HAL
open science

AI-IMU Dead-Reckoning

Martin Brossard, Axel Barrau, Silvère Bonnabel

▶ **To cite this version:**

Martin Brossard, Axel Barrau, Silvère Bonnabel. AI-IMU Dead-Reckoning. IEEE Transactions on Intelligent Vehicles, 2020, 10.1109/TIV.2020.2980758 . hal-02097099v2

HAL Id: hal-02097099

https://hal.science/hal-02097099v2

Submitted on 20 Apr 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

AI-IMU Dead-Reckoning

Martin BROSSARD* , Axel BARRAU† and Silvère BONNABEL*
*MINES ParisTech, PSL Research University, Centre for Robotics, 60 Bd Saint-Michel, 75006 Paris, France
†Safran Tech, Groupe Safran, Rue des Jeunes Bois-Châteaufort, 78772, Magny Les Hameaux Cedex, France

***Abstract*—In this paper we propose a novel accurate method for dead-reckoning of wheeled vehicles based only on an Inertial Measurement Unit (IMU). In the context of intelligent vehicles, robust and accurate dead-reckoning based on the IMU may prove useful to correlate feeds from imaging sensors, to safely navigate through obstructions, or for safe emergency stops in the extreme case of exteroceptive sensors failure. The key components of the method are the Kalman filter and the use of deep neural networks to dynamically adapt the noise parameters of the filter. The method is tested on the KITTI odometry dataset, and our dead-reckoning inertial method based *only* on the IMU accurately estimates 3D position, velocity, orientation of the vehicle and self-calibrates the IMU biases. We achieve on average a 1.10% translational error and the algorithm competes with top-ranked methods which, by contrast, use LiDAR or stereo vision. We make our implementation open-source at:**

`https://github.com/mbrossar/ai-imu-dr`

***Index Terms*—localization, deep learning, invariant extended Kalman filter, KITTI dataset, inertial navigation, inertial measurement unit**

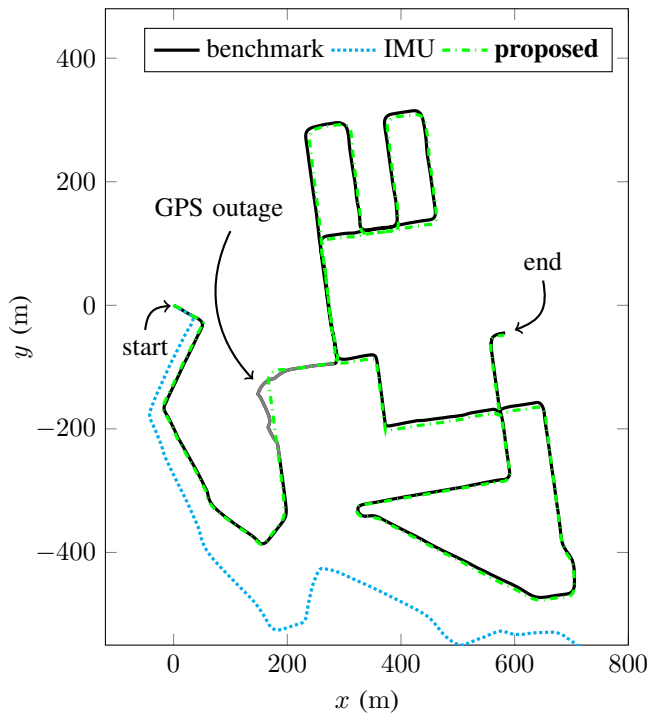

Fig. 1. Trajectory results on seq. 08 (drive #28, 2011/09/30) [3] of the KITTI dataset. The proposed method (green) accurately follows the benchmark trajectory for the entire sequence (4.2 km, 9 min), whereas the pure IMU integration (cyan) quickly diverges. Both methods use only IMU signals and are initialized with the benchmark pose and velocity. We see during the GPS outage that occurs in this sequence, our solution keeps estimating accurately the trajectory.

I. INTRODUCTION

INTELLIGENT vehicles need to know where they are located in the environment, and how they are moving through it. An accurate estimate of vehicle dynamics allows validating information from imaging sensors such as lasers, ultrasonic systems, and video cameras, correlating the feeds, and also ensuring safe motion throughout whatever may be seen along the road [1]. Moreover, in the extreme case where an emergency stop must be performed owing to severe occlusions, lack of texture, or more generally imaging system failure, the vehicle must be able to assess accurately its dynamical motion. For all those reasons, the Inertial Measurement Unit (IMU) appears as a key component of intelligent vehicles [2]. Note that Global Navigation Satellite System (GNSS) allows for global position estimation but it suffers from phase tracking loss in densely built-up areas or through tunnels, is sensitive to jamming, and may not be used to provide continuous accurate and robust localization information, as exemplified by a GPS outage in the well known KITTI dataset [3], see Figure 1.

Kalman filters are routinely used to integrate the outputs of IMUs. When the IMU is mounted on a car, it is common practice to make the Kalman filter incorporate side information about the specificity of wheeled vehicle dynamics, such as approximately null lateral and upward velocity assumption in the form of pseudo-measurements. However, the degree of confidence the filter should have in this side information is encoded in a covariance noise parameter which is difficult to set manually, and moreover which should dynamically adapt to the motion, e.g., lateral slip is larger in bends than in straight lines. Using the recent tools from the field of Artificial Intelligence (AI), namely deep neural networks, we propose a method to automatically learn those parameters and their dynamic adaptation for IMU only dead-reckoning. Our contributions, and the paper's organization, are as follows:

- we introduce a state-space model for wheeled vehicles as well as simple assumptions about the motion of the car in Section II;
- we implement a state-of-the-art Kalman filter [4,5] that exploits the kinematic assumptions and combines them with the IMU outputs in a statistical way in Section III-C. It yields accurate estimates of position, orientation and velocity of the car, as well as IMU biases, along with associated uncertainty (covariance);
- we exploit deep learning for dynamic adaptation of co-

variance noise parameters of the Kalman filter in Section IV-A. This module greatly improves filter's robustness and accuracy, see Section V-D;
- we demonstrate the performances of the approach on the KITTI dataset [3] in Section V. Our approach solely based on the IMU produces accurate estimates and competes with top-ranked LiDAR and stereo camera methods [6,7]; and we do not know of IMU based dead-reckoning methods capable to compete with such results;
- the approach is not restricted to inertial only dead-reckoning of wheeled vehicles. Thanks to the versatility of the Kalman filter, it can easily be applied for railway vehicles [8], coupled with GNSS, the backbone for IMU self-calibration, or for using IMU as a speedometer in path-reconstruction and map-matching methods [9]–[12].

A. Relation to Previous Literature

Autonomous vehicle must robustly self-localize with their embarked sensor suite which generally consists of odometers, IMUs, radars or LiDARs, and cameras [1,2,12]. Simultaneous Localization And Mapping based on inertial sensors, cameras, and/or LiDARs have enabled robust real-time localization systems, see e.g., [6,7]. Although these highly accurate solutions based on those sensors have recently emerged, they may drift when the imaging system encounters troubles.

As concerns wheeled vehicles, taking into account vehicle constraints and odometer measurements are known to increase the robustness of localization systems [13]–[16]. Although quite successful, such systems continuously process a large amount of data which is computationally demanding and energy consuming. Moreover, an autonomous vehicle should run in parallel its own robust IMU-based localization algorithm to perform maneuvers such as emergency stops in case of other sensors failures, or as an aid for correlation and interpretation of image feeds [2].

High precision aerial or military inertial navigation systems achieve very small localization errors but are too costly for consumer vehicles. By contrast, low and medium-cost IMUs suffer from errors such as scale factor, axis misalignment and random walk noise, resulting in rapid localization drift [17]. This makes the IMU-based positioning unsuitable, even during short periods.

Inertial navigation systems have long leveraged virtual and pseudo-measurements from IMU signals, e.g. the widespread Zero velocity UPdaTe (ZUPT) [18]–[20], as covariance adaptation [21]. In parallel, deep learning and more generally machine learning are gaining much interest for inertial navigation [22,23]. In [22] velocity is estimated using support vector regression whereas [23] use recurrent neural networks for end-to-end inertial navigation. Those methods are promising but restricted to pedestrian dead-reckoning since they generally consider slow horizontal planar motion, and must infer velocity directly from a small sequence of IMU measurements, whereas we can afford to use larger sequences. A more general end-to-end learning approach is [24], which trains deep networks end-to-end in a Kalman filter. Albeit promising, the method obtains large translational error $> 30\%$ in their stereo odometry experiment. Finally, [25] uses deep learning for estimating covariance of a local odometry algorithm that is fed into a global optimization procedure, and in [26] we used Gaussian processes to learn a wheel encoders error. Our conference paper [20] contains preliminary ideas, albeit not concerned at all with covariance adaptation: a neural network essentially tries to detect when to perform ZUPT.

Dynamic adaptation of noise parameters in the Kalman filter is standard in the tracking literature [27], however adaptation rules are application dependent and are generally the result of manual "tweaking" by engineers. Finally, in [28] the authors propose to use classical machine learning techniques to to learn static noise parameters (without adaptation) of the Kalman filter, and apply it to the problem of IMU-GNSS fusion.

II. IMU and Problem Modelling

An inertial navigation system uses accelerometers and gyros provided by the IMU to track the orientation $\mathbf{R}_n$, velocity $\mathbf{v}_n \in \mathbb{R}^3$ and position $\mathbf{p}_n \in \mathbb{R}^3$ of a moving platform relative to a starting configuration $(\mathbf{R}_0, \mathbf{v}_0, \mathbf{p}_0)$. The orientation is encoded in a rotation matrix $\mathbf{R}_n \in SO(3)$ whose columns are the axes of a frame attached to the vehicle.

A. IMU Modelling

The IMU provides noisy and biased measurements of the instantaneous angular velocity vector $\boldsymbol{\omega}_n$ and specific acceleration $\mathbf{a}_n$ as follows [17]

$$\boldsymbol{\omega}_n^{\text{IMU}} = \boldsymbol{\omega}_n + \mathbf{b}_n^{\boldsymbol{\omega}} + \mathbf{w}_n^{\boldsymbol{\omega}}, \tag{1}$$

$$\mathbf{a}_n^{\text{IMU}} = \mathbf{a}_n + \mathbf{b}_n^{\mathbf{a}} + \mathbf{w}_n^{\mathbf{a}}, \tag{2}$$

where $\mathbf{b}_n^{\boldsymbol{\omega}}$, $\mathbf{b}_n^{\mathbf{a}}$ are quasi-constant biases and $\mathbf{w}_n^{\boldsymbol{\omega}}$, $\mathbf{w}_n^{\mathbf{a}}$ are zero-mean Gaussian noises. The biases follow a random walk

$$\mathbf{b}_{n+1}^{\boldsymbol{\omega}} = \mathbf{b}_n^{\boldsymbol{\omega}} + \mathbf{w}_n^{\mathbf{b}_\omega}, \tag{3}$$

$$\mathbf{b}_{n+1}^{\mathbf{a}} = \mathbf{b}_n^{\mathbf{a}} + \mathbf{w}_n^{\mathbf{b}_\mathbf{a}}, \tag{4}$$

where $\mathbf{w}_n^{\mathbf{b}_\omega}$, $\mathbf{w}_n^{\mathbf{b}_\mathbf{a}}$ are zero-mean Gaussian noises.

The kinematic model is governed by the following equations

$$\mathbf{R}_{n+1}^{\text{IMU}} = \mathbf{R}_n^{\text{IMU}} \exp\left((\boldsymbol{\omega}_n dt)_\times\right), \tag{5}$$

$$\mathbf{v}_{n+1}^{\text{IMU}} = \mathbf{v}_n^{\text{IMU}} + \left(\mathbf{R}_n^{\text{IMU}} \mathbf{a}_n + \mathbf{g}\right) dt, \tag{6}$$

$$\mathbf{p}_{n+1}^{\text{IMU}} = \mathbf{p}_n^{\text{IMU}} + \mathbf{v}_n^{\text{IMU}} dt, \tag{7}$$

between two discrete time instants sampling at dt, where we let the IMU velocity be $\mathbf{v}_n^{\text{IMU}} \in \mathbb{R}^3$ and its position $\mathbf{p}_n^{\text{IMU}} \in \mathbb{R}^3$ in the world frame. $\mathbf{R}_n^{\text{IMU}} \in SO(3)$ is the 3×3 rotation matrix that represents the IMU orientation, i.e. that maps the IMU frame to the world frame, see Figure 2. Finally $(\mathbf{y})_\times$ denotes the skew symmetric matrix associated with cross product with $\mathbf{y} \in \mathbb{R}^3$. The true angular velocity $\boldsymbol{\omega}_n \in \mathbb{R}^3$ and the true specific acceleration $\mathbf{a}_n \in \mathbb{R}^3$ are the inputs of the system (5)-(7). In our application scenarios, the effects of earth rotation and Coriolis acceleration are ignored, Earth is considered flat, and the gravity vector $\mathbf{g} \in \mathbb{R}^3$ is a known constant.

All sources of error displayed in (1) and (2) are harmful since a simple implementation of (5)-(7) leads to a triple integration of raw data, which is much more harmful that the

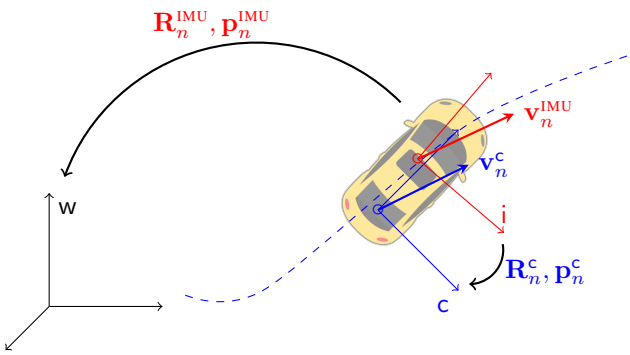

Fig. 2. The coordinate systems that are used in the paper. The IMU pose $(\mathbf{R}_n^{\text{IMU}}, \mathbf{p}_n^{\text{IMU}})$ maps vectors expressed in the IMU frame i (red) to the world frame w (black). The IMU frame is attached to the vehicle and misaligned with the car frame c (blue). The pose between the car and inertial frames $(\mathbf{R}_n^{\text{c}}, \mathbf{p}_n^{\text{c}})$ is unknown. IMU velocity $\mathbf{v}_n^{\text{IMU}}$ and car velocity $\mathbf{v}_n^{\text{c}}$ are respectively expressed in the world frame and in the car frame.

unique integration of differential wheel speeds [12]. Indeed, a bias of order ϵ has an impact of order $\epsilon t^2/2$ on the position after t seconds, leading to potentially huge drift.

B. Problem Modelling

We distinguish between three different frames, see Figure 2: i) the static world frame, w; ii) the IMU frame, i, where (1)-(2) are measured; and iii) the car frame, c. The car frame is an ideal frame attached to the car, that will be estimated online and plays a key role in our approach. Its orientation w.r.t. i is denoted $\mathbf{R}_n^{\text{c}} \in SO(3)$ and its origin denoted $\mathbf{p}_n^{\text{c}} \in \mathbb{R}^3$ is the car to IMU level arm. In the rest of the paper, we tackle the following problem:

IMU Dead-Reckoning Problem. *Given an initial known configuration* $(\mathbf{R}_0^{\text{IMU}}, \mathbf{v}_0^{\text{IMU}}, \mathbf{p}_0^{\text{IMU}})$*, perform in real-time IMU dead-reckoning, i.e. estimate the IMU and car variables*

$$\mathbf{x}_n := (\mathbf{R}_n^{\text{IMU}},\ \mathbf{v}_n^{\text{IMU}},\ \mathbf{p}_n^{\text{IMU}},\ \mathbf{b}_n^{\boldsymbol{\omega}},\ \mathbf{b}_n^{\mathbf{a}},\ \mathbf{R}_n^{\text{c}},\ \mathbf{p}_n^{\text{c}}) \tag{8}$$

using only the inertial measurements $\boldsymbol{\omega}_n^{\text{IMU}}$ *and* $\mathbf{a}_n^{\text{IMU}}$.

III. Kalman Filtering with Pseudo-Measurements

The Extended Kalman Filter (EKF) was first implemented in the Apollo program to localize the space capsule, and is now pervasively used in the localization industry, the radar industry, and robotics. It starts from a dynamical discrete-time non-linear law of the form

$$\mathbf{x}_{n+1} = f(\mathbf{x}_n,\ \mathbf{u}_n) + \mathbf{w}_n \tag{9}$$

where $\mathbf{x}_n$ denotes the state to be estimated, $\mathbf{u}_n$ is an input, and $\mathbf{w}_n$ is the process noise which is assumed Gaussian with zero mean and covariance matrix $\mathbf{Q}_n$. Assume side information is in the form of loose equality constraints $h(\mathbf{x}_n) \approx \mathbf{0}$ is available. It is then customary to generate a fictitious observation from the constraint function:

$$\mathbf{y}_n = h(\mathbf{x}_n) + \mathbf{n}_n, \tag{10}$$

and to feed the filter with the information that $\mathbf{y}_n = \mathbf{0}$ (pseudo-measurement) as first advocated by [29], see also [13,30] for application to visual inertial localization and general considerations. The noise is assumed to be a centered Gaussian $\mathbf{n}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{N}_n)$ where the covariance matrix $\mathbf{N}_n$ is set by the user and reflects the degree of validity of the information: the larger $\mathbf{N}_n$ the less confidence is put in the information.

Starting from an initial Gaussian belief about the state, $\mathbf{x}_0 \sim \mathcal{N}(\hat{\mathbf{x}}_0, \mathbf{P}_0)$ where $\hat{\mathbf{x}}_0$ represents the initial estimate and the covariance matrix $\mathbf{P}_0$ the uncertainty associated to it, the EKF alternates between two steps. At the propagation step, the estimate $\hat{\mathbf{x}}_n$ is propagated through model (9) with noise turned off, $\mathbf{w}_n = \mathbf{0}$, and the covariance matrix is updated through

$$\mathbf{P}_{n+1} = \mathbf{F}_n \mathbf{P}_n \mathbf{F}_n^T + \mathbf{G}_n \mathbf{Q}_n \mathbf{G}_n^T, \tag{11}$$

where $\mathbf{F}_n$, $\mathbf{G}_n$ are Jacobian matrices of $f(\cdot)$ with respect to $\mathbf{x}_n$ and $\mathbf{u}_n$. At the update step, pseudo-measurement is taken into account, and Kalman equations allow to update the estimate $\hat{\mathbf{x}}_{n+1}$ and its covariance matrix $\mathbf{P}_{n+1}$ accordingly.

To implement an EKF, the designer needs to determine the functions $f(\cdot)$ and $h(\cdot)$, and the associated noise matrices $\mathbf{Q}_n$ and $\mathbf{N}_n$. In this paper, noise parameters $\mathbf{Q}_n$ and $\mathbf{N}_n$ will be wholly learned by a neural network.

A. Defining the Dynamical Model $f(\cdot)$

We now need to assess the evolution of state variables (8). The evolution of $\mathbf{R}_n^{\text{IMU}}$, $\mathbf{p}_n^{\text{IMU}}$, $\mathbf{v}_n^{\text{IMU}}$, $\mathbf{b}_n^{\boldsymbol{\omega}}$ and $\mathbf{b}_n^{\mathbf{a}}$ is already given by the standard equations (3)-(7). The additional variables $\mathbf{R}_n^{\text{c}}$ and $\mathbf{p}_n^{\text{c}}$ represent the car frame with respect to the IMU. This car frame is rigidly attached to the car and encodes an unknown fictitious point where the pseudo-measurements of Section III-B are most advantageously made. As IMU is also rigidly attached to the car, and $\mathbf{R}_n^{\text{c}}$, $\mathbf{p}_n^{\text{c}}$ represent misalignment between IMU and car frame, they are approximately constant

$$\mathbf{R}_{n+1}^{\text{c}} = \mathbf{R}_n^{\text{c}} \exp((\mathbf{w}_n^{\mathbf{R}^{\text{c}}})_\times), \tag{12}$$

$$\mathbf{p}_{n+1}^{\text{c}} = \mathbf{p}_n^{\text{c}} + \mathbf{w}_n^{\mathbf{p}^{\text{c}}}. \tag{13}$$

where we let $\mathbf{w}_n^{\mathbf{R}^{\text{c}}}$, $\mathbf{w}_n^{\mathbf{p}^{\text{c}}}$ be centered Gaussian noises with small covariance matrices $\sigma^{\mathbf{R}^{\text{c}}}\mathbf{I}$, $\sigma^{\mathbf{p}^{\text{c}}}\mathbf{I}$ that will be learned during training. Noises $\mathbf{w}_n^{\mathbf{R}^{\text{c}}}$ and $\mathbf{w}_n^{\mathbf{p}^{\text{c}}}$ encode possible small variations through time of level arm due to the lack of rigidity stemming from dampers and shock absorbers.

B. Defining the Pseudo-Measurements $h(\cdot)$

Consider the different frames depicted on Figure 2. The velocity of the origin point of the car frame, expressed in the car frame, writes

$$\mathbf{v}_n^{\text{c}} = \begin{bmatrix} v_n^{\text{for}} \\ v_n^{\text{lat}} \\ v_n^{\text{up}} \end{bmatrix} = \mathbf{R}_n^{\text{c}T} \mathbf{R}_n^{\text{IMU}T} \mathbf{v}_n^{\text{IMU}} + (\boldsymbol{\omega}_n)_\times \mathbf{p}_n^{\text{c}}, \tag{14}$$

from basic screw theory, where $\mathbf{p}_n^{\text{c}} \in \mathbb{R}^3$ is the car to IMU level arm. In the car frame, we consider that the car lateral and vertical velocities are roughly null, that is, we generate two scalar pseudo observations of the form (10) that is,

$$\mathbf{y}_n = \begin{bmatrix} y_n^{\text{lat}} \\ y_n^{\text{up}} \end{bmatrix} = \begin{bmatrix} h^{\text{lat}}(\mathbf{x}_n) + \mathrm{n}_n^{\text{lat}} \\ h^{\text{up}}(\mathbf{x}_n) + \mathrm{n}_n^{\text{up}} \end{bmatrix} = \begin{bmatrix} v_n^{\text{lat}} \\ v_n^{\text{up}} \end{bmatrix} + \mathbf{n}_n, \tag{15}$$

where the noises $\mathbf{n}_= \left[\mathrm{n}_n^{\mathrm{lat}}, \mathrm{n}_n^{\mathrm{up}}\right]^T$ are assumed centered and Gaussian with covariance matrix $\mathbf{N}_n \in \mathbb{R}^{2\times 2}$. The filter is then fed with the pseudo-measurement that $y_n^{\mathrm{lat}} = y_n^{\mathrm{up}} = 0$.

Assumptions that v_n^{lat} and v_n^{up} are roughly null are common for cars moving forward on human made roads or wheeled robots moving indoor. Treating them as loose constraints, i.e., allowing the uncertainty encoded in $\mathbf{N}_n$ to be non strictly null, leads to much better estimates than treating them as strictly null [13].

It should be duly noted the vertical velocity v_n^{up} is expressed in the car frame, and thus the assumption it is roughly null generally holds for a car moving on a road even if the motion is 3D. It is quite different from assuming null vertical velocity in the world frame, which then boils down to planar horizontal motion.

The main point of the present work is that the validity of the null lateral and vertical velocity assumptions widely vary depending on what maneuver is being performed: for instance, v_n^{lat} is much larger in turns than in straight lines. The role of the noise parameter adapter of Section IV-A, based on AI techniques, will be to dynamically assess the parameter $\mathbf{N}_n$ that reflects confidence in the assumptions, as a function of past and present IMU measurements.

C. The Invariant Extended Kalman Filter (IEKF)

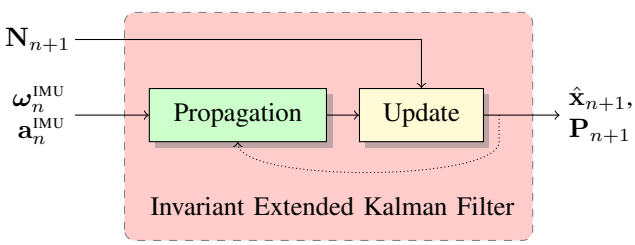

Fig. 3. Structure of the IEKF. The filter uses the noise parameter $\mathbf{N}_{n+1}$ of pseudo-measurements (15) to yield a real time estimate of the state $\hat{\mathbf{x}}_{n+1}$ along with covariance $\mathbf{P}_{n+1}$.

For inertial navigation, we advocate the use of a recent EKF variant, namely the Invariant Extended Kalman Filter (IEKF), see [4,5], that has recently given raise to a commercial aeronautics product [31] and to various successes in the field of visual inertial odometry [32]–[34]. We thus opt for an IEKF to perform the fusion between the IMU measurements (1)-(2) and (15) treated as pseudo-measurements. Its architecture, which is identical to the conventional EKF's, is recapped in Figure 3.

However, understanding in detail the IEKF [4] requires some background in Lie group geometry. The interested reader is referred to the Appendix where the exact equations are provided.

IV. Proposed AI-IMU Dead-Reckoning

This section describes our system for recovering all the variables of interest from IMU signals only. Figure 4 illustrates

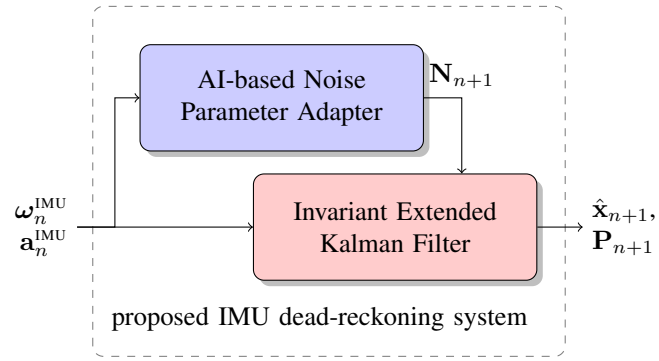

Fig. 4. Structure of the proposed system for inertial dead-reckoning. The measurement noise adapter feeds the filter with measurement covariance from raw IMU signals only.

the approach which consists of two main blocks summarized as follows:

- the filter integrates the inertial measurements (1)-(2) with dynamical model $f(\cdot)$ given by (3)-(7) and (12)-(13), and exploits (15) as measurements $h(\cdot)$ with covariance matrix $\mathbf{N}_n$ to refine its estimates;
- the noise parameter adapter determines in real-time the most suitable covariance noise matrix $\mathbf{N}_n$. This deep learning based adapter converts directly raw IMU signals (1)-(2) into covariance matrices $\mathbf{N}_n$ without requiring knowledge of any state estimate nor any other quantity.

The amplitude of process noise parameters $\mathbf{Q}_n$ are considered fixed by the algorithm, and are learned during training.

Note that the adapter computes covariances meant to improve localization accuracy, and thus the computed values may broadly differ from the actual statistical covariance of $\mathbf{y}_n$ (15), see Section V-D for more details. In this respect, our approach is related to [24] but the considered problem is more challenging: our state-space is of dimension 21 whereas [24] has a state-space of dimension only 3. Moreover, we compare our results in the sequel to state-of-the-art methods based on stereo cameras and LiDARs, and we show we may achieve similar results based on the moderately precise IMU only.

A. AI-based Measurement Noise Parameter Adapter

The measurement noise parameter adapter computes at each instant n the covariance $\mathbf{N}_{n+1}$ used in the filter update, see Figure 3. The base core of the adapter is a Convolutional Neural Network (CNN) [35]. The networks takes as input a window of N inertial measurements and computes

$$\mathbf{N}_{n+1} = \mathrm{CNN}\left(\{\boldsymbol{\omega}_i^{\mathrm{IMU}},\ \mathbf{a}_i^{\mathrm{IMU}}\}_{i=n-N}^{n}\right). \tag{16}$$

Our motivations for the above simple CNN-like architecture are threefold:

i) avoiding over-fitting by using a relatively small number of parameters in the network and also by making its outputs independent of state estimates;
ii) obtaining an interpretable adapter from which one can infer general and safe rules using reverse engineering,

e.g. to which extent must one inflate the covariance during turns, for e.g., generalization to all sorts of wheeled and commercial vehicles, see Section V-D;

$iii)$ letting the network be trainable. Indeed, as reported in [24], training is quite difficult and slow. Setting the adapter with a recurrent architecture [35] would make the training even much harder.

The complete architecture of the adapter consists of a bunch of CNN layers followed by a full layer outputting a vector $\mathbf{z}_n = \left[z_n^{\text{lat}}, z_n^{\text{up}}\right]^T \in \mathbb{R}^2$. The covariance $\mathbf{N}_{n+1} \in \mathbb{R}^{2\times 2}$ is then computed as

$$\mathbf{N}_{n+1} = \text{diag}\left(\sigma_{\text{lat}}^2 10^{\beta \tanh(z_n^{\text{lat}})},\ \sigma_{\text{up}}^2 10^{\beta \tanh(z_n^{\text{up}})}\right), \quad (17)$$

with $\beta \in \mathbb{R}_{>0}$, and where σ_{lat} and σ_{up} correspond to our initial guess for the noise parameters. The network thus may inflate covariance up to a factor 10^{β} and squeeze it up to a factor $10^{-\beta}$ with respect to its original values. Additionally, as long as the network is disabled or barely reactive (e.g. when starting training), we get $\mathbf{z}_n \approx \mathbf{0}$ and recover the initial covariance $\text{diag}\left(\sigma_{\text{lat}},\ \sigma_{\text{up}}\right)^2$.

Regarding process noise parameter $\mathbf{Q}_n$, we choose to fix it to a value $\mathbf{Q}$ and leave its dynamic adaptation for future work. However its entries are optimized during training, see Section IV-C.

B. Implementation Details

We provide in this section the setting and the implementation details of our method. We implement the full approach in Python with the PyTorch[1] library for the noise parameter adapter part. We set as initial values before training

$$\mathbf{P}_0 = \text{diag}\left(\sigma_0^{\mathbf{R}}\mathbf{I}_2, 0, \sigma_0^{\mathbf{v}}\mathbf{I}_2, \mathbf{0}_4, \sigma_0^{\mathbf{b}^{\boldsymbol{\omega}}}\mathbf{I}, \sigma_0^{\mathbf{b}^{\mathbf{a}}}\mathbf{I}, \sigma_0^{\mathbf{R}^{\mathbf{c}}}\mathbf{I}, \sigma_0^{\mathbf{p}^{\mathbf{c}}}\mathbf{I}\right)^2, \quad (18)$$

$$\mathbf{Q} = \text{diag}\left(\sigma_{\boldsymbol{\omega}}\mathbf{I},\ \sigma_{\mathbf{a}}\mathbf{I},\ \sigma_{\mathbf{b}^{\boldsymbol{\omega}}}\mathbf{I},\ \sigma_{\mathbf{b}^{\mathbf{a}}}\mathbf{I},\ \sigma_{\mathbf{R}^{\mathbf{c}}}\mathbf{I},\ \sigma_{\mathbf{p}^{\mathbf{c}}}\mathbf{I}\right)^2, \quad (19)$$

$$\mathbf{N}_n = \text{diag}\left(\sigma_{\text{lat}},\ \sigma_{\text{up}}\right)^2, \quad (20)$$

where $\mathbf{I} = \mathbf{I}_3$, $\sigma_0^{\mathbf{R}} = 10^{-3}\,\text{rad}$, $\sigma_0^{\mathbf{v}} = 0.3\,\text{m/s}$, $\sigma_0^{\mathbf{b}^{\boldsymbol{\omega}}} = 10^{-4}\,\text{rad/s}$, $\sigma_0^{\mathbf{b}^{\mathbf{a}}} = 3.10^{-2}\,\text{m/s}^2$, $\sigma_0^{\mathbf{R}^{\mathbf{c}}} = 3.10^{-3}\,\text{rad}$, $\sigma_0^{\mathbf{p}^{\mathbf{c}}} = 10^{-1}\,\text{m}$ in the initial error covariance $\mathbf{P}_0$, $\sigma_{\boldsymbol{\omega}} = 1.4.10^{-2}\,\text{rad/s}$, $\sigma_{\mathbf{a}} = 3.10^{-2}\,\text{m/s}^2$, $\sigma_{\mathbf{b}^{\boldsymbol{\omega}}} = 10^{-4}\,\text{rad/s}$, $\sigma_{\mathbf{b}^{\mathbf{a}}} = 10^{-3}\,\text{m/s}^2$, $\sigma_{\mathbf{R}^{\mathbf{c}}} = 10^{-4}\,\text{rad}$, $\sigma_{\mathbf{p}^{\mathbf{c}}} = 10^{-4}\,\text{m}$ for the noise propagation covariance matrix $\mathbf{Q}$, $\sigma_{\text{lat}} = 1\,\text{m/s}$, and $\sigma_{\text{up}} = 3\,\text{m/s}$ for the measurement covariance matrix. The zero values in the diagonal of $\mathbf{P}_0$ in (18) corresponds to a perfect prior of the initial yaw, position and zero vertical speed.

The adapter is a 1D temporal convolutional neural network with 2 layers. The first layer has kernel size 5, output dimension 32, and dilatation parameter 1. The second has kernel size 5, output dimension 32 and dilatation parameter 3, thus it set the window size equal to $N = 15$. The CNN is followed by a fully connected layer that output the scalars z^{lat} and z^{up}. Each activation function between two layers is a ReLU unit [35]. We define $\beta = 3$ in the right part of (17) which allows for each covariance element to be 10^3 higher or smaller than its original values.

C. Training

We seek to optimize the relative translation error t_{rel} computed from the filter estimates $\hat{\mathbf{x}}_n$, which is the averaged increment error for all possible sub-sequences of length $100\,\text{m}$ to $800\,\text{m}$.

Toward this aim, we first define the learnable parameters. It consists of the 6210 parameters of the adapter, along with the parameter elements of $\mathbf{P}_0$ and $\mathbf{Q}$ in (18)-(19), which add 12 parameters to learn. We then choose an Adam optimizer [36] with learning rate 10^{-4} that updates the trainable parameters. Training consists of repeating for a chosen number of epochs the following iterations:

$i)$ sample a part of the dataset;

$ii)$ get the filter estimates for then computing loss and gradient w.r.t. the learnable parameters;

$ii)$ update the learnable parameters with gradient and optimizer.

Following continual training [37], we suppose the number of epochs is huge, potentially infinite (in our application we set this number to 400). This makes sense for online training in a context where the vehicle gathers accurate ground-truth poses from e.g. its LiDAR system or precise GNSS. It requires careful procedures for avoiding over-fitting, such that we use dropout and data augmentation [35]. Dropout refers to ignoring units of the adapter during training, and we set the probability $p = 0.5$ of any CNN element to be ignored (set to zero) during a sequence iteration.

Regarding $i)$, we sample a batch of nine $1\,\text{min}$ sequences, where each sequence starts at a random arbitrary time. We add to data a small Gaussian noise with standard deviation 10^{-4}, a.k.a. data augmentation technique. We compute $ii)$ with standard backpropagation, and we finally clip the gradient norm to a maximal value of 1 to avoid gradient explosion at step $iii)$.

We stress the loss function consists of the relative translation error t_{rel}, i.e. we optimize parameters for improving the filter accuracy, disregarding the values actually taken by $\mathbf{N}_n$, in the spirit of [24].

V. Experimental Results

We evaluate the proposed method on the KITTI dataset [3], which contains data recorded from LiDAR, cameras and IMU, along with centimeter accurate ground-truth pose from different environments (e.g., urban, highways, and streets). The dataset contains 22 sequences for benchmarking odometry algorithms, 11 of them contain publicly available ground-truth trajectory, raw and synchronized IMU data. We download the raw data with IMU signals sampled at $100\,\text{Hz}$ ($dt = 10^{-2}\,\text{s}$) rather than the synchronized data sampled at $10\,\text{Hz}$, and discard seq. 03 since we did not find raw data for this sequence. The RT3003[2] IMU has announced gyro and accelerometer bias stability of respectively $36\,\text{deg/h}$ and $1\,\text{mg}$. The KITTI dataset has an online benchmarking system that ranks algorithms. However we could not submit our algorithm for online ranking since sequences used for ranking do not

[1]https://pytorch.org/

[2]https://www.oxts.com/

test seq.	length (km)	duration (s)	environment	IMLS [6]		ORB-SLAM2 [7]		IMU		**proposed**	
				t_{rel} (%)	r_{rel} (deg/km)	t_{rel} (%)	r_{rel} (deg/km)	t_{rel} (%)	r_{rel} (deg/km)	t_{rel} (%)	r_{rel} (deg/km)
01	2.6	110	highway	**0.82**	**1.0**	1.41	1.9	5.35	1.2	1.56	1.2
03	-	80	country	-	-	-	-	-	-	-	-
04	0.4	27	country	**0.33**	1.2	0.47	2.2	0.97	1.0	1.22	**0.4**
06	1.2	110	urban	**0.33**	**0.8**	0.73	2.2	5.78	1.9	1.57	1.9
07	0.7	110	urban	**0.33**	**1.5**	0.91	4.9	12.6	3.0	1.32	3.0
08	3.2	407	urban, country	**0.80**	**1.8**	1.03	3.0	549	5.6	1.08	3.2
09	1.7	159	urban, country	**0.55**	**1.2**	0.81	2.3	23.4	3.5	0.82	2.2
10	0.9	120	urban, country	**0.53**	1.7	**0.66**	3.1	4.58	2.5	1.05	2.5
average scores				**0.64**	**1.2**	0.99	2.6	171	3.1	0.97	2.3

Table 1. Results on [3]. IMU integration tends to drift or diverge, whereas the proposed method may be used as an alternative to LiDAR based (IMLS) and stereo vision based (ORB-SLAM2) methods, using only IMU information. Indeed, on average, our dead-reckoning solution performs better than ORB-SLAM2 and achieves a translational error being close to that of the LiDAR based method IMLS, which is ranked 3rd on the KITTI online benchmarking system. Data from seq. `03` was unavailable for testing algorithms, and sequences `00`, `02` and `05` are discussed separately in Section V-C. It should be duly noted IMLS, ORB-SLAM2, and the proposed AI-IMU algorithm, all use different sensors. The interest of ranking algorithms based on different information is debatable. Our goal here is rather to evidence that using data from a moderately precise IMU only, one can achieve similar results as state of the art systems based on imaging sensors, which is a rather surprising feature.

contain IMU data, which is reserved for training only. Our implementation is made open-source at:

`https://github.com/mbrossar/ai-imu-dr`.

A. Evaluation Metrics and Compared Methods

To assess performances we consider the two error metrics proposed in [3]:

1) Relative Translation Error (t_{rel}): which is the averaged relative translation increment error for all possible sub-sequences of length $100\,\mathrm{m}, \ldots, 800\,\mathrm{m}$, in percent of the traveled distance;

2) Relative Rotational Error (t_{rel}): that is the relative rotational increment error for all possible sub-sequences of length $100\,\mathrm{m}, \ldots, 800\,\mathrm{m}$, in degree per kilometer.

We compare four methods which alternatively use LiDAR, stereo vision, and IMU-based estimations:

- **IMLS** [6]: a recent state-of-the-art LiDAR-based approach ranked 3rd in the KITTI benchmark. The author provided us with the code after disabling the loop-closure module;
- **ORB-SLAM2** [7]: a popular and versatile library for monocular, stereo and RGB-D cameras that computes a sparse reconstruction of the map. We took the open-source code, disable loop-closure capability and evaluate the stereo algorithm without modifying any parameter;
- **IMU**: the direct integration of the IMU measurements based on (4)-(5), that is, pure inertial navigation;
- **proposed**: the proposed approach, that uses only the IMU signals and involves no other sensor.

B. Trajectory Results

We follow the same protocol for evaluating each sequence: i) we initialize the filter with parameters described in Section IV-B; ii) we train then the noise parameter adapter following Section IV-C for 400 epochs without the evaluated sequence (e.g. for testing seq. `10`, we train on seq. `00-09`) so that the noise parameter has *never* been confronted with the evaluated sequence; iii) we run the IMU-based methods on the full raw sequence with ground-truth initial configuration $(\mathbf{R}_0^{\text{IMU}}, \mathbf{v}_0^{\text{IMU}}, \mathbf{p}_0^{\text{IMU}})$, whereas we initialize remaining variables at zero ($\mathbf{b}_0^{\boldsymbol{\omega}} = \mathbf{b}_0^{\mathbf{a}} = \mathbf{p}_0^{\mathbf{c}} = \mathbf{0}$, $\mathbf{R}_0^{\mathbf{c}} = \mathbf{I}$); and iv) we get the estimates only on time corresponding to the odometry benchmark sequence. LiDAR and visual methods are directly evaluated on the odometry sequences.

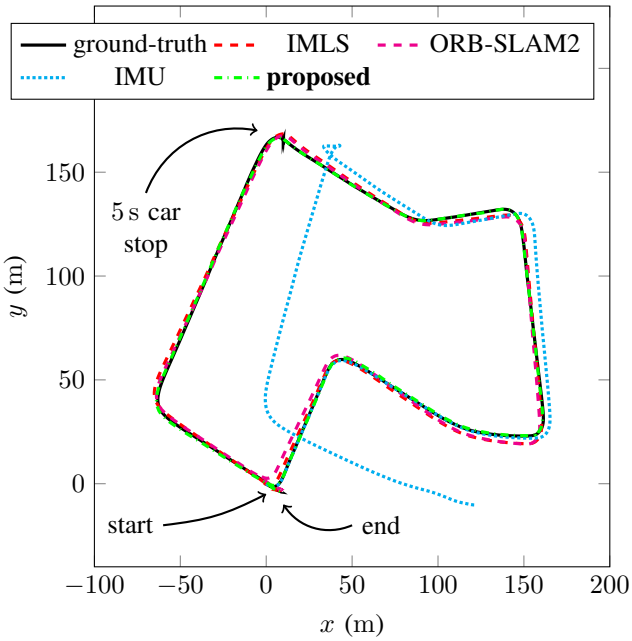

Fig. 5. Results on seq. `07` (drive #27, 2011/09/30) [3]. The proposed method competes with LiDAR and visual odometry methods, whereas the IMU integration broadly drifts after the car stops.

Results are averaged in Table 1 and illustrated in Figures 1, 5 and 6, where we exclude sequences `00`, `02` and `05`

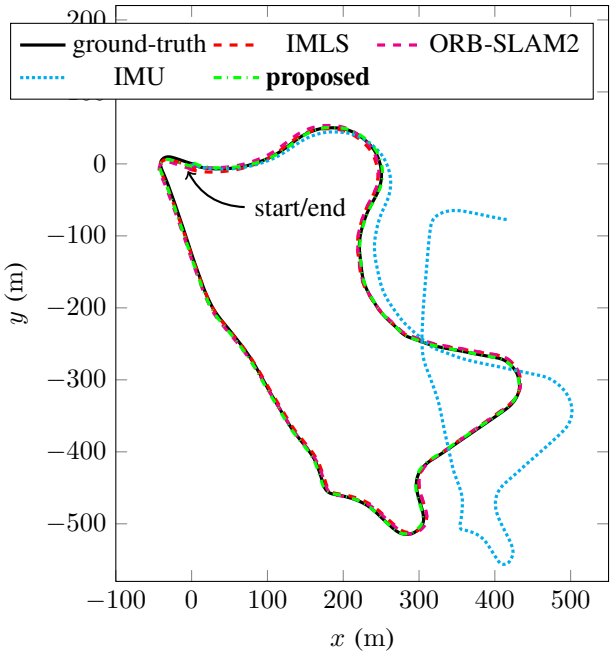

Fig. 6. Results on seq. 09 (drive #33, 2011/09/30) [3]. The proposed method competes with LiDAR and visual odometry methods, whereas the IMU integration drifts quickly after the first turn.

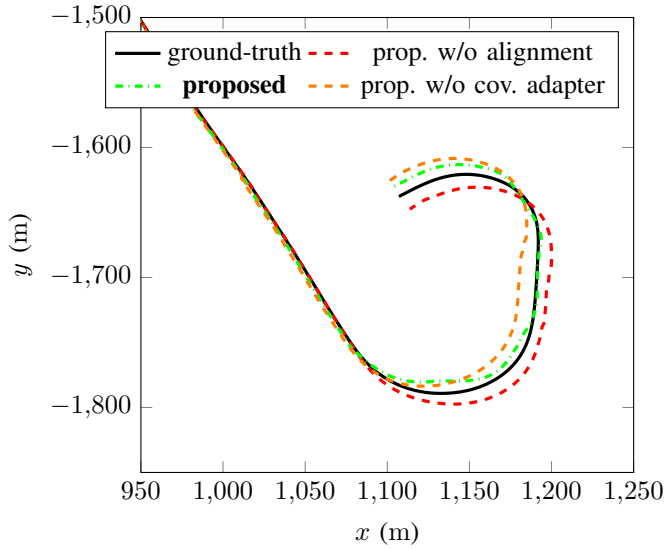

Fig. 7. End trajectory results on the highway seq. 01 (drive #42, 2011/10/30) [3]. Dynamically adapting the measurement covariance and considering misalignment between car and inertial frames enhance the performances of the proposed method from a translational error of 1.94% to one of 1.11%.

which contain problems with the data, and will be discussed separately in Section V-C. From these results, we see that:

- LiDAR and visual methods perform generally well in all sequences, and the LiDAR method achieves slightly better results than its visual counterpart;
- our method competes on average with the latter image based methods, see Table 1;
- directly integrating the IMU signals leads to rapid drift of the estimates, especially for the longest sequences but even for short periods;
- Our method looks unaffected by stops of the car, as in seq. 07, see Figure 5.

The results are remarkable as we use none of the vision sensors, nor wheel odometry. We only use the IMU, which moreover has moderate precision.

We also sought to compare our method to visual inertial odometry algorithms. However, we could not find open-source of such method that performs well on the full KITTI dataset. We tested [38] but the code in still under development (results sometimes diverge), and the authors in [39] evaluate their not open-source method for short sequences ($\leq 30\,\mathrm{s}$). The paper [33,40] evaluate their visual inertial odometry methods on seq. 08, both get a final error around $20\,\mathrm{m}$, which is four times what our method gets, with final distance to ground-truth of only $5\,\mathrm{m}$. This clearly evidences that methods taylored for ground vehicles [13,15] may achieve higher accuracy and robustness that general methods designed for smartphones, drones and aerial vehicles.

C. Results on Sequences 00, 02 and 05

Following the procedure described in Section V-B, the proposed method seems to have degraded performances on seq. 00, 02 and 05, see e.g. Figure 9. However, the behavior is wholly explainable: data are missing for a couple of seconds due to logging problems which appear both for IMU and ground-truth. This is illustrated in Figure 10 for seq. 02 where we plot available data over time. We observe jump in the IMU and ground-truth signals, that illustrate data are missing between $t = 1$ and $t = 3$. The problem was corrected manually when using those sequences in the training phase described in Section V.

Although those sequences could have been discarded due to logging problems, we used them for testing without correcting their problems. This naturally results in degraded performance, but also evidences our method is remarkably robust to such problems in spite of their inherent harmfulness. For instance, the $2\,\mathrm{s}$ time jump of seq. 02 results in estimate shift, but no divergence occurs for all that, see Figure 9.

D. Discussion

The performances are owed to three components: i) the use of a recent IEKF that has been proved to be well suited for IMU based localization; ii) incorporation of side information in the form of pseudo-measurements with dynamic noise parameter adaptation learned by a neural network; and iii) accounting for a "loose" misalignment between the IMU frame and the car frame.

As concerns i), it should be stressed the method is perfectly suited to the use of a conventional EKF and is easily adapted if need be. However we advocate the use of an IEKF owing to its accuracy and convergence properties. To illustrate the benefits of points ii) and iii), we consider two sub-versions

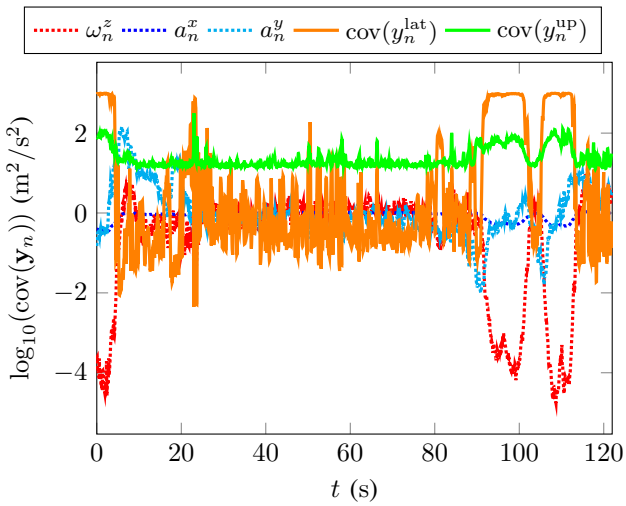

Fig. 8. Covariance values computed by the adapter on the highway seq. `01` (drive #42, 2011/10/30) [3]. We clearly observe a large increase in the covariance values when the car is turning between $t = 90\,\mathrm{s}$ and $t = 110\,\mathrm{s}$.

of the proposed algorithm. One without alignment, i.e. where $\mathbf{R}_n^{\mathrm{c}}$ and $\mathbf{p}_n^{\mathrm{c}}$ are not included in the state and fixed at their initial values $\mathbf{R}_n^{\mathrm{c}} = \mathbf{I}$, $\mathbf{p}_n^{\mathrm{c}} = \mathbf{0}$, and a second one that uses the static filter parameters (18)-(20).

End trajectory results for the highway seq. `01` are plotted in Figure 7, where we see that the two sub-version methods have trouble when the car is turning. Therefore their respective translational errors t_{rel} are higher than the full version of the proposed method: the proposed method achieves 1.11%, the method without alignment level arm achieves 1.65%, and the absence of covariance adaptation yields 1.94% error. All methods have the same rotational error $r_{rel} = 0.12\,\mathrm{deg}/\mathrm{m}$. This could be anticipated for the considered sequence since the full method has the same rotational error than standard IMU integration method.

As systems with equipped AI-based approaches may be hard to certify for commercial or industrial use [41], we note adaptation rules may be inferred from the AI-based adapter, and encoded in an EKF using pseudo-measurements. To this aim, we plot the covariances computed by the adapter for seq. `01` in Figure 8. The adapter clearly increases the covariances during the bend, i.e. when the gyro yaw rate is important. This is especially the case for the zero velocity measurement (15): its associated covariance is inflated by a factor of 10^2 between $t = 90\,\mathrm{s}$ and $t = 110\,\mathrm{s}$. This illustrates the kind of information the adapter has learned. Interestingly, we see large discrepancies may occur between the actual statistical uncertainty (which should clearly be below $100\,\mathrm{m}^2/\mathrm{s}^2$) and the inflated covariances whose values are computed for the sole purpose of filter's performance enhancement. Indeed, such a large noise parameter inflation indicates the AI-based part of the algorithm has learned and recognizes that pseudo-measurements have no value for localization at those precise moments, so the filter should barely consider them.

VI. Conclusion

This paper proposes a novel approach for inertial dead-reckoning for wheeled vehicles. Our approach exploits deep neural networks to dynamically adapt the covariance of simple assumptions about the vehicle motions which are leveraged in an invariant extended Kalman filter that performs localization, velocity and sensor bias estimation. The entire algorithm is fed with IMU signals only, and requires no other sensor. The method leads to surprisingly accurate results, and opens new perspectives. In future work, we would like to address the learning of the Kalman covariance matrices for images, and also the issue of generalization from one vehicle to another.

Acknowledgments

The authors would like to thank J-E. Deschaud for sharing the results of the IMLS algorithm [6], and Paul Chauchat for relevant discussions.

References

[1] G. Bresson, Z. Alsayed, L. Yu *et al.*, "Simultaneous Localization and Mapping: A Survey of Current Trends in Autonomous Driving," *IEEE Transactions on Intelligent Vehicles*, vol. 2, no. 3, pp. 194–220, 2017.

[2] OxTS, "Why it is Necessary to Integrate an Inertial Measurement Unit with Imaging Systems on an Autonomous Vehicle," https : / / www . oxts . com / technical-notes/why-use-ins-with-autonomous-vehicle/, 2018.

[3] A. Geiger, P. Lenz, C. Stiller *et al.*, "Vision meets robotics: The KITTI dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[4] A. Barrau and S. Bonnabel, "The Invariant Extended Kalman Filter as a Stable Observer," *IEEE Transactions on Automatic Control*, vol. 62, no. 4, pp. 1797–1812, 2017.

[5] ——, "Invariant Kalman Filtering," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, no. 1, pp. 237–257, 2018.

[6] J.-E. Deschaud, "IMLS-SLAM: Scan-to-Model Matching Based on 3D Data," in *International Conference on Robotics and Automation*. IEEE, 2018.

[7] R. Mur-Artal and J. Tardos, "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras," *Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[8] F. Tschopp, T. Schneider, A. W. Palmer *et al.*, "Experimental Comparison of Visual-Aided Odometry Methods for Rail Vehicles," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1815–1822, 2019.

[9] M. Mousa, K. Sharma, and C. G. Claudel, "Inertial Measurement Units-Based Probe Vehicles: Automatic Calibration, Trajectory Estimation, and Context Detection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 10, pp. 3133–3143, 2018.

[10] J. Wahlstrom, I. Skog, J. G. P. Rodrigues *et al.*, "Map-Aided Dead-Reckoning Using Only Measurements of Speed," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 3, pp. 244–253, 2016.

[11] A. Mahmoud, A. Noureldin, and H. S. Hassanein, "Integrated Positioning for Connected Vehicles," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–13, 2019.

[12] R. Karlsson and F. Gustafsson, "The Future of Automotive Localization Algorithms: Available, reliable, and scalable localization: Anywhere and anytime," *IEEE Signal Processing Magazine*, vol. 34, no. 2, pp. 60–69, 2017.

[13] K. Wu, C. Guo, G. Georgiou *et al.*, "VINS on Wheels," in *International Conference on Robotics and Automation*. IEEE, 2017, pp. 5155–5162.

[14] A. Brunker, T. Wohlgemuth, M. Frey *et al.*, "Odometry 2.0: A Slip-Adaptive EIF-Based Four-Wheel-Odometry Model for Parking," *IEEE Transactions on Intelligent Vehicles*, vol. 4, no. 1, pp. 114–126, 2019.

[15] F. Zheng and Y.-H. Liu, "SE(2)-Constrained Visual Inertial Fusion for Ground Vehicles," *IEEE Sensors Journal*, vol. 18, no. 23, pp. 9699–9707, 2018.

[16] M. Buczko, V. Willert, J. Schwehr *et al.*, "Self-Validation for Automotive Visual Odometry," in *Intelligent Vehicles Symposium*. IEEE, 2018, pp. 1–6.

[17] M. Kok, J. D. Hol, and T. B. Schön, "Using Inertial Sensors for Position and Orientation Estimation," *Foundations and Trends® in Signal Processing*, vol. 11, no. 1-2, pp. 1–153, 2017.

test seq.	length (km)	duration (s)	environment	IMLS [6]		ORB-SLAM2 [7]		IMU		**proposed**	
				t_{rel} (%)	r_{rel} (deg/km)	t_{rel} (%)	r_{rel} (deg/km)	t_{rel} (%)	r_{rel} (deg/km)	t_{rel} (%)	r_{rel} (deg/km)
00	3.7	454	urban	**0.50**	**1.8**	0.84	2.9	426	46.8	6.81	24.8
02	5.1	466	urban	**0.53**	**1.4**	0.79	2.7	346	8.7	3.37	3.8
05	2.2	278	urban	**0.32**	**1.4**	0.55	2.2	189	5.2	3.05	8.7

Table 2. Results on [3] on seq. 00, 02 and 05. The degraded results of the proposed method are wholly explained by a problem of missing data, see Section V-C and Figure 10.

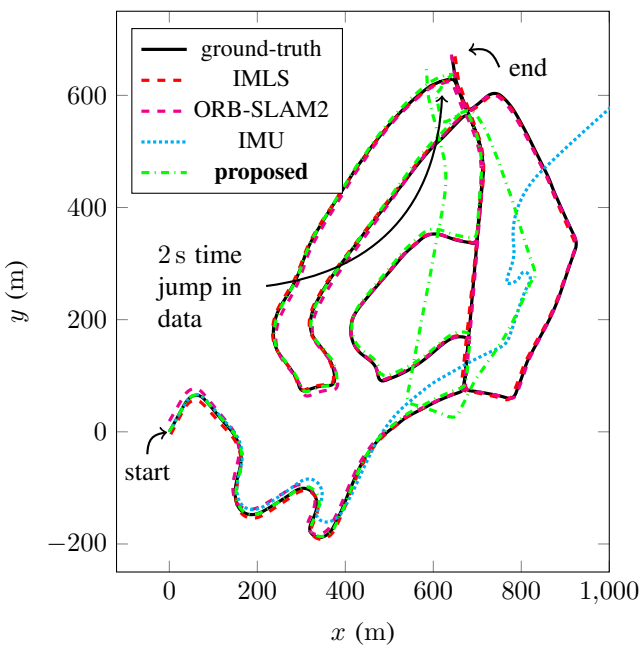

Fig. 9. Results on seq. 02 (drive #34, 2011/09/30) [3]. The proposed method competes with LiDAR and visual odometry methods until a problem in data occurs (2 seconds are missing). It is remarkable that the proposed method be robust to such trouble causing a shift estimates but no divergence.

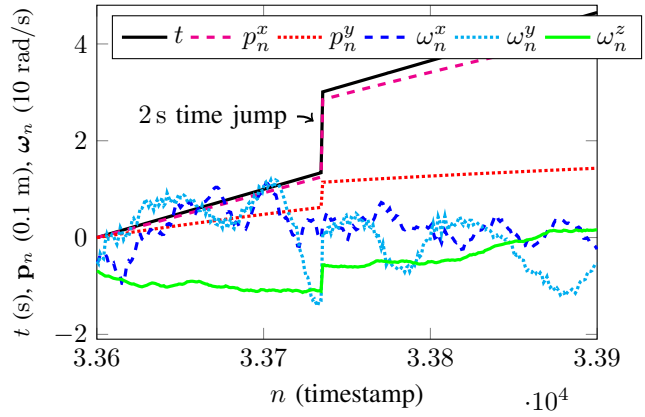

Fig. 10. Data on seq. 02 (drive #34, 2011/09/30) [3], as function of timestamps number n. A 2 s time jump happen around $n = 33750$, i.e. data have not been recorded during this jump. It leads to jump in ground-truth and IMU signals, causing estimate drift.

[18] A. Ramanandan, A. Chen, and J. Farrell, "Inertial Navigation Aiding by Stationary Updates," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 1, pp. 235–248, 2012.

[19] G. Dissanayake, S. Sukkarieh, E. Nebot *et al.*, "The Aiding of a Low-cost Strapdown Inertial Measurement Unit Using Vehicle Model Constraints for Land Vehicle Applications," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 5, pp. 731–747, 2001.

[20] M. Brossard, A. Barrau, and S. Bonnabel, "RINS-W: Robust Inertial Navigation System on Wheels," *Submitted to IROS 2019*, 2019. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02057117/file/main.pdf

[21] F. Aghili and C.-Y. Su, "Robust Relative Navigation by Integration of ICP and Adaptive Kalman Filter Using Laser Scanner and IMU," *IEEE/ASME Transactions on Mechatronics*, vol. 21, no. 4, pp. 2015–2026, 2016.

[22] H. Yan, Q. Shan, and Y. Furukawa, "RIDI: Robust IMU Double Integration," in *European Conference on Computer Vision*, 2018.

[23] C. Chen, X. Lu, A. Markham *et al.*, "IONet: Learning to Cure the Curse of Drift in Inertial Odometry," in *Conference on Artificial Intelligence AAAI*, 2018.

[24] T. Haarnoja, A. Ajay, S. Levine *et al.*, "Backprop KF: Learning Discriminative Deterministic State Estimators," in *Advances in Neural Information Processing Systems*, 2016.

[25] K. Liu, K. Ok, W. Vega-Brown *et al.*, "Deep Inference for Covariance Estimation: Learning Gaussian Noise Models for State Estimation," in *International Conference on Robotics and Automation*. IEEE, 2018.

[26] M. Brossard and S. Bonnabel, "Learning Wheel Odometry and IMU Errors for Localization," in *International Conference on Robotics and Automation*. IEEE, 2019.

[27] F. Castella, "An Adaptive Two-Dimensional Kalman Tracking Filter," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-16, no. 6, pp. 822–829, 1980.

[28] P. Abbeel, A. Coates, M. Montemerlo *et al.*, "Discriminative Training of Kalman Filters," in *Robotics: Science and Systems*, vol. 2, 2005.

[29] M. Tahk and J. Speyer, "Target tracking problems subject to kine-matic constraints," *IEEE Transactions on Automatic Control*, vol. 32, no. 2, pp. 324–326, 1990.

[30] D. Simon, "Kalman Filtering with State Constraints: A Survey of Linear and Nonlinear Algorithms," *IET Control Theory & Applications*, vol. 4, no. 8, pp. 1303–1318, 2010.

[31] A. Barrau and S. Bonnabel, "Aligment Method for an Inertial Unit," Patent 15/037,653, 2016.

[32] M. Brossard, S. Bonnabel, and A. Barrau, "Unscented Kalman Filter on Lie Groups for Visual Inertial Odometry," in *International Conference on Intelligent Robots and Systems*. IEEE/RSJ, 2018.

[33] S. Heo and C. G. Park, "Consistent EKF-Based Visual-Inertial Odometry on Matrix Lie Group," *IEEE Sensors Journal*, vol. 18, no. 9, pp. 3780–3788, 2018.

[34] R. Hartley, M. G. Jadidi, J. W. Grizzle *et al.*, "Contact-Aided Invariant Extended Kalman Filtering for Legged Robot State Estimation," in *Robotics Science and Systems*, 2018.

[35] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. The MIT press, 2016.

[36] D. P. Kingma and J. Ba, "ADAM: A Method for Stochastic Optimization," in *International Conference on Learning Representations*, 2014.

[37] G. Parisi, R. Kemker, J. Part *et al.*, "Continual Lifelong Learning with Neural Networks: A Review," *Neural Networks*, vol. 113, pp. 54–71, 2019.

[38] T. Qin, J. Pan, S. Cao *et al.*, "A General Optimization-based Framework for Local Odometry Estimation with Multiple Sensors," 2019.

[39] M. Ramezani and K. Khoshelham, "Vehicle Positioning in GNSS-Deprived Urban Areas by Stereo Visual-Inertial Odometry," *IEEE Transactions on Intelligent Vehicles*, vol. 3, no. 2, pp. 208–217, 2018.

[40] S. Heo, J. Cha, and C. G. Park, "EKF-Based Visual Inertial Navigation Using Sliding Window Nonlinear Optimization," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–10, 2018.

[41] N. Sünderhauf, O. Brock, W. Scheirer *et al.*, "The limits and potentials of deep learning for robotics," *The International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 405–420, 2018.

APPENDIX A

The Invariant Extended Kalman Filter (IEKF) [4,5] is an EKF based on an alternative state error. One must define a linearized error, an underlying group to derive the exponential map, and then methodology is akin to the EKF's.

1) Linearized Error: the filter state $\mathbf{x}_n$ is given by (8). The state evolution is given by the dynamics (5)-(7) and (12)-(13), see Section II. Along the lines of [4], variables $\boldsymbol{\chi}_n^{\text{IMU}} := (\mathbf{R}_n^{\text{IMU}}, \mathbf{v}_n^{\text{IMU}}, \mathbf{p}_n^{\text{IMU}})$ are embedded in the Lie group $SE_2(3)$ (see Appendix B for the definition of $SE_2(3)$ and its exponential map). Then biases vector $\mathbf{b}_n = \left[\mathbf{b}_n^{\boldsymbol{\omega}T}, \mathbf{b}_n^{\mathbf{a}T}\right]^T \in \mathbb{R}^6$ is merely treated as a vector, that is, as an element of $\mathbb{R}^6$ viewed as a Lie group endowed with standard addition, $\mathbf{R}_n^{\mathsf{c}}$ is treated as element of Lie group $SO(3)$, and $\mathbf{p}_n^{\mathsf{c}} \in \mathbb{R}^3$ as a vector. Once the state is broken into several Lie groups, the linearized error writes as the concatenation of corresponding linearized errors, that is,

$$\mathbf{e}_n = \left[\boldsymbol{\xi}_n^{\text{IMU}T} \quad \mathbf{e}_n^{\mathbf{b}T} \quad \boldsymbol{\xi}_n^{\mathbf{R}^{\mathsf{c}}T} \quad \mathbf{e}_n^{\mathbf{p}^{\mathsf{c}}T}\right]^T \sim \mathcal{N}\left(\mathbf{0}, \mathbf{P}_n\right), \tag{21}$$

where state uncertainty $\mathbf{e}_n \in \mathbb{R}^{21}$ is a zero-mean Gaussian variable with covariance $\mathbf{P}_n \in \mathbb{R}^{21\times 21}$. As (15) are measurements expressed in the robot's frame, they lend themselves to the Right IEKF methodology. This means each linearized error is mapped to the state using the corresponding Lie group exponential map, and multiplying it *on the right* by elements of the state space. This yields:

$$\boldsymbol{\chi}_n^{\text{IMU}} = \exp_{SE_2(3)}\left(\boldsymbol{\xi}_n^{\text{IMU}}\right)\hat{\boldsymbol{\chi}}_n^{\text{IMU}}, \tag{22}$$

$$\mathbf{b}_n = \hat{\mathbf{b}}_n + \mathbf{e}_n^{\mathbf{b}}, \tag{23}$$

$$\mathbf{R}_n^{\mathsf{c}} = \exp_{SO(3)}\left(\boldsymbol{\xi}_n^{\mathbf{R}^{\mathsf{c}}}\right)\hat{\mathbf{R}}_n^{\mathsf{c}}, \tag{24}$$

$$\mathbf{p}_n^{\mathsf{c}} = \hat{\mathbf{p}}_n^{\mathsf{c}} + \mathbf{e}_n^{\mathbf{p}^{\mathsf{c}}}, \tag{25}$$

where $\hat{(\cdot)}$ denotes estimated state variables.

2) Propagation Step: we apply (5)-(7) and (12)-(13) to propagate the state and obtain $\hat{\mathbf{x}}_{n+1}$ and associated covariance through the Riccati equation (11) where the Jacobians $\mathbf{F}_n$, $\mathbf{G}_n$ are related to the evolution of error (21) and write:

$$\mathbf{F}_n = \mathbf{I}_{21\times 21} + \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathbf{R}_n & \mathbf{0} & \mathbf{0}_{3\times 6} \\ (\mathbf{g})_\times & \mathbf{0} & \mathbf{0} & -(\mathbf{v}_n^{\text{IMU}})_\times \mathbf{R}_n & -\mathbf{R}_n & \mathbf{0}_{3\times 6} \\ \mathbf{0} & \mathbf{I}_3 & \mathbf{0} & -(\mathbf{p}_n^{\text{IMU}})_\times \mathbf{R}_n & \mathbf{0} & \mathbf{0}_{3\times 6} \\ & & & \mathbf{0}_{12\times 21} & & \end{bmatrix} dt, \tag{26}$$

$$\mathbf{G}_n = \begin{bmatrix} \mathbf{R}_n & \mathbf{0} & \mathbf{0}_{3\times 12} \\ (\mathbf{v}_n^{\text{IMU}})_\times \mathbf{R}_n & \mathbf{R}_n & \mathbf{0}_{3\times 12} \\ (\mathbf{p}_n^{\text{IMU}})_\times \mathbf{R}_n & \mathbf{0} & \mathbf{0}_{3\times 12} \\ \mathbf{0}_{12\times 3} & \mathbf{0}_{12\times 3} & \mathbf{I}_{12\times 12} \end{bmatrix} dt, \tag{27}$$

with $\mathbf{R}_n = \mathbf{R}_n^{\text{IMU}}$, $\mathbf{0} = \mathbf{0}_{3\times 3}$, and $\mathbf{Q}_n$ denotes the classical covariance matrix of the process noise as in Section IV-B.

3) Update Step: the measurement vector $\mathbf{y}_{n+1}$ is computed by stacking the motion information

$$\mathbf{y}_{n+1} = \begin{bmatrix} v_{n+1}^{\text{lat}} \\ v_{n+1}^{\text{up}} \end{bmatrix} = \mathbf{0}, \tag{28}$$

with assessed uncertainty a zero-mean Gaussian variable with covariance $\mathbf{N}_{n+1} = \text{cov}\left(\mathbf{y}_{n+1}\right)$. We then compute an updated state $\hat{\mathbf{x}}_{n+1}^+$ and updated covariance $\mathbf{P}_{n+1}^+$ following the IEKF methodology, i.e. we compute

$$\mathbf{S} = \left(\mathbf{H}_{n+1}\mathbf{P}_{n+1}\mathbf{H}_{n+1}^T + \mathbf{N}_{n+1}\right), \tag{29}$$

$$\mathbf{K} = \mathbf{P}_{n+1}\mathbf{H}_{n+1}^T/\mathbf{S}, \tag{30}$$

$$\mathbf{e}^+ = \mathbf{K}\left(\mathbf{y}_{n+1} - \hat{\mathbf{y}}_{n+1}\right), \tag{31}$$

$$\hat{\boldsymbol{\chi}}_{n+1}^{\text{IMU}+} = \exp_{SE_2(3)}\left(\boldsymbol{\xi}^{\text{IMU}+}\right)\hat{\boldsymbol{\chi}}_{n+1}^{\text{IMU}}, \tag{32}$$

$$\mathbf{b}_{n+1}^+ = \mathbf{b}_{n+1} + \mathbf{e}^{\mathbf{b}+} \tag{33}$$

$$\hat{\mathbf{R}}_{n+1}^{\mathsf{c}+} = \exp_{SO(3)}\left(\boldsymbol{\xi}^{\mathbf{R}^{\mathsf{c}}+}\right)\hat{\mathbf{R}}_{n+1}^{\mathsf{c}}, \tag{34}$$

$$\hat{\mathbf{p}}_{n+1}^{\mathsf{c}+} = \hat{\mathbf{p}}_{n+1}^{\mathsf{c}} + \mathbf{e}^{\mathbf{p}^{\mathsf{c}}+}, \tag{35}$$

$$\mathbf{P}_{n+1}^+ = \left(\mathbf{I}_{21} - \mathbf{K}\mathbf{H}_{n+1}\right)\mathbf{P}_{n+1}, \tag{36}$$

summarized as Kalman gain (30), state innovation (31), state update (32)-(35) and covariance update (36), where $\mathbf{H}_{n+1}$ is the measurement Jacobian matrix with respect to linearized error (21) and thus given as:

$$\mathbf{H}_n = \mathbf{A}\left[\mathbf{0} \quad \mathbf{R}_n^{\text{IMU}T} \quad \mathbf{0} \quad -(\mathbf{p}_n^{\mathsf{c}})_\times \quad \mathbf{0} \quad \mathbf{B} \quad \mathbf{C}\right], \tag{37}$$

where $\mathbf{A} = [\mathbf{I}_2 \ \mathbf{0}_2]$ selects the two first row of the right part of (37), $\mathbf{B} = \mathbf{R}_n^{\mathsf{c}T}\mathbf{R}_n^{\text{IMU}T}\left(\mathbf{v}_n^{\text{IMU}}\right)_\times$ and $\mathbf{C} = -(\boldsymbol{\omega}_n^{\text{IMU}} - \mathbf{b}_n^{\boldsymbol{\omega}})_\times$.

APPENDIX B

The Lie group $SE_2(3)$ is an extension of the Lie group $SE(3)$ and is described as follows, see [4] where it was first introduced. A 5×5 matrix $\boldsymbol{\chi}_n \in SE_2(3)$ is defined as

$$\boldsymbol{\chi}_n = \begin{bmatrix} \mathbf{R}_n & \mathbf{v}_n & \mathbf{p}_n \\ \mathbf{0}_{2\times 3} & & \mathbf{I}_2 \end{bmatrix} \in SE_2(3). \tag{38}$$

The uncertainties $\boldsymbol{\xi}_n \in \mathbb{R}^9$ are mapped to the Lie algebra $\mathfrak{se}_2(3)$ through the transformation $\boldsymbol{\xi}_n \mapsto \boldsymbol{\xi}_n^\wedge$ defined as

$$\boldsymbol{\xi}_n = \left[\boldsymbol{\xi}_n^{\mathbf{R}T}, \ \boldsymbol{\xi}_n^{\mathbf{v}T}, \ \boldsymbol{\xi}_n^{\mathbf{p}T}\right]^T, \tag{39}$$

$$\boldsymbol{\xi}_n^\wedge = \begin{bmatrix} \left(\boldsymbol{\xi}_n^{\mathbf{R}}\right)_\times & \boldsymbol{\xi}_n^{\mathbf{v}} & \boldsymbol{\xi}_n^{\mathbf{p}} \\ & \mathbf{0}_{2\times 5} & \end{bmatrix} \in \mathfrak{se}_2(3), \tag{40}$$

where $\boldsymbol{\xi}_n^{\mathbf{R}} \in \mathbb{R}^3$, $\boldsymbol{\xi}_n^{\mathbf{v}} \in \mathbb{R}^3$ and $\boldsymbol{\xi}_n^{\mathbf{p}} \in \mathbb{R}^3$. The closed-form expression for the exponential map is given as

$$\exp_{SE_2(3)}\left(\boldsymbol{\xi}_n\right) = \mathbf{I} + \boldsymbol{\xi}_n^\wedge + a(\boldsymbol{\xi}_n^\wedge)^2 + b(\boldsymbol{\xi}_n^\wedge)^3, \tag{41}$$

where $a = \frac{1-\cos\left(\|\boldsymbol{\xi}_n^{\mathbf{R}}\|\right)}{\|\boldsymbol{\xi}_n^{\mathbf{R}}\|}$ and $b = \frac{\|\boldsymbol{\xi}_n^{\mathbf{R}}\| - \sin\left(\|\boldsymbol{\xi}_n^{\mathbf{R}}\|\right)}{\|\boldsymbol{\xi}_n^{\mathbf{R}}\|^3}$, and which inherently uses the exponential of $SO(3)$, defined as

$$\exp_{SO(3)}\left(\boldsymbol{\xi}_n^{\mathbf{R}}\right) = \exp\left(\left(\boldsymbol{\xi}_n^{\mathbf{R}}\right)_\times\right) \tag{42}$$

$$= \mathbf{I} + \left(\boldsymbol{\xi}_n^{\mathbf{R}}\right)_\times + a\left(\boldsymbol{\xi}_n^{\mathbf{R}}\right)_\times^2. \tag{43}$$