions from the original signal were the most accurate. We could expect the accuracy difference between the 100-Hz and the 200 Hz-DNN lower for the IO of a car.

Next, the experiment was extended to EuRoC dataset for 200 Hz, 100 Hz, and 50 Hz in Table. 11. The observation achieved with KITTI dataset differed. For a drone, lower-sampling the original data decreased the performances. The 50 Hz and the 100 Hz-DNN one had difficulties in modeling relative kinematics, as shown in Fig. 12. As expected, the motion space of the drone is wider and more complex to model than the motion space of a car. Therefore, a high-frequency signal that holds more information may be

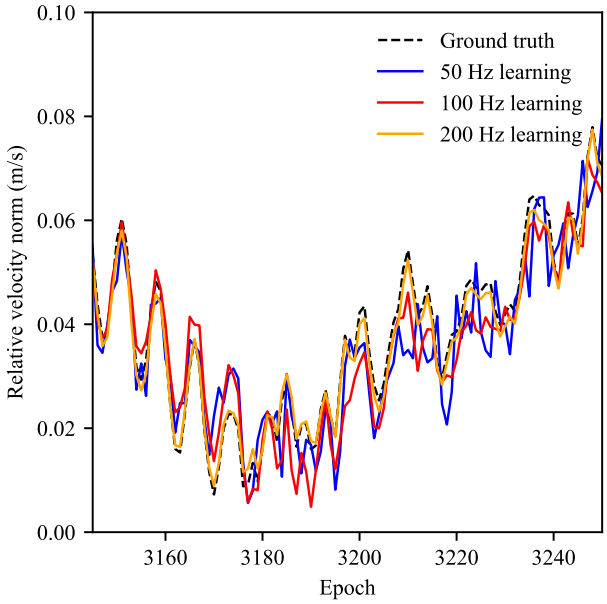



**FIGURE 12.** Norm of predicted velocity for different frequencies - V2_01_easy. The prediction of a small part of V2_01_easy is illustrated to visualize the differences between the three cases.

required for the inertial odometry of a drone. Even though the 100 Hz-DNN showed better results than the 50 Hz one, we can notice their performances were close to each other, whereas both of them are outperformed by the 200 Hz-DNN.

Through this experiment, we could not precisely identify the minimal frequency required to achieve IO. The purpose of this section was to introduce the notion of data frequency for DNN-based IO while respecting the same neural network architecture for each case defined. In future work, we would like to expand this notion to the output frequency only.

### D. LOSS FUNCTION

#### 1) ANALYSIS SETTINGS

The purpose of this section is to analyze different variations of the loss function introduced in (5). For this analysis, the neural network illustrated in Fig. 9 was trained to output $N$ sets of relative velocities and quaternions. The window size was set to 11 due to computational-time reasons. From this architecture choice, the variants of the loss function can be investigated for two different distributions of features, as mentioned in Section VII-A.

The three-axis relative velocity outputted by the network has a shape of $[B, N, 3]$ where $B$ is the batch size and $N$ the window size. The relative quaternion has a shape of $[B, N, 4]$. Through the learning process, the prediction error is computed for each component of the relative velocity within the batch. Then, the mean is considered as the loss function by default. Nevertheless, one would tend to sum up the errors. For example, it would make sense to sum up the error of each relative velocity axis rather than considering the mean, due to some scaling problems. From this premise, we defined four representations of the loss function, which we refer to as MSS, MSM, MMM, and MSM. The first, second, and third characters indicate the mean (M) error of the batch, the mean (M) or the sum (S) of the error along with the window of size $N$, the mean (M) or the sum (S) of the error along with the axis of the relative output, respectively. For better visualization, the loss function of the relative velocity was described in (6). The same pattern was applied to the quaternion loss function defined in (5b) to ensure logic in this analysis.

$$\begin{cases} \mathcal{L}_{MSS_1} = \dfrac{1}{B}\displaystyle\sum_{i=1}^{B}\sum_{j=1}^{N}\sum_{k=1}^{3} error_k & \text{(6a)} \\ \mathcal{L}_{MSM_1} = \dfrac{1}{B}\displaystyle\sum_{i=1}^{B}\sum_{j=1}^{N}\dfrac{1}{3}\sum_{k=1}^{3} error_k & \text{(6b)} \\ \mathcal{L}_{MMM_1} = \dfrac{1}{B}\displaystyle\sum_{i=1}^{B}\dfrac{1}{N}\sum_{j=1}^{N}\dfrac{1}{3}\sum_{k=1}^{3} error_k & \text{(6c)} \\ \mathcal{L}_{MMS_1} = \dfrac{1}{B}\displaystyle\sum_{i=1}^{B}\dfrac{1}{N}\sum_{j=1}^{N}\sum_{k=1}^{3} error_k & \text{(6d)} \end{cases}$$

Next, we considered three error representations for the relative velocity loss to expand our analysis. We investigated the squared error (SE), the absolute error (AE), and the Huber error (HE). In the original loss function (5a), the error was squared to assign a larger weight when a larger error occurs. On the contrary, AE is computed on the same linear scale for each prediction. In other words, AE is used when the data is corrupted with outliers, whereas SE is sensitive to these outliers. Nevertheless, the derivative of AE is not continuous. Thus, the convergence toward the optimal IO model is harder to reach during the learning process. On the other hand, the gradient of SE decreases as the error gets closer to zero, which favors the convergence of the loss function. Huber loss was defined to handle these drawbacks. Its representation is given in (7c). HE relies on a tunable hyper-parameter $\delta$. When the AE of the relative kinematics is inferior to $\delta$, the error acts as the SE and conversely acts as the AE when the error is superior to $\delta$. Therefore, the user can set their own definition of an outlier with $\delta$. By default, we set $\delta$ to one.

$$\begin{cases} SE_k = \|\Delta\hat{\mathrm{v}}_k - \Delta\mathrm{v}_k\|_2 & \text{(7a)} \\ AE_k = \|\Delta\hat{\mathrm{v}}_k - \Delta\mathrm{v}_k\|_1 & \text{(7b)} \\ HE_k = \begin{cases} 0.5 * SE_k & \text{if } AE_k < \delta \\ AE_k - 0.5 & \text{otherwise} \end{cases} & \text{(7c)} \end{cases}$$

It is worth noting that some pre-processing were performed over KITTI dataset to restrict the influence of outliers. As explained in Section IV, data has been linearized and separated into several parts when timestamp issues occurred. Furthermore, the ground truth of the car orientation is originally set between $[-\pi, \pi]$. Thus, the orientation can suddenly change from $\pi$ to $-\pi$ and vice versa. When such change occurs, the relative orientation can be considered as an outlier

**TABLE 12. Comparison of performances for different variations of the loss function. The network in Fig. 9 has been trained with four loss function representations we referred to as MSS, MSM, MMM and MMS. For each representation, three widespread errors, SE, AE, and HE were considered during the learning process. Through the testing process, the prediction errors of the relative kinematics at time $t-(N-1)/2$ and time $t$ are shown in the following table. The best results among the three errors are indicated in green whereas the best results among the four representations are bordered with blue brackets. The best combinations are written in green font and bordered with blue brackets. For this analysis, tracks refer to the same drive sequences introduced in Table. 4.**

(a) Average error of the relative quaternion error

| | | At time $t-(N-1)/2$ | | | | At time $t$ | | | |
|---|---|---|---|---|---|---|---|---|---|
| Track \ Loss | Error | MSS | MSM | MMM | MMS | MSS | MSM | MMM | MMS |
| Track 1 | SE | 2.20e-5 | **2.02e-5** | 2.53e-5 | [1.95e-5] | **1.81e-4** | **1.79e-4** | 1.90e-4 | **[1.78e-4]** |
| | AE | [3.82e-5] | 4.24e-5 | 5.19e-5 | 5.47e-5 | [2.04e-4] | 2.07e-4 | 2.18e-4 | 2.28e-4 |
| | HE | **2.15e-5** | 2.05e-5 | **2.19e-5** | **[1.87e-5]** | 1.92e-4 | 1.87e-4 | **[1.80e-4]** | 1.82e-4 |
| Track 2 | SE | 2.36e-5 | **2.00e-5** | **2.22e-5** | **[1.76e-5]** | **1.27e-4** | **1.24e-4** | 1.30e-4 | **[1.22e-4 ]** |
| | AE | [3.52e-5] | 4.06e-5 | 4.75e-5 | 5.36e-5 | [1.45e-4] | 1.49e-4 | 1.57e-4 | 1.70e-4 |
| | HE | **2.07e-5** | 2.17e-5 | 2.28e-5 | [1.80e-5] | 1.34e-4 | 1.32e-4 | **1.27e-4** | [1.25e-4] |
| Track 3 | SE | 1.28e-4 | 1.10e-5 | 1.52e-5 | [1.08e-5] | **1.19e-4** | **1.18e-4** | 1.23e-4 | **[1.18e-4]** |
| | AE | [2.86e-5] | 3.33e-5 | 3.81e-5 | 4.18e-5 | 1.36e-4 | [1.36e-4] | 1.38e-4 | 1.51e-4 |
| | HE | **1.10e-5** | **1.01e-5** | **1.27e-5** | **[9.41e-6]** | 1.23e-4 | 1.21e-4 | **1.20e-4** | [1.19e-4] |
| Track 4 | SE | 1.97e-5 | **1.65e-5** | 2.01e-5 | **[1.47e-5]** | **1.17e-4** | **1.14e-4** | 1.23e-4 | **[1.13e-4]** |
| | AE | [3.50e-5] | 3.97e-5 | 4.67e-5 | 5.19e-5 | [1.36e-4] | 1.39e-4 | 1.48e-4 | 1.59e-4 |
| | HE | **1.85e-5** | 1.73e-5 | **1.92e-5** | [1.49e-5] | 1.25e-4 | 1.21e-4 | **1.17e-4** | [1.15e-4] |
| Track 5 | SE | 1.37e-5 | 1.25e-5 | 1.50e-5 | [1.24e-5] | **5.73e-5** | **5.72e-5** | 6.35e-5 | **[5.64e-5]** |
| | AE | [2.87e-5] | 3.46e-5 | 3.96e-5 | 4.13e-5 | [7.65e-5] | 8.02e-4 | 8.29e-5 | 8.70e-5 |
| | HE | **1.25e-5** | **1.18e-5** | **1.44e-5** | **[1.10e-5]** | 6.41e-5 | 6.08e-5 | **5.87e-5** | [5.77e-5] |

(b) Average RMSE of the relative velocity

| | | At time $t-(N-1)/2$ | | | | At time $t$ | | | |
|---|---|---|---|---|---|---|---|---|---|
| Track \ Loss | Error | MSS | MSM | MMM | MMS | MSS | MSM | MMM | MMS |
| Track 1 | SE | 6.31e-4 | [5.42e-4] | 6.82e-4 | 6.78e-4 | 1.26e-3 | [1.10e-3] | 1.41e-3 | 1.17e-3 |
| | AE | **4.80e-4** | **[3.26e-4]** | **3.48e-4** | **4.50e-4** | **9.94e-4** | **[8.92e-4]** | **9.23e-4** | **9.76e-4** |
| | HE | 7.54e-4 | 6.39e-4 | 9.09e-4 | [5.24e-4] | 1.28e-3 | [9.95e-4] | 1.25e-3 | 1.05e-3 |
| Track 2 | SE | **[4.84e-4]** | 5.24e-4 | 7.54e-4 | 5.24e-4 | 8.60e-4 | 8.68e-4 | 1.04e-3 | [7.83e-4] |
| | AE | 5.47e-5 | **[4.23e-4]** | **5.40e-4** | **4.70e-4** | **8.25e-4** | **[7.04e-4]** | **7.79e-4** | **7.53e-4** |
| | HE | 7.36e-4 | [5.12e-4] | 5.96e-4 | 5.33e-4 | 1.01e-3 | [7.73e-4] | 9.01e-4 | 8.17e-4 |
| Track 3 | SE | 1.69e-4 | 1.48e-4 | 2.47e-4 | [1.48e-4] | 6.43e-4 | [5.85e-4] | 7.82e-4 | 6.31e-4 |
| | AE | **[1.42e-4]** | 1.64e-4 | **1.75e-4** | 1.72e-4 | **5.87e-4** | [5.64e-4] | **6.35e-4** | 5.94e-4 |
| | HE | 1.82e-4 | **1.44e-4** | 1.94e-4 | **[1.34e-4]** | 6.75e-4 | **[5.45e-4]** | 6.60e-4 | **5.92e-4** |
| Track 4 | SE | 2.63e-4 | [1.99e-4] | 3.13e-4 | 2.11e-4 | 6.51e-4 | [5.40e-4] | 7.84e-4 | 5.97e-4 |
| | AE | **2.03e-4** | **[1.92e-4]** | **2.23e-4** | 2.73e-4 | **5.55e-4** | [5.20e-4] | **5.56e-4** | 5.69e-4 |
| | HE | 2.37e-4 | 1.99e-4 | 2.89e-4 | **[1.85e-4]** | 6.55e-4 | **[4.97e-4]** | 6.68e-4 | **5.44e-4** |
| Track 5 | SE | 2.55e-4 | [2.39e-4] | 2.82e-4 | 2.42e-4 | 4.71e-4 | [3.69e-4] | 5.68e-4 | 4.13e-4 |
| | AE | **[2.41e-4]** | 2.58e-4 | **2.49e-4** | 2.90e-4 | **3.82e-4** | 4.36e-4 | **[3.76e-4]** | 4.00e-4 |
| | HE | 2.48e-4 | **2.37e-4** | 2.76 e-4 | **[2.33e-4]** | 4.74e-4 | **[3.40e-4]** | 4.79e-4 | **3.69e-4** |

by the DNN. Therefore, the initial range of orientation was extended.

For each case defined for this analysis, the learning process was achieved three times, and the average error at time $t-(N-1)/2$ and time $t$ was considered. Through the testing process, the relative quaternion error without the logarithm defined in (5b) was used as the orientation metric in Table. 12a. The RMSE of the relative velocity was also considered in Table. 12b.

### 2) EVALUATION

Among the four representations tested, MSM and MMS showed better performances, whereas MMM provided the worse accuracy for both relative kinematics. Regarding the error representation, AE outperformed HE and SE for the relative velocity prediction only. However, its usage deteriorated the prediction of the relative quaternion, compared with the two others. On the other hand, the results obtained from HE and SE were equivalent for both regression tasks. Nonetheless, HE slightly showed higher accuracy. With a better optimization of the $\delta$ hyper-parameter, the performances of HE may be improved. It is worth noting that MSE is the most common loss function in deep learning by default. In other words, the combination of the MMM representation and the SE is mostly used. However, we demonstrated that we could outperform this configuration.

From this analysis, one overall conclusion can be drawn. We may need to conclude that there is no optimal answer to the question of which loss function suits the best for IO problems. According to the DNN architecture and the loss representation, higher or lower accuracy can be obtained. As aforementioned, the purpose of this section is to offer an

overview of our loss function and combination possibilities with quantitative analysis. Even though everything has not been tested, we reaffirm the difficulty of generalizing the loss function. Furthermore, we demonstrated the loss function could be considered as a DNN hyper-parameter where few adjustments could achieve better performances. In future work, the setting of the Huber loss function will be investigated to improve the performances of our network.

### E. REPRESENTATION OF THE RELATIVE QUATERNION

#### 1) ANALYSIS SETTINGS

Two quaternion representations are analyzed in this section. In [56], the relative quaternion is represented as the difference between two successive quaternions whereas the Hamilton product was adopted in [57]. We refer to these two representations as to the difference quaternion (DQ) and the Hamilton quaternion (HQ), respectively. The purpose of this section is to investigate the characteristics of each representation for the IO problems.

The details of the analysis are as follows. The CRK defined in Section V-A was trained with KITTI dataset to output $N$ sets of relative velocities and quaternions with $N = 11$. From this architecture choice, the representation of the relative quaternion can be investigated for two different distributions of features, as mentioned in Section VII-A. Originally, HQ was outputted by our neural network and the loss function defined in (5) was used during the learning process. We refer to this case as HQ-L. However, the orientation metric defined in (5b) was not adequate for the DQ representation. Therefore, the orientation metric was replaced with the SE. We refer to this case as DQ-SE. From the same premise, we define HQ-SE as the case where HQ is outputted and the SE was used as a metric. When the SE was used, each component of the outputted relative quaternion was normalized within the range $[-1, 1]$ to prevent scaling issues.

For each case defined for this analysis, the learning process was achieved three times with the MMS and MSM configurations and the average error at time $t - (N - 1)/2$ and time $t$ was considered. We analyzed the accuracy of the prediction at 10 Hz for five tracks, where ten predictions were summed up for DQ representation, and ten predictions were multiplied using (3a) for HQ representation. We expected the differences between the three defined loss functions to be more perceptible with this error representation. The evaluation was based on two orientation metrics: the RMSE, and the quaternion error without the logarithm term defined in (5b). The results are described in Table. 13a. On the other hand, we granted less importance to the accuracy of the velocity prediction. Therefore, we analyzed the RMSE at 100 Hz in Table. 13b. The tracks used for the testing process of this experiment are referred in Table. 4.

#### 2) EVALUATION

For this analysis, we originally used the NRK defined in Fig. 9 to output relative quaternion. However, the NRK was not able to predict DQ representation from features of angular velocity and timestamp interval only. The neural network could only model the fourth component of the relative quaternion where gyroscope features played a major role in the prediction. Nonetheless, the estimation was far from the ground truth. On the other hand, the NRK could predict HQ representation, affirming our architecture learnt to model the kinematics. However, the output had to be normalized in the case of HQ-SE. To have a fair comparison between the two representations, we used the CRK to predict relative kinematics with additional inputs. This time, the neural network architecture could model the DQ representation. As in [56], the use of the previous quaternion considerably improved the accuracy of DQ prediction.

Through the testing process, the HQ representation provided a better orientation estimation than the DQ one. Theoretically, HQ is determined from angular velocity and timestamp, as described in (3a). However, such a direct correlation is nonexistent for DQ representation. Therefore, its estimation was not possible with the NRK where only features coming from gyroscope and timestamp were used. Nevertheless, its prediction was possible with the help of previous quaternion features, which must have played a major role. Conversely, the features of the previous quaternion played a minor role in the HQ prediction.

The SE error used for HQ-SE and DQ-SE cases is not correlated to the IO problems. During the learning process, the error of each component is minimized independently. When a loss function with a higher comprehension of the quaternion representation is used such as in HQ-L case, better performances are achieved. The same accuracy improvement was observed in [57]. Nevertheless, a minor drawback was observed. The first component of the relative quaternion is not used directly in the orientation metric defined in (5b). Its prediction is used to normalize the relative quaternion outputted by the network and the loss function is mainly computed from the three other components of the relative quaternion. Therefore, the accuracy of the prediction of the first component is negatively impacted. Nonetheless, the first component of the quaternion is set to one in (4). Thus, the DNN can be designed to output only the three other components of HQ representation. In the HQ-SE case, the prediction of the first component was better but could have been responsible for the deterioration of the prediction.

From this analysis, we may conclude that HQ-L suits the best to our neural network architecture. Furthermore, this section extends the conclusion drawn in Section VII-D. The loss function should be considered as a hyper-parameter that is highly correlated to the navigation problem.

### F. USE OF FUTURE DATA

#### 1) ANALYSIS SETTINGS

According to (3), the previous state is required in the double integration process. However, inherent problems associated with acceleration and gyroscope impact negatively the overall

**TABLE 13. Evaluation of the relative quaternion representation with KITTI dataset. The CRK defined in Table. 5 has been trained to output relative kinematics with two different representations, DQ and HQ. HQ-SE and DQ-SE refer to the cases where the loss function defined in (5b) was replaced with the SE to output HQ and DQ, respectively. In the case of HQ-L, the network was trained to output HQ and the loss function defined in (5) was used during the backpropagation. The prediction errors of the relative kinematics at time $t-(N-1)/2$ and $t$ are displayed in the following table. The best results between these three cases are indicated in green. For this analysis, tracks refer to the same drive sequences defined in Table. 4.**

(a) Average orientation metrics at 10 Hz.

| | | At time $t-(N-1)/2$ | | | | At time $t$ | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | MSM | | MMS | | MSM | | MMS | |
| Track | Loss | QuatError | RMSE | QuatError | RMSE | QuatError | RMSE | QuatError | RMSE |
| Track 1 | DQ-SE | 2.46e-4 | 5.95e-5 | 3.11e-4 | 7.47e-5 | 1.19e-3 | 2.29e-4 | 1.28e-3 | 2.43e-4 |
| | HQ-SE | 2.51e-4 | 5.78e-5 | 2.54e-4 | 5.74e-5 | 1.14e-3 | 2.18e-4 | 1.14e-3 | 2.16e-4 |
| | HQ-L | **1.54e-4** | **3.31e-5** | **1.45e-4** | **3.11e-5** | **9.94e-4** | **1.94e-4** | **1.00e-3** | **1.96e-4** |
| Track 2 | DQ-SE | 3.27e-4 | 1.37e-4 | 3.40e-4 | 1.92e-4 | 9.66e-4 | 2.51e-4 | 9.42e-4 | 3.56e-4 |
| | HQ-SE | 2.27e-4 | 5.63e-5 | **2.18e-4** | **5.28e-5** | 8.08e-4 | 1.62e-4 | **8.04e-4** | **1.61e-4** |
| | HQ-L | **1.76e-4** | **4.86e-5** | 2.84e-4 | 1.25e-4 | **7.20e-4** | **1.45e-4** | 8.19e-4 | 1.85e-4 |
| Track 3 | DQ-SE | 1.03e-4 | 2.29e-5 | 8.04e-5 | 1.83e-5 | 6.58e-4 | 1.39e-4 | **6.55e-4** | **1.40e-4** |
| | HQ-SE | 1.02e-4 | 2.22e-5 | 1.10e-4 | 2.37e-5 | 6.58e-4 | **1.38e-4** | 7.01e-4 | 1.45e-4 |
| | HQ-L | **5.91e-5** | **1.15e-5** | **4.86e-5** | **9.88e-6** | **6.49e-4** | 1.44e-4 | 6.58e-4 | 1.45e-4 |
| Track 4 | DQ-SE | 1.67e-4 | 4.55e-5 | 1.69e-4 | 4.70e-5 | 7.02e-4 | 1.39e-4 | 7.19e-4 | 1.43e-4 |
| | HQ-SE | 1.54e-4 | **3.61e-5** | **1.62e-4** | **3.80e-5** | 6.90e-4 | 1.34e-4 | 7.17e-4 | **1.38e-4** |
| | HQ-L | **1.25e-4** | 3.81e-5 | 1.89e-4 | 8.70e-5 | **6.42e-4** | **1.27e-4** | **7.10e-4** | 1.54e-4 |
| Track 5 | DQ-SE | 1.32e-4 | 3.37e-5 | 1.43e-4 | 3.61e-5 | 4.35e-4 | 8.92e-5 | 4.40e-4 | 9.27e-5 |
| | HQ-SE | 1.33e-4 | 3.47e-5 | 1.35e-4 | 3.41e-5 | 4.41e-4 | 9.12e-5 | 4.60e-4 | 9.42e-5 |
| | HQ-L | **9.24e-5** | **2.65e-5** | **8.57e-5** | **2.47e-5** | **3.70e-4** | **7.45e-5** | **3.80e-4** | **7.69e-5** |

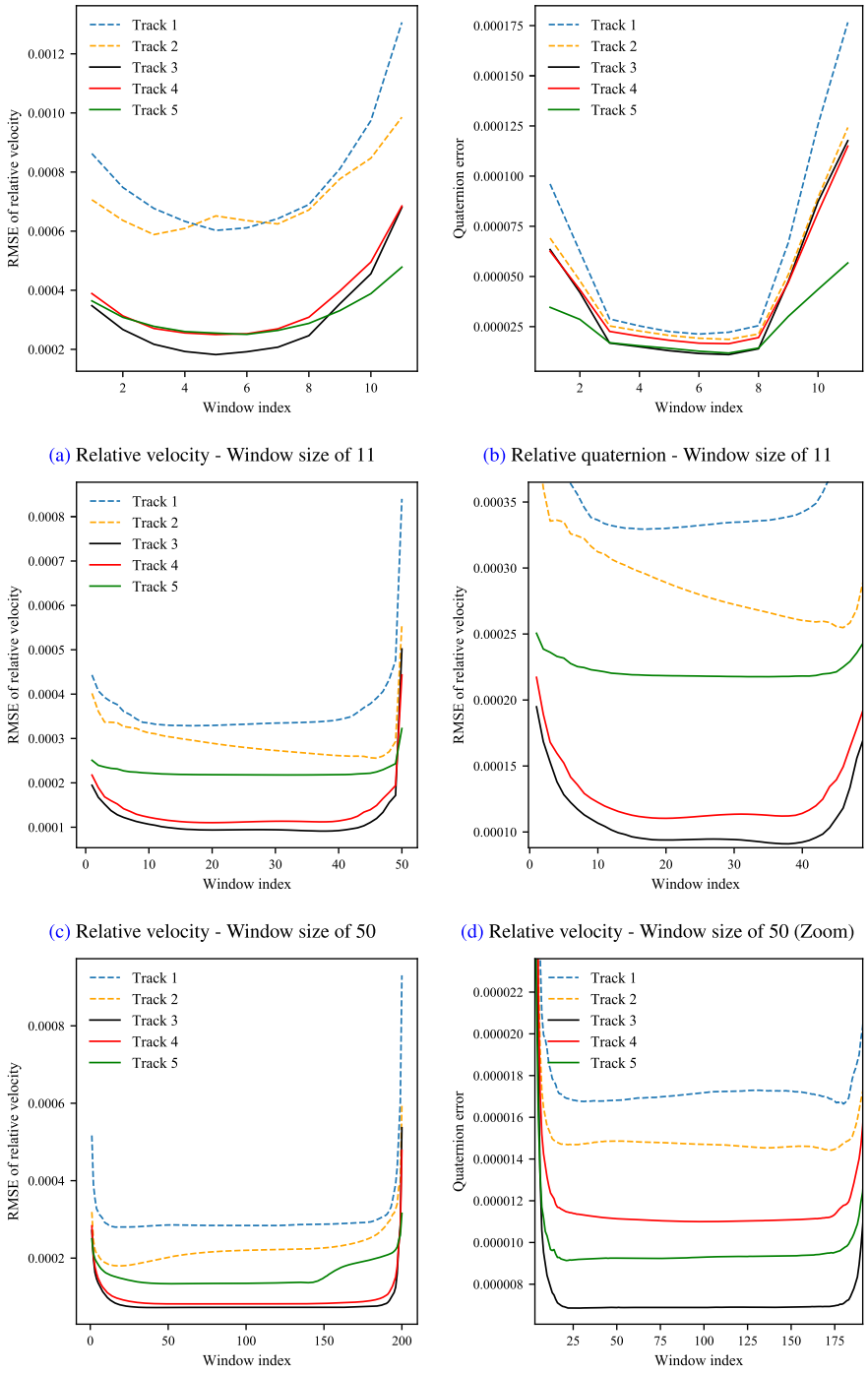
(b) Average RMSE of the relative velocity at 100 Hz

| | | At time $t-(N-1)/2$ | | At time $t$ | |
|---|---|---|---|---|---|
| Track | Loss | MSM | MMS | MSM | MMS |
| Track 1 | DQ-SE | **3.32e-4** | **3.86e-4** | 9.24e-4 | 1.01e-3 |
| | HQ-SE | 4.58e-4 | 4.14e-4 | 9.40e-4 | 9.05e-4 |
| | HQ-L | 4.89e-4 | 4.29e-4 | **9.10e-4** | **8.85e-4** |
| Track 2 | DQ-SE | **4.20e-4** | 4.39e-4 | 7.10e-4 | 8.00e-4 |
| | HQ-SE | 4.29e-4 | 4.91e-4 | **6.70e-4** | 7.47e-4 |
| | HQ-L | 4.21e-4 | **4.19e-4** | 6.90e-4 | **6.97e-4** |
| Track 3 | DQ-SE | 1.18e-4 | 1.32e-4 | 5.64e-4 | 6.34e-4 |
| | HQ-SE | 1.37e-4 | 1.43e-4 | 5.40e-4 | 5.75e-4 |
| | HQ-L | **1.07e-4** | **1.08e-4** | **5.02e-4** | **5.41e-4** |
| Track 4 | DQ-SE | 1.54e-4 | 1.65e-4 | 5.14e-4 | 5.70e-4 |
| | HQ-SE | 1.61e-4 | 1.80e-4 | 4.80e-4 | 5.12e-4 |
| | HQ-L | **1.30e-4** | **1.36e-4** | **4.36e-4** | **4.69e-4** |
| Track 5 | DQ-SE | **2.35e-4** | **2.36e-4** | 3.65e-4 | 3.95e-4 |
| | HQ-SE | 2.55e-4 | 3.02e-4 | 3.43e-4 | 4.00e-4 |
| | HQ-L | 2.37e-4 | 2.40e-4 | **3.12e-4** | **3.31e-4** |

performances of such a process. From this assessment, previous works resorted to deep learning with multiple layers to extract higher-level latent features coming from past data to estimate and compensate the noise and bias defined in (2). Similarly to the text recognition field, the use of features coming from future data provides a higher understanding and achieves better modeling of the problem. However, this accuracy improvement costs a prediction delay. The time length to be considered from past and future data has not been introduced in the field of IO. Therefore, the influence of the distribution between past and future features over the performances is analyzed in this section.

The details of the analysis are as follows. Since the distribution of features is analyzed, the neural network in Fig. 9 was selected and trained by using a window size $N$ to predict $N$ sets of relative velocities and quaternions. Through the testing process, a data sequence with a length $L$ was selected. At the first iteration, the input window of size $N$, covering the range $[0, N]$, was fed into the network to estimate $N$ relative kinematics at time $[1, N+1]$. At the second iteration, the window was shifted by one sample to cover the range $[1, N+1]$ and was fed into the network to output $N$ relative kinematics at time $[2, N+2]$. At the end of the process, each relative kinematics within the range $]N, L-N]$ was estimated $N$ times with different index position within the output window. The neural network architecture is made of bidirectional LSTM. Therefore, the index position within the window indicates the distribution between past and future features that were used for the relative kinematic prediction. For the first indexes of the window, future features were mainly used to predict relative kinematics whereas last indexes rely on past features.

From this premise, we computed the average RMSE between relative velocity and its ground truth according to the index position within the output window for five tracks of KITTI dataset. The tracks used during the testing process of this experiment are referred in Table. 4. For this analysis, different window sizes were tested and their performances were illustrated in Fig. 13.




(a) Relative velocity - Window size of 11

(b) Relative quaternion - Window size of 11

(c) Relative velocity - Window size of 50

(d) Relative velocity - Window size of 50 (Zoom)

(e) Relative velocity - Window size of 200

(f) Relative quaternion - window size of 200 (Zoom)

**FIGURE 13. Distribution analysis of future and past features. The network illustrated in Fig. 9 was trained with different window sizes of inputs. The accuracy of the prediction was analyzed based on the index position within the window that reflects the distribution between past and future features used for the estimation of relative kinematics.**

### 2) EVALUATION

For both regression tasks, the same behavior was observed whatever the window size tested. Therefore, the performances of our network may be correlated to the distribution between past and future features rather than the number of future features used. When future or past features are ubiquitous compared to the overall distribution, the accuracy drastically decreases. The deterioration especially happened when past features were a majority. The accuracy of the prediction tended to be stable when the position was located around the

middle of the window. In other words, the performances were close to the optimal one when the distribution between past and future features was equivalent. Nonetheless, the optimal distribution could not be generalized for this analysis. Based on the randomness effect of the learning process, the track tested, the metric analyzed, observation slightly differs. For example, Track 2 from the validation dataset reached its optimal accuracy when the distribution between future and past features was unbalanced.

From this analysis, we conclude that a balanced distribution between past and futures features provides good performances for our architecture. The same behavior has been observed with several architectures of CRK introduced in this paper. Nevertheless, the most optimal distribution seems to be highly correlated to the motion of the vehicle, the learning process and the architecture of the DNN. Therefore, such a hyper-parameter could not be generalized from this analysis. In future work, the correlation between the motion and the distribution of the features over the performances could be an interesting trail to be investigated.

### G. INPUT NORMALIZATION

#### 1) ANALYSIS SETTINGS

The scale of inputs and outputs used to train a DNN is an important aspect to be considered. A large range of input can result in large weight values, making the learning process unstable and slow. Moreover, input variables can have different scale and unit, making the update weight process different for each component which might hurt the learning process. Normalization and standardization techniques are applied to ensure an optimal learning process to solve this issue. However, the best way to scale input data is not absolute because the problem is complex. For example, standardizing features by removing the mean and scaling to unit variance, as shown in (8a), might behave negatively if the input x does not respect a Gaussian distribution.

$$\begin{cases} Std(\mathrm{x}) = \sqrt{\dfrac{1}{N-1}\sum_{i=0}^{N}(\mathrm{x}_i - \bar{\mathrm{x}})^2} & \text{(8a)} \\ N(\mathrm{x})_{[0,1]} = \dfrac{\mathrm{x} - \mathrm{x}_{min}}{\mathrm{x}_{max} - \mathrm{x}_{min}} & \text{(8b)} \\ N(\mathrm{x})_{[-1,1]} = 2 * N(\mathrm{x})_{[0,1]} - 1 & \text{(8c)} \end{cases}$$

Another good rule of thumb is to normalize every data between $[-1, 1]$ or $[0, 1]$ to have a similar scale among inputs and avoid mathematical artefacts associated with floating numbers. However, scaling might be unnecessary when the data range belongs to such intervals or has a limited distribution. Given these uncertainties, it is important to investigate the beneficial differences in the performances of these techniques for different inputs correlated to the IO problems.

The CRK defined in Table. 5 was used to output the relative kinematic at time $t$. From this architecture choice, the different techniques of normalization can be investigated for five inputs representative of the IO. For this analysis,

**TABLE 14. Analysis of input normalization. The neural network illustrated in Fig. 5 was trained to output relative kinematics. Based on (8), each input was normalized as follows. A white cell indicates the input was not normalized. Red, blue and green cells with the label '*Std*', '$N_{[0,1]}$', '$N_{[-1,1]}$' refer to as standardization, normalization between [0,1] and normalization between [-1,1], respectively.**

| Input / No. | $\mathbf{a}_{t-1}^{IMU}$ | $\mathbf{w}_{t-1}^{IMU}$ | dt | $\mathbf{q}_{t-1}^{IMU}$ | $\mathbf{v}_{t-1}^{IMU}$ |
|---|---|---|---|---|---|
| E-0 | | | | | |
| E-1 | *Std* | *Std* | *Std* | *Std* | *Std* |
| E-2 | *Std* | *Std* | | *Std* | *Std* |
| E-3 | *Std* | *Std* | | | *Std* |
| E-4 | *Std* | *Std* | | *Std* | |
| E-5 | *Std* | *Std* | | | |
| E-6 | $N_{[0,1]}$ | $N_{[0,1]}$ | $N_{[0,1]}$ | $N_{[0,1]}$ | $N_{[0,1]}$ |
| E-7 | $N_{[0,1]}$ | $N_{[0,1]}$ | | $N_{[0,1]}$ | $N_{[0,1]}$ |
| E-8 | $N_{[0,1]}$ | $N_{[0,1]}$ | | | $N_{[0,1]}$ |
| E-9 | $N_{[0,1]}$ | $N_{[0,1]}$ | | $N_{[0,1]}$ | |
| E-10 | $N_{[0,1]}$ | $N_{[0,1]}$ | | | |
| E-11 | $N_{[-1,1]}$ | $N_{[-1,1]}$ | $N_{[-1,1]}$ | $N_{[-1,1]}$ | $N_{[-1,1]}$ |
| E-12 | $N_{[-1,1]}$ | $N_{[-1,1]}$ | | $N_{[-1,1]}$ | $N_{[-1,1]}$ |
| E-13 | $N_{[-1,1]}$ | $N_{[-1,1]}$ | | | $N_{[-1,1]}$ |
| E-14 | $N_{[-1,1]}$ | $N_{[-1,1]}$ | | $N_{[-1,1]}$ | |
| E-15 | $N_{[-1,1]}$ | $N_{[-1,1]}$ | | | |

16 experiments from E-0 to E-15 were designed, as referred in Table. 14. E-0 refers to the case where inputs were not normalized. For computational-time reasons, the size of the input window was set to 11.

From this premise, the 72 sequences referred in Table. 1 were selected to compute the RMSE between the relative velocity and its ground truth. The orientation metric defined in (5b) without the logarithm term was also considered. Both metrics are described in Table. 15. It is worth noting that each learning process was achieved four times and the average error was considered.

#### 2) EVALUATION

For E-0, three of the four learning process achieved satisfying results. The fourth one faced instability and could not learn properly the kinematic model. For this reason, the average error displayed in Table. 15 was negatively impacted and became the worst one of all experiments performed. Therefore, input data should be normalized.

Normalizing inputs does not necessarily end up with good performances. Since quaternion components, timestamp or previous absolute velocity did not follow Gaussian distribution, we expected them not to be eligible for standardization. However, the results tended to the conclusion that we should not standardize any inputs even for IMU sensors, leading to a quick convergence of the learning process and a deterioration of the accuracy. Nevertheless, standardization techniques have been used in [16] and showed interesting results. Thus, the homogeneity between the inputs may be a criterion to consider while dealing with DNN.

The best accuracy was obtained with input normalization between $[0, 1]$, which led to an improvement of the performances for both kinematic predictions. The normalization between $[-1, 1]$ provided satisfying results but was still outperformed. It is worth noting that standardizing quaternion drastically decreased the performances. For the two

**TABLE 15. Results of experiments in Table. 14. The average RMSEs of the relative velocity and quaternion error at time $t$ over 72 sequences are described in this table. The data distribution is referred in Table. 1. Among the 16 experiments achieved, the worse predictions are highlighted with a red cell whereas the best ones are shown in green. The second best results are written in bold character.**

| Loss / No. | Velocity RMSE | | Relative quaternion error | | |
|---|---|---|---|---|---|
| | Training | Validation | Training | Validation | Epochs |
| E-0 | 5.717e-4 | 1.097e-3 | 1.512e-4 | 1.595e-4 | 93 |
| E-1 | 7.298e-4 | 1.111e-3 | 1.336e-4 | 1.496e-4 | 38 |
| E-2 | 7.309e-4 | 1.077e-3 | 1.338e-4 | 1.522e-4 | 42 |
| E-3 | 5.645e-4 | 1.096-e3 | 1.209e-3 | 1.378e-4 | 72 |
| E-4 | 6.814e-4 | 1.126e-3 | 1.338e-4 | 1.491e-4 | 44 |
| E-5 | 5.942e-4 | 1.126e-3 | 1.216e-4 | 1.363e-4 | 85 |
| E-6 | 5.689e-4 | 1.007e-3 | 1.208e-4 | 1.308e-4 | 151 |
| E-7 | 5.634e-4 | 1.003e-3 | 1.2174e-4 | 1.286e-4 | 173 |
| E-8 | 5.621e-4 | 8.670e-4 | **1.206e-4** | **1.303e-4** | 165 |
| E-9 | **5.376e-4** | 1.050e-3 | 1.230e-4 | 1.308e-4 | 155 |
| E-10 | 5.484e-4 | **8.750e-4** | 1.224e-4 | 1.312e-4 | 132 |
| E-11 | 6.273e-4 | 9.968e-4 | 1.276e-4 | 1.401e-4 | 66 |
| E-12 | 5.985e-4 | 9.757e-4 | 1.214e-4 | 1.395e-4 | 67 |
| E-13 | 5.826e-4 | 9.562e-4 | 1.206e-4 | 1.337e-4 | 79 |
| E-14 | 6.661e-4 | 1.0059e-3 | 1.243e-4 | 1.396e-4 | 74 |
| E-15 | 5.496e-4 | 1.014e-3 | 1.189e-4 | 1.329e-4 | 87 |

other normalization techniques, the performances slightly increased when the previous quaternion was not normalized.

From this analysis, we could not generalize the most optimal normalization technique for IO problems. However, we could affirm the normalization between [0, 1] suits well to our architecture. Therefore, we confirmed the importance of input normalization. As a reminder, feature normalization is not exclusive to inputs and can be overextended to other layers of the neural network. A recent method relies on batch normalization layers to standardize the output of a previous activation layer and ensure independence between convolution layers [62].

For regression problems, scaling the output is usually not necessary. In the case of multiple regression, the learning process will be driven by the output with the largest scale if proper weights were not introduced in the cost function and ground truth have a different dynamic range. The relative quaternion had to be normalized when the SE error was used as a loss function in Section VII-E. Besides, a large output range can result in a large gradient error making the weight update unstable.

### *H. DATA BALANCE*

#### 1) OVERSAMPLING AND UNDERSAMPLING

Data balancing notion is usually associated with classification problems but is just as important for regression tasks. The DNN must be able to predict the low and high change in velocity with the same accuracy. During the learning process, if high relative velocity is represented as a minority, the DNN will mainly build up its prediction using abundant samples and will consider the rare samples as outliers, leading to poor accuracy, as illustrated in Fig. 14. The same applies to the prediction of the relative quaternion. However, the available dataset for IMU measurements are limited and prevent to maintain a good data distribution.

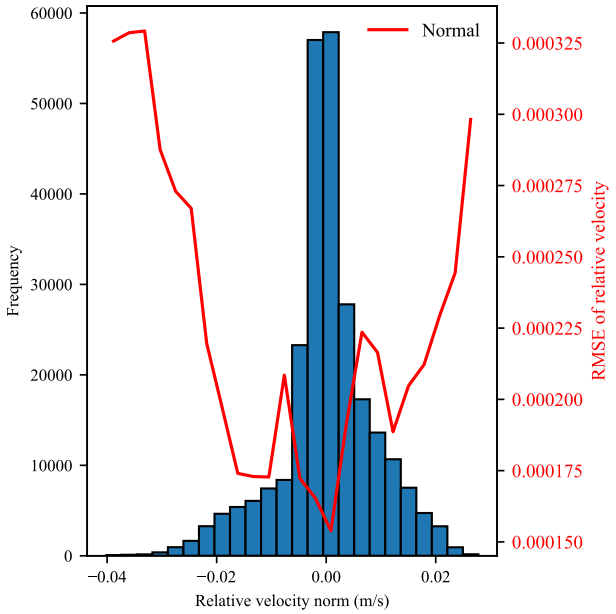



**FIGURE 14. Analysis of the correlation between the histogram of the relative velocity and the performances. The network illustrated in Fig. 5 was trained to output relative kinematics at time $t - (N-1)/2$. For each range of the relative velocity norm, the RMSE of the relative velocity was shown in red. The distribution of the KITTI training dataset is illustrated in blue.**

Therefore, oversampling and undersampling techniques can be used to balance the data. A straightforward oversampling method can be defined as copying, generating random data from the minority class, whereas undersampling will reduce abundant observations. Nevertheless, they can lead to overfitting or underfitting, respectively.

As a preliminary analysis, we decided to oversample and undersample the training dataset of KITTI, as illustrated in Fig. 15. Based on the distribution of the relative velocity norm, samples were either copied or removed. For each copied data, we added a small Gaussian noise with a standard deviation of $10^{-4}$ to the IMU inputs. For this analysis, we trained the NRK defined in Fig. 5 to output relative kinematics at time $t - (N-1)/2$. From this architecture choice, the balancing problem becomes easier to deal with when only one relative kinematic value is outputted. The window size was set to 11 due to computational-time reasons. The RMSE of the relative velocity for the training and validation dataset was illustrated in Fig. 16. It is worth noting that each learning process was achieved three times, and the average error was considered.

We expected a better prediction for the under-represented relative velocity in exchange for lower accuracy for the abundant samples. The proposed approach had the effect we expected, but only for the training dataset. For the validation dataset, we observed a deterioration of the prediction for rare samples, whereas performances were similar for abundant samples. Therefore, overfitting may have occurred despite the use of dropout and early stopping technique. For this analysis, two ranges of relative velocity norm have been down-sample by half. Nonetheless, the performances were similar to the

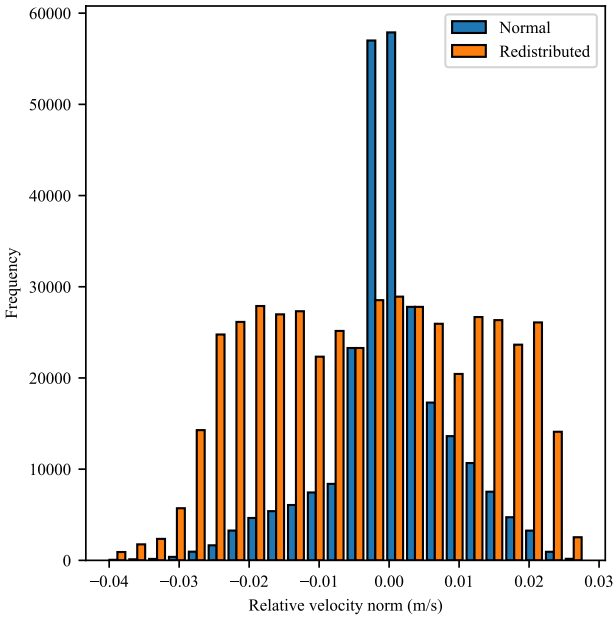



**FIGURE 15.** Redistribution of the KITTI dataset. From the original data distribution of relative velocity norm, oversampling and undersampling have been applied to the training dataset. The redistributed histogram is illustrated in orange whereas the normal one in blue.

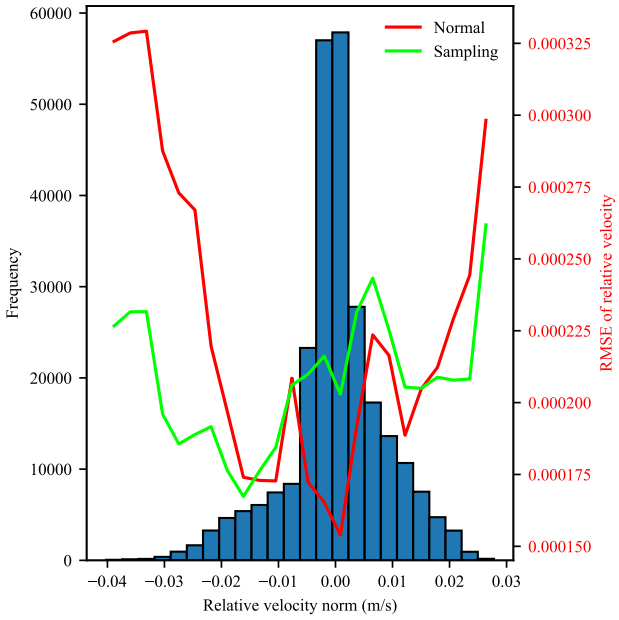
normal learning process. Therefore, undersampling can be worth considering with caution when data is abundant in the dataset.

Oversampling and undersampling techniques are not commonly used when dealing with data balancing. One can lead to overfitting, whereas the other one can lead to a loss of information. The purpose of this section was to propose a preliminary quantitative analysis of the problem of data balancing for DNN-based IO. In addition to the limit previously introduced, we highlight the difficulty of data balance. In this section, the dataset has been redistributed based on the relative velocity norm, whereas orientation has been ignored in the process. Therefore, balancing one relative kinematic may hurt the balance of the other one. This can be drawn as a limit to our architecture. Furthermore, balancing data entails the use of additional hyper-parameters that need to be defined, making the DNN optimization process longer and more difficult.

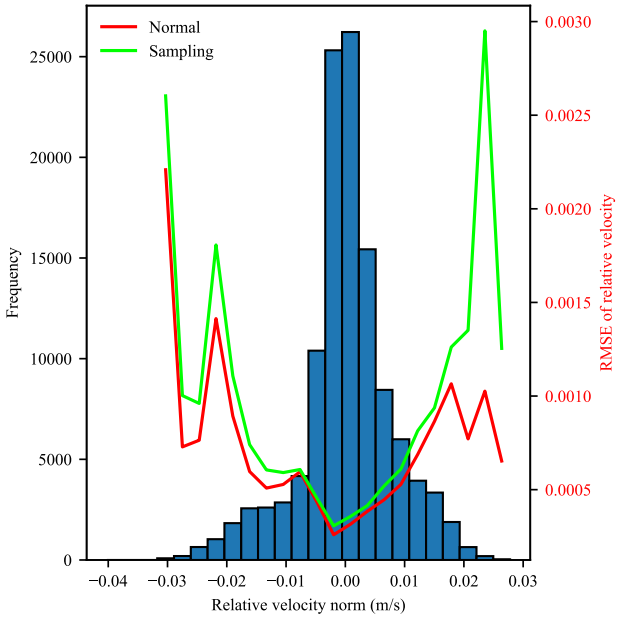
### 2) RELEVANCE FUNCTION

Another strategy for handling imbalanced data is to design a cost function that penalizes regression based on a relevance function (RF). The influence of rare samples is boosted during loss function computation, whereas abundant data will have a minor penalty cost. However, such a function is highly correlated to the problem we are dealing with and can be difficult to design, especially when the data distribution is difficult to visualize, as similar to the quaternion representation.

For this second analysis for data balancing, we decided to apply a weight to three velocities-axis during the loss function computation to improve the prediction of high relative speed. First, we divided each axis of the relative velocity into $C$




(a) Training dataset




(b) Validation dataset

**FIGURE 16.** Analysis of the oversampling and undersampling techniques. The network illustrated in Fig. 5 was trained to output relative kinematics at time $t - (N-1)/2$ for two different datasets: the normal and redistributed KITTI dataset. Through the testing process, the RMSE of the relative velocity was illustrated in red and green, respectively.

categories. Then, we defined a relevance function as follows:

$$W(\Delta v_{k,c}) = \min(\mu_1, \log(\mu_2 \times \frac{N_s}{N_{s_{k,c}}})) \tag{9}$$

where each axis $k$ of a relative velocity $\Delta v$ has an attributed weight $W$ according to their category distribution $c$ and two adjustable parameters $\mu_1, \mu_2$. $N_s$ refers to the number of samples in the dataset. $N_{s_{k,c}}$ refers to the number of samples in the category distribution $c$ of the $k$-axis relative velocity. The log

function was used to smooth the weight and prevent the RF from hurting the training process of abundant samples. For the relative velocity of a car, we ignored the z-axis and set its weight to $\mu_1$. By default, $\mu_1$ and $\mu_2$ was set to 0.75 and 0.85.

Similarly to relative velocity, we wanted to balance the quaternion distribution. However, the loss function used and the model space of a quaternion prevent defining a weight for each quaternion component. Therefore, we analyzed the quaternion distribution with a distance function that is represented by the orientation metric without the logarithm term defined in (5b). The details of the process are as follows. A relative quaternion in KITTI dataset was selected and used as a reference quaternion. Then, the distance function is computed with other relative quaternions of the dataset. The distance values are separated into $2 * C$ categories, and weights are attributed according to the distribution of the distance. Finally, the previously defined RF with the same parameters is used to balance the distribution of the quaternion.

For both distribution balancing, we made the choice to apply the RF on the training dataset only. However, the validation dataset is used as a criterion to cease the learning process and prevent overfitting. Therefore, an unbalanced validation dataset may hurt the learning process. For this reason, our choice is questionable and may be interesting to analyze in future work.

The details of the analysis are as follows. For the same reason introduced in Section VII-H1, we trained the NRK defined in Fig. 5 to output relative kinematics at time $t - (N - 1)/2$. For this analysis, we considered four cases. The first case is the normal training where the RF was not used. The second, we refer to as Velocity RF, is the case where the RF was applied only for the relative velocity. The third and fourth refer to the cases where the RF was applied only for the quaternion and the case where the RF was applied for both relative kinematics. We refer to these cases as Orientation and Both RF. For each case, the RMSE of $x$ and $y$-axis of the relative velocity and the relative quaternion error were analyzed. The results are reported in Fig. 17. It is worth noting that each learning process was achieved three times, and the average error was considered to avoid the randomness effect from the learning process.

For each case, a better accuracy was obtained for relative velocity and relative quaternion of the training dataset that was under-represented. The use of the RF for the orientation only resulted in a minor deterioration of the relative velocity prediction. Conversely, when the RF was used for the velocity only, a minor deterioration was observed for the relative quaternion prediction. On the other hand, when both RF were used, the overall performances were improved for the training dataset. Furthermore, the prediction of abundant samples did not deteriorate. For the validation dataset, the prediction of the relative velocity was slightly deteriorated in the majority, as observed with the oversampling technique. Therefore, overfitting may have occurred once again. Nonetheless, minor improvements have been observed for a few range of relative kinematics.

Conversely to oversampling and undersampling, the distribution of the dataset was not changed. Furthermore, this approach suits better to simultaneous regression of the relative kinematics where the balance of each output can be configurable. Nonetheless, the optimization of the hyper-parameters becomes more difficult to achieve. The purpose of this section was not to suggest an optimal method but rather a starting point to introduce the importance of data balancing while dealing with DNN-based IO. Different techniques, hyper-parameters, approaches should be evaluated to improve the performances of the DNN-based IO. Nevertheless, there is a high chance to face overfitting for a specific range of values when the amount of samples is insufficient. Therefore, the best solution is to ensure a balance in the data distribution for relative kinematics while acquiring data.

## VIII. DISCUSSION

### A. HYPER-PARAMETERS RELATED TO DNN

This section briefly summarizes hyper-parameters to be considered when dealing with neural networks. For a given architecture and its associated loss function that evaluates the model performances, hyper-parameters have to be optimized to match the best accuracy possible.

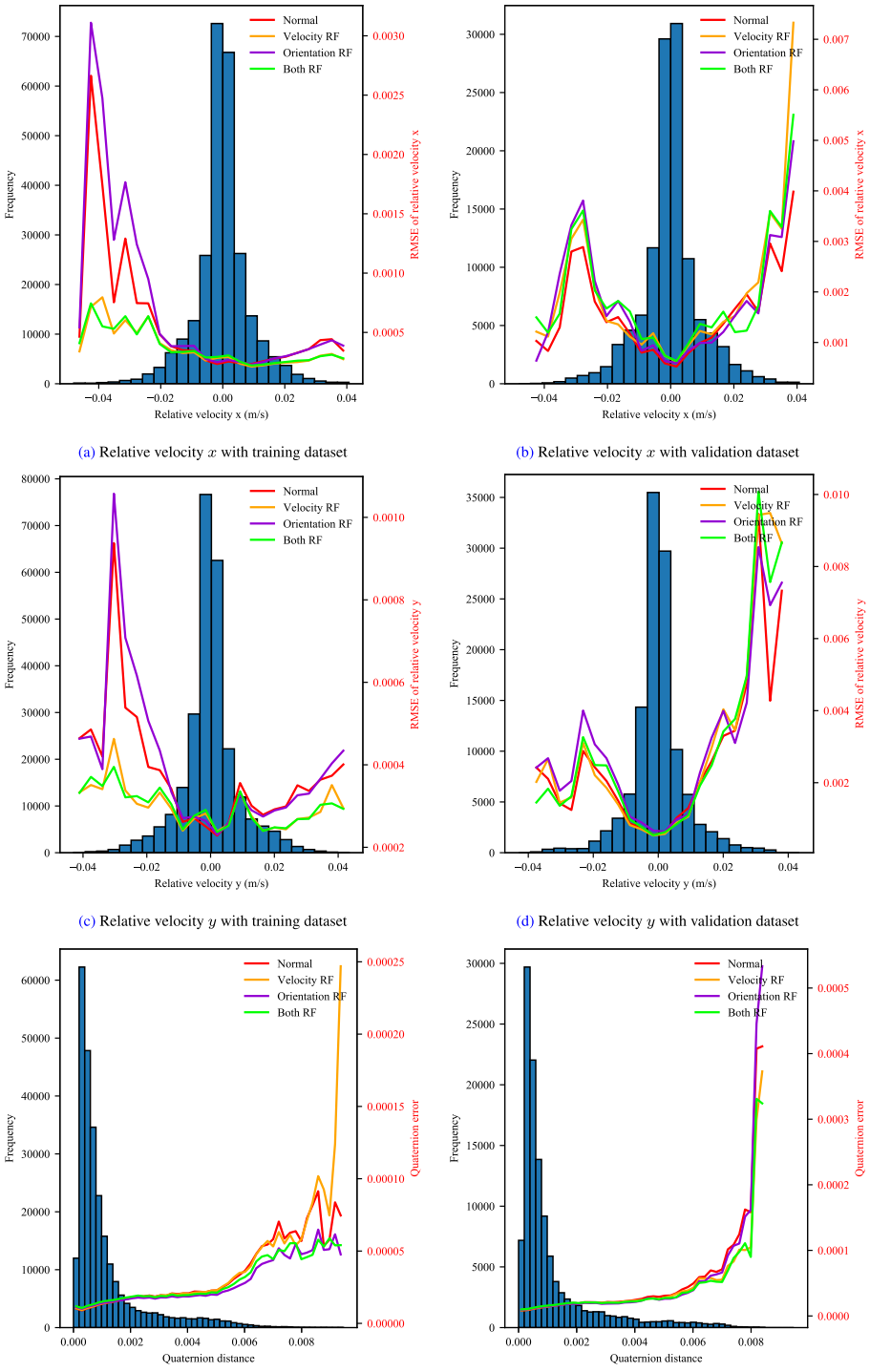
Among these parameters, the most important one is the learning rate. It characterizes the step size of the weight update and is usually defined in the range between [0,1]. A learning rate too large will result in a large weight update and produce poor performances, while a small value will make the training process longer and may not converge towards the optimal solution.

Learning rate goes hand in hand with the optimization algorithm Adam, which is predominant among all other optimizers. Nevertheless, it is still possible for other algorithms to perform better and maybe interesting to investigate in future work.

Once other parameters of the neural network have been decided, the adaptive learning rate can be coupled with an optimizer. As the training progresses, from a computed score, we can decide to reduce the learning rate to converge towards the most optimal solution that could have been missed with a larger learning rate.

The batch size used during the training process will determine the frequency of updates. The smaller the batch size is, the less accurate the gradient estimation will be. On the other hand, a larger batch size will require a larger GPU memory but will provide a more accurate result. Notwithstanding, a large batch size can result in accuracy loss.

After fixing these hyper-parameters, hidden units can be fine-tuned, and additional layers can be added to make the neural network deeper. However, deeper neural networks do not necessarily result in better performances. Nevertheless, they play an important role in prediction accuracy. Simultaneously, the activation function can impact the convergence ability of the network and should be carefully selected. Several techniques to initialize the network weight exist for the optimization of the training procedure Weight can be




(a) Relative velocity $x$ with training dataset

(b) Relative velocity $x$ with validation dataset

(c) Relative velocity $y$ with training dataset

(d) Relative velocity $y$ with validation dataset

(e) Relative quaternion prediction with training dataset

(f) Relative quaternion prediction with validation dataset

**FIGURE 17. Analysis of the relevance function. The network defined in Fig. 5 was trained to output relative kinematics at time $t-(N-1)/2$ using the relevance function defined in (9). The RMSE of the relative velocity and the quaternion error of four cases were analysed. We refer to these four cases as Normal, Velocity RF, Orientation RF and Both RF. Their errors are represented in red, orange, purple and green, respectively. The histogram is shown in the background to support the visualization of metrics errors indicated on the y-axis.**

initialized to the same constant values, but in practice, randoms values are given to ensure unsymmetrical hidden units. Other techniques such as Xavier, He normal, orthogonal initialization exist and can be used to prevent exploding or vanishing gradient [63]. One of the most severe issues with recurrent neural networks is the vanishing and exploding gradient. While there are many methods to combat this, such as gradient clipping for exploding gradients and more

complicated architectures, including the LSTM and GRU for vanishing gradients, orthogonal initialization is an interesting yet straightforward approach.

After all these optimization processes, the neural network may overfit. Regularization techniques such as L2, L1, or norm constraints for weight exist to solve this issue and can be complemented with dropout layers. This last one will ignore with a probability p a neuron during the training phase encouraging parameters of the model to act differently. Several techniques exist to create an adaptive dropout [64], [65]. Nonetheless, there is no generalization for dropout. Based on the neural network architecture, an investigation should be done. It is worth noting that dropout is ineffective with convolution layers where there are only a few parameters that need less regularization. In most recent works, they have been replaced with batch normalization layers whereas dropout layers are used with fully connected layers [62]. However, even with dropout, overfitting remains possible. Then, early stopping can be used as a complement, and stop the training process when the best accuracy was obtained with the validation dataset to prevent overfitting. Nonetheless, this method was questioned but remained appreciated by the community.

Noise and bias present within input data might complicate their underlying relationship and hurt the learning process. Hence, pre-processing methods are used to ensure an optimal feature extraction and eliminate noise from raw data but may discard worthwhile information. Based on the knowledge of the data we are dealing with, several filtering processes exist. The most widely spread and most straightforward technique, named average filtering, filters the data by calculating the average value of given window sizes. An alternative consists of adding a customized weight for every component within the windows to influence their impact on the filtering process. Many other techniques exist, such as Savitzky-Golay and Hamming window filtering, could be interesting to investigate in the future for IO problem [66].

### B. OUTLIER

In this section, we highlight the outlier phenomenon. An outlier is a data that is significantly distant from other dataset points. Depending on what we are trying to achieve, the information carried by outliers may be caused during data acquisition and may not be relevant for the prediction. Indeed, modeling accuracy decreases as the percentage of outlier inside a training dataset increases. For example, ReLU activation function is mainly impacted by these values when the architecture of the neural network is not too deep. The deeper the NN is, the more the ReLU activation function will regularize and converge faster.

Ideally, outliers should be excluded from the dataset. However, detecting that anomalous instances is not always possible. The current state-of-the-art DNN will use large training datasets to drown out their impact on the modeling accuracy. Other methods, such as robust error, loss functions, or regularization, have also been introduced to avoid noise and outliers influence. For some situations, we might want our model to be aware of such deviant behavior. This is the case for variation in delta time, which reflects time sampling issues and has an essential role in the pose estimation. As mentioned in Section III-D, many sequences from KITII dataset contained consecutive duplicated data and were not considered as relevant for IO. Therefore, we decided to either correct these outliers with linear approximation or remove them from our training and validation dataset. Even though this pre-processing method may not be perfect, a majority of outliers were removed, and the few remaining ones are automatically regularized by the neural network itself during the learning process. As aforementioned, the outlier does not necessarily mean that the data is incorrect. As mentioned in Section VII-D, the Euler angle of the KITTI dataset could vary from $-\pi$ to $\pi$ for two consecutive frames and vice versa, resulting in high relative quaternions that could be considered as outliers.

### C. PERSPECTIVE OF NETWORK DESIGN

In this section, we discuss a future perspective for our neural network architecture. Inside the kinematic block in Fig. 1, the features coming from LSTM are fed into fully connected layers to output relative kinematics. In this paper, we used either many-to-one LSTM or many-to-many to output one or $N$ sets of relative kinematics where N is the size of the input window. In [16], [56], the $N$ features coming from convolution were used to predict $N$ sets of outputs. In [57], the dimension of the feature vector was first reduced and then concatenated to output one set of relative kinematic. In Section VII-F, the influence of the distribution of future and past features over the performances was introduced. When one is ubiquitous, the accuracy of the prediction significantly decreases. Therefore, we may adapt the transition between the last LSTM and the linear layers to select specific features and improve the performances to output shorter-dimension relative kinematics.

We imagined future perspectives for our neural network architecture, which we explain as follows. First, we considered a manual selection of the features among the $N$ feature vectors outputted by the LSTM. However, a loss of information could occur during the learning process. Therefore, each feature vector from each cell state of the LSTM should be considered. Then, the network should determine which features suit the best for the prediction automatically. A solution could be to concatenate the $N$ feature vectors into one unique vector. However, the length of the concatenated vector may be important. A higher-dimensional vector will significantly increase the parameters between two layers, the computation time, and the risk of overfitting. Therefore, the dimension of this feature vector should be reduced. Pooling layers could be used, but their use is less common with LSTM layers. As an alternative solution, additional fully connected layers could be used. Relevant features from each LSTM cell state would be mixed into one vector. Then, the $N$ resulting vectors could be concatenated and fed into other fully connected layers before regressing the relative kinematics. This method would

decrease the number of parameters between two consecutive layers. However, a more in-depth parameter configuration is required.

In Section VII-A, additional inputs such as previous quaternions and previous velocities were introduced. Both were provided by a third device we consider as ground truth. Therefore, their use is limited in the case of inertial odometry (IO) with a low-cost IMU only. Such concern was first introduced in [56]. During the learning process, the ground truth values of the quaternion were used as an input whereas predicted values were used during the testing process. An additional computation method based on the design of the neural network was supervising the prediction error. In this paper, we did not investigate such aspect of the testing process. Therefore, it would be interesting to consider this trail in future work.

In this section, we introduced some perspectives to be investigated. Such architecture questions correlated to IO-problem should be discussed and investigated for future work.

## IX. CONCLUSION

This paper presented a deep analysis of DNN-based IO from various aspects as a general guideline that could help the community in their future work. The proposed analysis is based on a bidirectional LSTM and multi-task learning loss function that suits the kinematics in an end-to-end manner. The DNN hyper-parameters correlated to IO were introduced and globally analyzed in distinct sections. First, new inputs correlated to the IO were introduced. From this analysis, the input multiplication and the use of the rotation matrix to represent orientation were highlighted as promising trails to consider in future works. Then, a reminder about the difficulties of generalizing hyper-parameters has been established. Minor changes for the window size of the input and loss functions led to an improvement of the overall performances. In this paper, the question of relative quaternion representation has been discussed. The representation of the Hamilton product with its associated loss function suits the best to the IO problem. The question of the feature distribution has also been introduced. We demonstrated that a balanced distribution between past and future features tends to optimal performances. Finally, additional notions such as data frequency and data balance were introduced in the DNN-based IO field. Although a conclusion could not be drawn for our analyses, the results remain interesting for the future of the IO research. For each hyper-parameter, new perspectives have been opened for the IO. In future work, we would like to focus on a specific hyper-parameter and offer a deeper analysis to achieve greater performances.

## REFERENCES

[1] R. Chatila and J. Laumond, "Position referencing and consistent world modeling for mobile robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, Mar. 1985, pp. 138–145.
[2] W. Kang and Y. Han, "SmartPDR: Smartphone-based pedestrian dead reckoning for indoor localization," *IEEE Sensors J.*, vol. 15, no. 5, pp. 2906–2916, May 2015.
[3] Y. Fuke and E. Krotkov, "Dead reckoning for a lunar rover on uneven terrain," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 1996, pp. 411–416.
[4] M. Kourogi and T. Kurata, "Personal positioning based on walking locomotion analysis with self-contained sensors and a wearable camera," in *Proc. 2nd IEEE ACM Int. Symp. Mixed Augmented Reality*, 2003, p. 103.
[5] V. Renaudin, "Evaluating indoor positioning systems in a shopping mall: The lessons learned from the IPIN 2018 competition," *IEEE Access*, vol. 7, pp. 148594–148628, 2019.
[6] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., CVPR*, vol. 1, Jun. 2004, pp. 1–8.
[7] J. Rehder, K. Gupta, S. Nuske, and S. Singh, "Global pose estimation with limited GPS and long range visual odometry," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2012, pp. 627–633.
[8] R. H. Venkatnarayan and M. Shahzad, "Enhancing indoor inertial odometry with WiFi," *Proc. ACM Interact., Mobile, Wearable Ubiquitous Technol.*, vol. 3, no. 2, pp. 1–27, Jun. 2019.
[9] Z. Xin, Y. Cai, T. Lu, X. Xing, S. Cai, J. Zhang, Y. Yang, and Y. Wang, "Localizing discriminative visual landmarks for place recognition," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 5979–5985.
[10] N. Ahmad, R. A. R. Ghazilla, N. M. Khairi, and V. Kasi, "Reviews on various inertial measurement unit (IMU) sensor applications," *Int. J. Signal Process. Syst.*, vol. 1, no. 2, pp. 256–262, 2013
[11] N. Yazdi, F. Ayazi, and K. Najafi, "Micromachined inertial sensors," *Proc. IEEE*, vol. 86, no. 8, pp. 1640–1659, Aug. 1998.
[12] N. Bonnor, "Principles of GNSS, inertial, and multisensor integrated navigation systems – second EditionPaul D. Groves artech house, 2013, 776 pp ISBN-13: 978-1-60807-005-3," *J. Navigat.*, vol. 67, no. 1, pp. 191–192, Jan. 2014.
[13] A. H. Mohamed and K. P. Schwarz, "Adaptive Kalman filtering for INS/GPS," *J. Geodesy*, vol. 73, no. 4, pp. 193–203, May 1999.
[14] A. R. Jimenez, F. Seco, J. C. Prieto, and J. Guevara, "Indoor pedestrian navigation using an INS/EKF framework for yaw drift reduction and a foot-mounted IMU," in *Proc. 7th Workshop Positioning, Navigat. Commun.*, Mar. 2010, pp. 135–143.
[15] Z. Wang, H. Zhao, S. Qiu, and Q. Gao, "Stance-phase detection for ZUPT-aided foot-mounted pedestrian navigation system," *IEEE/ASME Trans. Mechatronics*, vol. 20, no. 6, pp. 3170–3181, Dec. 2015.
[16] M. Brossard, A. Barrau, and S. Bonnabel, "AI-IMU dead-reckoning," *IEEE Trans. Intell. Vehicles*, vol. 5, no. 4, pp. 585–595, Dec. 2020.
[17] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015.
[18] M. Edel and E. Koppe, "An advanced method for pedestrian dead reckoning using BLSTM-RNNs," in *Proc. Int. Conf. Indoor Positioning Indoor Navigat. (IPIN)*, Oct. 2015, pp. 1–6.
[19] F. Gu, K. Khoshelham, C. Yu, and J. Shang, "Accurate step length estimation for pedestrian dead reckoning localization using stacked autoencoders," *IEEE Trans. Instrum. Meas.*, vol. 68, no. 8, pp. 2705–2713, Oct. 2018.
[20] B. Wang, X. Liu, B. Yu, R. Jia, and X. Gan, "Pedestrian dead reckoning based on motion mode recognition using a smartphone," *Sensors*, vol. 18, no. 6, p. 1811, Jun. 2018.
[21] Q. Wang, H. Luo, L. Ye, A. Men, F. Zhao, Y. Huang, and C. Ou, "Pedestrian heading estimation based on spatial transformer networks and hierarchical LSTM," *IEEE Access*, vol. 7, pp. 162309–162322, 2019.
[22] B. Wagstaff and J. Kelly, "LSTM-based zero-velocity detection for robust inertial navigation," in *Proc. Int. Conf. Indoor Positioning Indoor Navigat. (IPIN)*, Sep. 2018, pp. 1–8.
[23] N. Kawaguchi, J. Nozaki, T. Yoshida, K. Hiroi, T. Yonezawa, and K. Kaji, "End-to-end walking speed estimation method for smartphone pdr using DualCNN-LSTM," in *Proc. IPIN (Short Papers/Work-in-Progress Papers)*, 2019, pp. 463–470.
[24] J. Huang, Z. Huang, and K. Chen, "Combining low-cost inertial measurement unit (IMU) and deep learning algorithm for predicting vehicle attitude," in *Proc. IEEE Conf. Dependable Secure Comput.*, Aug. 2017, pp. 237–239.
[25] M. Abolfazli Esfahani, H. Wang, K. Wu, and S. Yuan, "AbolDeepIO: A novel deep inertial odometry network for autonomous vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 5, pp. 1941–1950, May 2020.
[26] A. Canziani, A. Paszke, and E. Culurciello, "An analysis of deep neural network models for practical applications," 2016, *arXiv:1605.07678*. [Online]. Available: http://arxiv.org/abs/1605.07678
[27] W. Samek, A. Binder, G. Montavon, S. Lapuschkin, and K.-R. Muller, "Evaluating the visualization of what a deep neural network has learned," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 11, pp. 2660–2673, Nov. 2017.

[28] T. Minematsu, A. Shimada, H. Uchiyama, and R.-I. Taniguchi, "Analytics of deep neural network-based background subtraction," *J. Imag.*, vol. 4, no. 6, p. 78, Jun. 2018.
[29] T. Sattler, Q. Zhou, M. Pollefeys, and L. Leal-Taixe, "Understanding the limitations of CNN-based absolute camera pose regression," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 3302–3312.
[30] N. El-Sheimy and A. Youssef, "Inertial sensors technologies for navigation applications: State of the art and future trends," *Satell. Navigat.*, vol. 1, no. 1, p. 2, Dec. 2020.
[31] H. C. Lefevre, *The Fiber-Optic Gyroscope*. Norwood, MA, USA: Artech House, 2014.
[32] F. Aronowitz, "The laser gyro," in *Laser Applications*, M. Ross, Ed. New York, NY, USA: Academic, 1971, p. 131.
[33] M. Kok, J. D. Hol, and T. B. Schön, "Using inertial sensors for position and orientation estimation," *Found. Trends Signal Process.*, vol. 11, nos. 1–2, pp. 1–153, 2017, doi: 10.1561/2000000094.
[34] A. Youssef and N. El-Sheimy, "Gyroscope using torus shaped channels 1955 and image processing," U.S. Patent 62 796 231 24, Jan. 2019.
[35] F. Delhaye, "HRG by SAFRAN: The game-changing technology," in *Proc. IEEE Int. Symp. Inertial Sensors Syst. (INERTIAL)*, Mar. 2018, pp. 1–4.
[36] D. Titterton, J. L. Weston, and J. Weston, *Strapdown Inertial Navigation Technology*, vol. 17. London, U.K.: IET, 2004.
[37] N. El-Sheimy, H. Hou, and X. Niu, "Analysis and modeling of inertial sensors using allan variance," *IEEE Trans. Instrum. Meas.*, vol. 57, no. 1, pp. 140–149, Jan. 2008.
[38] R. Munguía, "A GPS-aided inertial navigation system in direct configuration," *J. Appl. Res. Technol.*, vol. 12, no. 4, pp. 803–814, Aug. 2014.
[39] J. A. Farrell, T. D. Givargis, and M. J. Barth, "Real-time differential carrier phase GPS-aided INS," *IEEE Trans. Control Syst. Technol.*, vol. 8, no. 4, pp. 709–721, Jul. 2000.
[40] H. Hellmers, A. Norrdine, J. Blankenbach, and A. Eichhorn, "An IMU/magnetometer-based indoor positioning system using Kalman filtering," in *Proc. Int. Conf. Indoor Positioning Indoor Navigat.*, Oct. 2013, pp. 1–9.
[41] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart, "Robust visual inertial odometry using a direct EKF-based approach," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2015, pp. 298–304.
[42] 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.
[43] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint Kalman filter for vision-aided inertial navigation," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 2007, pp. 3565–3572.
[44] A. Solin, S. Cortes, E. Rahtu, and J. Kannala, "Inertial odometry on handheld smartphones," in *Proc. 21st Int. Conf. Inf. Fusion (FUSION)*, Jul. 2018, pp. 1–5.
[45] A. Ramanandan, A. Chen, and J. A. Farrell, "Inertial navigation aiding by stationary updates," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 1, pp. 235–248, Mar. 2012.
[46] G. Dissanayake, S. Sukkarieh, E. Nebot, and H. Durrant-Whyte, "The aiding of a low-cost strapdown inertial measurement unit using vehicle model constraints for land vehicle applications," *IEEE Trans. Robot. Autom.*, vol. 17, no. 5, pp. 731–747, Oct. 2001.
[47] H. Yan, Q. Shan, and Y. Furukawa, "RIDI: Robust IMU double integration," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 621–636.
[48] A. Barrau and S. Bonnabel, "Invariant Kalman filtering," *Annu. Rev. Control, Robot., Auton. Syst.*, vol. 1, pp. 237–257, May 2018.
[49] M. Brossard, A. Barrau, and S. Bonnabel, "RINS-W: Robust inertial navigation system on wheels," 2019, *arXiv:1903.02210*. [Online]. Available: http://arxiv.org/abs/1903.02210
[50] S. Cortes, A. Solin, and J. Kannala, "Deep learning based speed estimation for constraining strapdown inertial navigation on smartphones," in *Proc. IEEE 28th Int. Workshop Mach. Learn. Signal Process. (MLSP)*, Sep. 2018, pp. 1–6.
[51] W. Liu, D. Caruso, E. Ilg, J. Dong, A. I. Mourikis, K. Daniilidis, V. Kumar, and J. Engel, "TLIO: Tight learned inertial odometry," *IEEE Robot. Autom. Lett.*, vol. 5, no. 4, pp. 5653–5660, Oct. 2020.
[52] 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.*, Apr. 2018, pp. 1–9.
[53] C. Chen, X. Lu, J. Wahlstrom, A. Markham, and N. Trigoni, "Deep neural network based inertial odometry using low-cost inertial measurement units," *IEEE Trans. Mobile Comput.*, early access, Dec. 19, 2020, doi: 10.1109/TMC.2019.2960780.
[54] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[55] 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. Autom. (ICRA)*, May 2020, pp. 3146–3152.
[56] M. A. Esfahani, H. Wang, K. Wu, and S. Yuan, "OriNet: Robust 3-D orientation estimation with a single particular IMU," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 399–406, Apr. 2020.
[57] 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, Aug. 2019.
[58] O. J. Woodman, "An introduction to inertial navigation," Comput. Lab., Univ. Cambridge, Cambridge, U.K., Tech. Rep. UCAM-CL-TR-696, Aug. 2007. [Online]. Available: https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-696.pdf
[59] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 1725–1732.
[60] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1231–1237, Sep. 2013.
[61] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The EuRoC micro aerial vehicle datasets," *Int. J. Robot. Res.*, vol. 35, no. 10, pp. 1157–1163, Sep. 2016.
[62] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 448–456.
[63] D. Mishkin and J. Matas, "All you need is a good init," 2015, *arXiv:1511.06422*. [Online]. Available: http://arxiv.org/abs/1511.06422
[64] S. Wager, S. Wang, and P. Liang, "Dropout training as adaptive regularization," 2013, *arXiv:1307.1493*. [Online]. Available: http://arxiv.org/abs/1307.1493
[65] L. Ba, "Adaptive dropout for training deep neural networks," Ph.D. dissertation, Dept. Elect. Comput. Eng., Univ. Toronto, Toronto, ON, Canada, 2013.
[66] W. Wettayaprasit, N. Laosen, and S. Chevakidagarn, "Data filtering technique for neural networks forecasting," in *Proc. 7th WSEAS Int. Conf. Simulation, Modelling Optim.*,Sep. 2007, pp. 225–230.

**QUENTIN ARNAUD DUGNE–HENNEQUIN** received the M.S. degree in computer science from ESIR University, France, in 2019. He is currently working with the Central Library, Kyushu University. His current research interests include indoor navigation, computer vision, and deep learning.

**HIDEAKI UCHIYAMA** (Member, IEEE) received the B.E., M.E., and D.E. degrees from Keio University, in 2006, 2007, and 2010, respectively. From 2012 to 2014, he worked with Toshiba Corporation. From 2014 to 2018, he was an Assistant Professor with the Graduate School of Information Science and Electrical Engineering, Kyushu University, where he has been an Associate Professor with the Central Library, since 2018. His research interests include indoor navigation, computer vision, and augmented reality.

**JOÃO PAULO SILVA DO MONTE LIMA** (Member, IEEE) received the Ph.D. degree in computer science from the Universidade Federal de Pernambuco, Recife, Brazil, in 2014. He joined the Universidade Federal Rural de Pernambuco, Recife, in 2013, where he is currently an Associate Professor. He is also a Senior Scientist with the Voxar Labs research group, Universidade Federal de Pernambuco. His research interests include 3D tracking, augmented reality, computer vision, and computer graphics.