HAL
open science

Learning Wheel Odometry and IMU Errors for Localization

Martin Brossard, Silvere Bonnabel

To cite this version:

Martin Brossard, Silvere Bonnabel. Learning Wheel Odometry and IMU Errors for Localization. International Conference on Robotics and Automation (ICRA), May 2019, Montreal, Canada. hal-01874593v2

HAL Id: hal-01874593

https://hal.science/hal-01874593v2

Submitted on 2 Mar 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.

Learning Wheel Odometry and IMU Errors for Localization

Martin BROSSARD and Silvère BONNABEL
MINES ParisTech, PSL Research University, Centre for Robotics, 60 Boulevard Saint-Michel, 75006 Paris, France

***Abstract*—Odometry techniques are key to autonomous robot navigation, since they enable self-localization in the environment. However, designing a robust odometry system is particularly challenging when camera and LiDAR are uninformative or unavailable. In this paper, we leverage recent advances in deep learning and variational inference to correct dynamical and observation models for state-space systems. The methodology trains Gaussian processes on the residual between the original model and the ground truth, and is applied on publicly available datasets for robot navigation based on two wheel encoders, a fiber optic gyro, and an Inertial Measurement Unit (IMU). We also propose to build an Extended Kalman Filter (EKF) on the learned model using wheel speed sensors and the fiber optic gyro for state propagation, and the IMU to update the estimated state. Experimental results clearly demonstrate that the (learned) corrected models and EKF are more accurate than their original counterparts.**

***Index Terms*—Gaussian process, odometry estimation, variational inference, Kalman filter**

I. INTRODUCTION

Self-localization and odometry based dead-reckoning estimation are a backbone for numerous robotics applications. However, designing an efficient odometry system based on mechanics is an engineering challenge which leads to complex models [1–3], whose performances still depend on unmodeled effects, e.g., in slippery and unequal terrains such as outdoors or for planetary rover exploration [1].

The caveats of wheel speed based odometry have in part prompted the recent interest in visual inertial navigation systems [4–6]. State-of-the-art odometry systems rely on a camera or a LiDAR which may be coupled with wheel speed and/or inertial sensors [4–9]. When camera and LiDAR become uninformative [8], that is, at night, in the presence of snow, or in military applications where active sensors are prohibited, localization based on wheel encoders and inertial sensors proves useful to select key-frames and more generally to obtain robust estimates [6].

This paper introduces a method to correct a state-space observed dynamical model by leveraging recent advances from the deep learning and variational inference communities [10–12]. Along the lines of [13,14], we combine the deterministic model with a Gaussian Process (GP) model to give more accurate state predictions. Deep kernel learning and stochastic variational inference [15–17] are then used for optimizing the GP parameters from a set of training data and to cope with high dimensionality in the input sequence and large amount of training data. We summarize our contributions as:

- Training a GP to learn the residuals between ground truth and deterministic models based on wheel speed sensors, or alternatively IMU;
- Using the recent methodology [16,17] from machine learning based on stochastic variational inference [15] and deep neural networks to ensure scalability of the approach;
- Experimental demonstration on recent datasets [18,19] that efficient odometry corrections may be learned both for a consumer car and a Segway;
- Building an Extended Kalman Filter (EKF) based on the corrected model and using automatic differentiation [20] for Jacobians computation to address robot localization.

A. Related Works

GPs have proved consequential in machine leaning and are used as a practical tool to solve various robotics problems [12,21], notably inverse dynamics learning [22]. Non-systematic errors have be shown to be partly identifiable, as in [13,14] for the the control of a blimp, and recent works advocate the use of GPs for learning error residuals such as LiDAR bias [23]. Close of our work is the recent reference [1], which models the odometry error as residual between a realistic parametric model and odometry output, and trains a GP that serves to efficiently compute image key-frames at low frequency. The main difference with our approach is that we inspire from deep kernel learning to improve scalability and accuracy [17], address the problem through multi-output GP to account for correlation in the residual components, and provide novel experimental results for a Segway and a customer car, whereas [1] is concerned with a 6 wheels ExoMars Test Rover. Moreover, in contrast to [1] we also address correction of the IMU outputs, and use corrections for EKF-based localization.

Deep learning and recently variational inference [11,15] are gaining much interest in robotics. Deep neural networks are used in [24] for odometry estimation of an autonomous electric cart, and in [25] for learning corrections for a specific estimator, sensor and environment through a deep pose correction network. [26] applies deep learning in an end-to-end manner for pure inertial odometry estimation, and obtains extremely low drift estimates on shopping trolley or baby-stroller trajectories. [27] studies visual odometry from the perspective of end-to-end deep learning. [28] combines a machine learning technique with an EKF for fusion of wheel speed sensors and GPS. Finally, [29] considers the

problem of system identification of helicopter dynamics, and poses the dynamics modeling problem as a high-dimensional regression problem which is solved with the help of stochastic variational inference.

B. Paper's Organization

Section II presents the physical model for robot motion based on wheel speeds an one axis FoG gyro. Section III introduces our GP-based model correction. Section IV shows experimental evaluation of the method using recent datasets. Section V evidences that the corrected models may be used to improve the estimates of an EKF that combines wheel speed sensors and fiber optic gyro with IMU for localization. The codes for reproducing the results of the paper are available at `https://github.com/CAOR-MINES-ParisTech/lwoi`.

II. Odometry Model

We consider a navigating robot equipped with: (i) two wheel encoders; (ii) an accurate one axis gyro such as a Fiber optic Gyro (FoG); and (iii) low-cost three axis gyros and accelerometers embedded in a commodity IMU. The wheel encoders measure the wheel angular velocity, the FoG gyro obtains accurate heading velocity. At this stage, the IMU is not used in our odometry model, but will prove useful at Section V.

The state is defined as the position, orientation, and angular velocities of the robot, i.e.,

$$\mathbf{x}_n = (x_n,\ y_n,\ z_n,\ \phi_n,\ \theta_n,\ \psi_n,\ p_n,\ q_n,\ r_n)^T, \tag{1}$$

where (x_n, y_n, z_n) represents the robot's position, $(\phi_n, \theta_n, \psi_n)$ are the Euler angles that parameterize the rotation matrix $\mathbf{R}_n$ whose columns are the axes of the robot frame, and (p_n, q_n, r_n) are the body-frame angular rates.

Let δt be the wheel encoder time rate, and assume time stamp n corresponds to time $n\delta t$. The wheel speed and gyro measurements are $\mathbf{u}_n = (v_l, v_r, \delta\psi)^T$, where v_l and v_r are the vehicle speed according to respectively the left and right wheels, and $\delta\psi$ is the change in heading measured by the FoG gyro. The state evolution model is then defined through the following kinematic equation under planar motion and constant angular velocity assumptions:

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \mathbf{z}_n \delta t + \mathbf{w}_n, \tag{2}$$

with $\mathbf{z}_n^T = \left(v\cos(\psi_n), v\sin(\psi_n), 0, \dot{\phi}_n,\ \dot{\theta}_n, \delta\psi/\delta t, 0, 0, 0\right)$ and where $v = (v_r + v_l)/2$ is the speed of the vehicle at the center of the wheelbase, and $\mathbf{w}_n$ is a Gaussian noise. See [18] for the expressions of the roll and pitch velocities $\dot{\phi}_n$ and $\dot{\theta}_n$.

III. Gaussian Process Estimation of Dynamical Systems Modeling Errors

A. Preliminaries

A Gaussian process [12,21] defines distribution

$$f(\cdot) \sim \mathcal{GP}\left(m(\cdot),\ k(\cdot,\cdot)\right) \tag{3}$$

over functions $f : \mathbb{R}^l \to \mathbb{R}$, with $m : \mathbb{R}^l \to \mathbb{R}$ and $k : \mathbb{R}^l \times \mathbb{R}^l \to \mathbb{R}_{\geq 0}$, and characterized by the fact that given any $\mathbf{u}_1, \ldots, \mathbf{u}_n$ the vector $f(\mathbf{u}_1), \ldots, f(\mathbf{u}_n)$ is assumed to be a multivariate Gaussian with $\mathbb{E}\left[f(\mathbf{u}_i)\right] = m(\mathbf{u}_i)$ and $\mathrm{Cov}\left(f(\mathbf{u}_i), f(\mathbf{u}_j)\right) = k(\mathbf{u}_i, \mathbf{u}_j)$. The training set $D = \{\mathbf{u}_1, \ldots, \mathbf{u}_d; \mathbf{z}_1, \ldots, \mathbf{z}_d\}$ consists of d input-output pairs under the assumption that $\mathbf{z}_i = f(\mathbf{u}_i) + \boldsymbol{\epsilon}_i$, with $\boldsymbol{\epsilon}_i \sim \mathcal{N}(0, \sigma^2)$ a Gaussian noise. For convenience we denote $\mathbf{U} = (\mathbf{u}_1, \ldots, \mathbf{u}_d)$, $\mathbf{Z} = (\mathbf{z}_1, \ldots, \mathbf{z}_d)$ and let the Gram matrix $\mathbf{K}$ be defined by $\mathbf{K}_{ij} = k(\mathbf{u}_i, \mathbf{u}_j)$. In this paper $m(\cdot)$ will be assumed known and fixed, and $k(\cdot,\cdot)$ belongs to a class of covariance functions [21] parameterized by some parameters $\boldsymbol{\theta}$. Given the training set D, model estimation is done through maximization of the log likelihood $\log p\left(\mathbf{Z}|\mathbf{U}\right)$ which is easily expressed (and maximized) as a function of hyperparameters $(\boldsymbol{\theta}, \sigma^2)$ yielding optimized covariance matrix $\mathbf{K}_*$ and variance σ_*^2. When considering a new input $\mathbf{u}'$, inference is performed by Gaussian conditioning [21]

$$p\left(f(\mathbf{u}')|\mathbf{Z}, \mathbf{U}\right) = \mathcal{N}(\boldsymbol{\mu}', \boldsymbol{\Sigma}'), \text{ where} \tag{4}$$

$$\boldsymbol{\mu}' = m(\mathbf{u}') + \mathbf{K}_*(\mathbf{u}', \mathbf{U})\left[\mathbf{K}_*(\mathbf{U}, \mathbf{U}) + \sigma_*^2\mathbf{I}\right]^{-1}(\mathbf{Z} - m(\mathbf{U})),$$

$$\boldsymbol{\Sigma}' = \mathbf{K}_*(\mathbf{u}', \mathbf{u}') - \mathbf{K}_*(\mathbf{u}', \mathbf{U})\left[\mathbf{K}_*(\mathbf{U}, \mathbf{U}) + \sigma_*^2\mathbf{I}\right]^{-1}\mathbf{K}_*(\mathbf{U}, \mathbf{u}').$$

Note that, for a training set of size d, exact GP inference requires a matrix inversion having complexity $O(d^3)$.

B. Gaussian Process for Dynamical Discrete Time Systems

Consider a discrete time dynamical system

$$\mathbf{x}_{n+1} = \mathbf{x}_n \boxplus f\left(\mathbf{u}_n,\ \mathbf{w}_n\right), \tag{5}$$

where $\mathbf{x}_n \in \mathcal{X}$ denotes the state, $\mathbf{u}_n$ the control input, $\mathbf{w}_n \sim \mathcal{N}\left(\mathbf{0}, \mathbf{Q}_n\right)$ a Gaussian noise. The operator $\boxplus : \mathcal{X} \times \mathcal{X} \to \mathcal{X}$ symbolizes state addition that extends the vector space addition to a combination of vector and orientation (quaternion) additions. Furthermore, we assume that the time rate of the state model is slower than the time rate of input measurements, such that the input at time n

$$\mathbf{u}_n = \left(\mathbf{u}_{n,0},\ \ldots,\ \mathbf{u}_{n,k-1}\right) \tag{6}$$

contains a sequence of k sub-inputs $\mathbf{u}_{n,i}$, $i = 0, \ldots, k-1$ and we define $\Delta t = k\delta t$ as the time between the instants $n-1$ and n, whereas we let δt being the time between two consecutive sub-inputs $\mathbf{u}_{n,i-1}$ and $\mathbf{u}_{n,i}$. Treating the input stream $\mathbf{u}_{n,i}$ into independent windows of the form (6) is well suited to machine learning approaches [25,26], and also particularly suits well pseudo-measurements such as a planar-motion constraints, see [8,30].

We can then model the increment function $f(\cdot)$ as non-zero mean Gaussian process

$$f(\cdot) \sim \mathcal{GP}\left(m_f(\cdot),\ k_f(\cdot,\cdot)\right). \tag{7}$$

Model (2) (with noise turned off) is reasonable and we postulate it provides the mean $m_f(\cdot)$ along the lines of [1,13].

Looking at (5), training data outputs are of the form $\mathbf{x}_{i+1} \boxminus \mathbf{x}_i$ (a proper sense must be given depending on operation $\boxplus$), and thus the training set writes $D_f = \{\mathbf{u}_1, \ldots, \mathbf{u}_d; \mathbf{z}_1, \ldots, \mathbf{z}_d\}$ with $\mathbf{z}_i = \mathbf{x}_{i+1} \boxminus \mathbf{x}_i$. The GP machinery of Section III-A can then be used, and the hyperparameters to be learnt consist of the kernel $k_f(\cdot,\cdot)$ and the likelihood $p(D_f|f)$, i.e., the magnitude of the noise.

C. Multi-Output Gaussian Processes

The standard basic GP approach recapped in Section III-A focuses on scalar outputs GP model, whereas our outputs consist of p dimensional increments of the state (5), with $p = \dim \mathcal{X}$, and are thus multi-dimensional. In contrast to e.g., [1,23], we do not model and treat each degree of freedom separately, but propose instead to address directly GP regression with vector outputs of dimension p. In odometry model correction, it makes particularly sense for pose residual where the orientation error highly influences the translation residual. We thus resort to a multi-output model where correlations between output components are explicitly represented through a shared input space [31]. The hyperparameter σ^2 (which leads to diagonal output covariance noise) is then replaced with a semidefinite positive (covariance) matrix $\tilde{\mathbf{\Sigma}}$.

D. Neural Networks for Kernel Learning

It is of major importance to take into account:

(i) The sequential and high dimensional nature of one single input $\mathbf{u}_n$, see (6): considering e.g. a class of radial basis functions for the kernel would be inefficient for accurate inference [21];

(ii) The computational complexity of (4) which scales as $O(d^3)$, rendering it computationally intractable for large datasets.

To fix ideas, in the dataset [18] of Section IV, we dispose of $d = 1.1 \cdot 10^5$ training points $\mathbf{u}_i$, with each $\mathbf{u}_i \in \mathbb{R}^{3\times 100}$.

To accommodate (i), we advocate combining convolutional or recurrent neural network with a base kernel to create a deep kernel [12,16], i.e., we design the kernel as

$$k_f(\cdot,\cdot) = k'_f\left(g_f(\cdot),\ g_f(\cdot)\right), \tag{8}$$

with $k'_f(\cdot,\cdot)$ a base (radial basis function, Matern) kernel, and $g_f(\cdot)$ a function parameterized by neural networks that maps the high dimensional vector $\mathbf{u}_n$ to a small vector $g_f(\mathbf{u}_n)$. The function $g_f(\cdot)$ produces thus intermediate low-dimensional inputs that are then integrated into the base kernel $k'_f(\cdot,\cdot)$. Beyond decreasing the evaluation execution time, the interest of the method is to obtain flexible kernels able to explain outliers and more accurate estimates [17].

Regarding (ii), we use approximate kernel learning based on inducing points, where m inducing points (a.k.a. pseudo-inputs) encode an $O(m^2)$ approximation to the original Gram matrix $\mathbf{K}(\mathbf{U},\mathbf{U})$. The approximate model has $O(dm^2)$ complexity for training and $O(m^3)$ complexity for evaluation on $\mathbf{u}'$, see (4). Resorting to stochastic variational inference [11,16] for optimization, and following [17], we jointly optimize over the hyperparameters $(\boldsymbol{\theta}, \tilde{\mathbf{\Sigma}})$, the m inducing pseudo-inputs $\bar{\mathbf{U}} = (\bar{\mathbf{u}}_1, \ldots, \bar{\mathbf{u}}_m)$, and the weights in the deep neural network $g_f(\cdot)$. In our experiments, we set $m = 100$ and have $g_f(\bar{\mathbf{u}}_i) \in \mathbb{R}^{20}$ yielding a scalable approach that allows handling large datasets.

Fig. 1. The considered datasets [18] (left) and [19] (right) contain data logs of two wheel encoders, a fiber optic gyro and a consumer IMU, recorded, respectively on a Segway and a consumer car.

E. Extension to the Observation Model

Suppose dynamical model (5) is partially observed through noisy non-linear observations

$$\mathbf{y}_n = h\left(\mathbf{x}_n,\ \mathbf{e}_n\right), \tag{9}$$

with $\mathbf{e}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{J}_n)$ a Gaussian measurement noise. To evaluate errors in the observation map as well we can model the observation map as a Gaussian process

$$h(\cdot) \sim \mathcal{GP}\left(m_h(\cdot),\ k_h(\cdot,\cdot)\right), \tag{10}$$

and corrections may be made along the lines of the preceding section. Note that, similar treatments based on kernels of the form (8) may be used for the measurement kernel $k_h(\cdot,\cdot)$ that thus combines a base kernel $k'_h(\cdot,\cdot)$ with neural networks $g_h(\cdot)$ if the number of measurements at instant n is large.

IV. Experimental Results

We evaluate the proposed approach on two publicly available datasets [18,19] containing data of two wheel encoders, a FoG gyro and an additional consumer IMU embarked on two different platforms, see Figure 1:

1) the *University of Michigan North Campus long-term dataset* [18], which contains 35 hours of data recorded on a *Segway* platform over the course of 15 months in 27 sequences. The vehicle repeatedly explores a campus, both indoors and outdoors, on varying trajectories;
2) the *complex urban LiDAR dataset* [19], that consists of data recorded on a consumer *car* which moves in complex urban environments, e.g. metropolitan areas, large building complexes and underground parking lots.

We extract from each dataset the sensor logs that we interpolate and synchronize when necessary at $1/\delta t = 100\,\text{Hz}$. We use the provided calibration parameters and transformation matrices to move from one sensor frame to another, and we consider the provided ground truth poses as a benchmark for training and evaluating our approach. The ground-truth is used for learning $SE(3)$ position and heading increment corrections inspired from [25]. We compute the ground truth speed from differentiation of the ground truth position to

Dataset	m-ATE translation (m)		m-ATE rotation (deg)	
	physical	**corrected**	physical	**corrected**
[18]	0.160	**0.091**	2.75	**1.78**
[19]	0.035	**0.024**	0.49	**0.24**

Fig. 2. Mean Absolute Trajectory Error (m-ATE) averaged on segments of duration 1 s on test data. "Physical" corresponds to standard wheel speed based model (2) with noise turned off, and "corrected" to its counterpart using method of Section III. The corrected model reduces the physical error residual both for translation and rotation errors for each dataset [18,19].

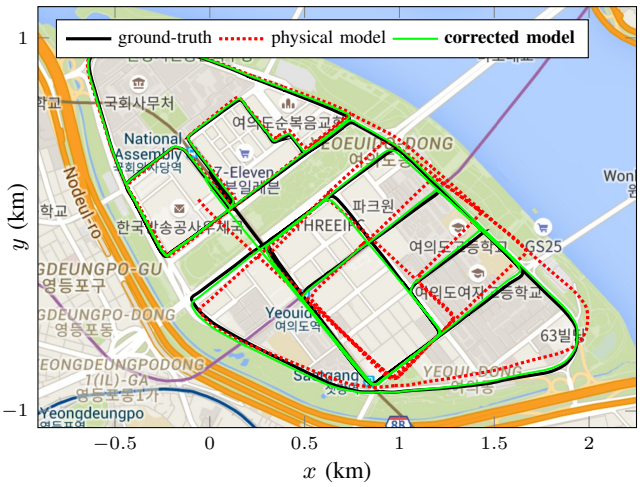

Fig. 3. Ground truth and estimated trajectories of a car from the physical and corrected models on the large 21 km test sequence `urban16` of dataset [19]. The corrected model improves its physical counterpart by clearly reducing the drift error of the estimates. Figure best seen in color.

compute the measurement residuals (13)-(14), see Section V. Each dataset is treated independently from the other, i.e. we have two distinct corrected models, one for each dataset.

To evaluate the performances of the models and the filters, we use the following error metrics [25]:

1) *mean Absolute Trajectory Error* (m-ATE), that averages the error with respect to a ground truth trajectory for a given sequence;
2) *cumulative Absolute Trajectory Error* (c-ATE), which sums the error m-ATE up to a given point in a trajectory. c-ATE can show clearer trends than m-ATE, because it is less affected by fortunate trajectory overlaps.

In the following, we call the "physical model" the standard wheel speed based model (2), and "corrected model" its counterpart using the proposed approach of Section III.

A. Parameter Setting & Training

For each dataset, we divide data into three sets: training data for optimizing the hyperparameters (70 %); cross-validation data to tune the prior physical model and optimizer parameters (15%); and test data that are never used in the learning process and allow assessing the performance of the approach (15%) [10]. Data is divided as follows: for [18], we use the first 19 sequences as training, the four following for cross-validation and the four last for testing; for [19], `urban14` and `urban17` is cross-validation data, `urban15-16` is test data, and the remaining sequences are training data[1]. We thus dispose of training set of size d respectively $1.1 \cdot 10^5$ and $2.8 \cdot 10^3$ for datasets [18] and [19].

Based on results on the cross-validation data, we set for the two datasets the same: model frequency rate, $1/\Delta t = 1\,\text{Hz}$; number of optimized inducing points, $m = 100$; kernel architecture; optimizer learning rate, 0.01; and number of epochs for training, 100. The kernel architecture consists in defining: the warping function $g_f(\cdot)$ as a two-layer 1D Convolutional Neural Networks (CNN) followed by a full layer that outputs a vector of dimension 20; and the kernel $k'_f(\cdot,\cdot)$ as Matern 5/2 kernels [where $k_f(\cdot,\cdot) = k'_f(g_f(\cdot), g_f(\cdot))$, see (8) in Section III]. In this setting we set the propagation input $\mathbf{u}_n \in \mathbb{R}^{3\times k}$ as a sequence of $k = 100$ measurements. We learn the parameters on normalized data using the Adam optimizer [32] on the deep probabilistic software Pyro[2]. Specifically, we build our code on the Gaussian Process "Contributed Code" of the Pyro development branch.

B. Posterior Model Evaluation

Once the model parameters are optimized, we compare the performances of the corrected models on test data as follow. We first analyze the decrease of the residuals by computing the m-ATE on segments of duration 1 s both for the physical and corrected models. Estimates from the corrected model are averaged over 100 samples, see (4). Results are given in Figure 2 and show a clear improvement both for the translation and rotation residuals, with minimal improvement 29 % and up to 51 % of the m-ATE rotation for car dataset [19].

We then compare the physical and corrected dynamical models for long term sequences by comparing trajectories in dataset [19], see Figure 3. For each test sequence, we run the dynamical model over the full sequence, where the state is initialized with ground truth.

We still sample 100 corrected model estimates and average the trajectory output to compute m-ATE and c-ATE, see Figures 4-5. Both errors are favorable to the corrected model, and particularly the c-ATE that clearly reveals a tread: the corrected models accumulate lower errors. Video attachment allows visualizing benefits of the approach, that essentially yield corrections to the model during turns. We also indicate that directly taking the output mean of the corrections instead of sampling and averaging leads to similar results.

V. Robot Dead Reckoning with Corrected Wheel Encoders and IMU Outputs

In this section, we utilise IMU measurements as complementary sensors for localization. As explained previously, $f(\cdot)$ may be corrected using GPs and a dataset D_f containing ground truth of the pose of the robot, wheel speeds and FoG measurements. Similarly, as explained in Section III-E, $h(\cdot)$ may be corrected using ground truth and IMU measurements in a dataset D_h.

[1]We use the sequences without IMU log for the propagation model alone.
[2]http://pyro.ai/

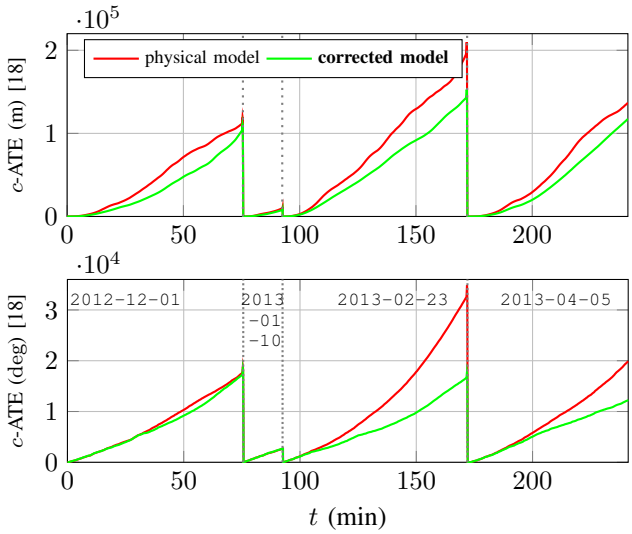

Fig. 4. Translation (above) and rotation (bottom) cumulative Absolute Trajectory Error (c-ATE) as function of time for physical and corrected models for four sequences of Segway dataset [18]. Errors are set to zero when starting a new sequence. The proposed corrected model accumulates lower error residual than the physical model.

A. Integrated IMU Measurement Model

To utilize the IMU measurements as complementary sensors in an EKF to update the state (1), we integrate the measurements on a sliding window, i.e., we consider the pre-integrated measurements of [33]. This leads to the observation model

$$\mathbf{y}_n = \left(\log_{SO(3)}\left(\Delta\mathbf{R}_n\right), \Delta\mathbf{v}_n, \Delta\mathbf{p}_n\right)^T, \tag{11}$$

where $\log_{SO(3)}(\cdot)$ is the logarithm map of the Lie group $SO(3)$, that is, $\log_{SO(3)}(\Delta\mathbf{R}) = \boldsymbol{\omega} \in \mathbb{R}^3$ is the angular velocity vector such that $\Delta\mathbf{R}$ is obtained by starting from the identity and turning around $\boldsymbol{\omega}$ during one unit of time, and $\exp_{SO(3)}(\cdot)$ is the inverse map of $\log_{SO(3)}(\cdot)$. In (11) we let

$$\Delta\mathbf{R}_n = \mathbf{R}_n^T\mathbf{R}_{n+1} = \prod_{i=0}^{r-1}\delta\mathbf{R}_i, \tag{12}$$

$$\Delta\mathbf{v}_n = \mathbf{R}_n^T\left(\mathbf{v}_{n+1} - \mathbf{v}_n - \mathbf{g}\Delta t\right) = \sum_{i=0}^{r-1}\delta\mathbf{v}_i, \tag{13}$$

$$\Delta\mathbf{p}_n = \mathbf{R}_n^T\left(\mathbf{p}_{n+1} - \mathbf{p}_n - \mathbf{v}_n\Delta t - \Delta t^2/2\right), \tag{14}$$

with $\mathbf{g}$ representing the gravity vector, $\mathbf{v}_n = (\dot{x}_n, \dot{y}_n, \dot{z}_n)$ is the robot velocity and $\mathbf{p}_n = (x_n, y_n, z_n)$ the robot position (by assuming planar motion we set $\dot{z}_n = 0$ [18]). We dispose of a sequence of r accelerations and angular velocities $(\mathbf{a}_i, \boldsymbol{\omega}_i)$, $i = 0, \ldots, r-1$ measured in the robot frame by the IMU such that the quantities $\delta\mathbf{R}_i = \prod_{j=0}^{i-1}\exp_{SO(3)}\left(\boldsymbol{\omega}_j\Delta t/r\right), (\delta\mathbf{R}_0 = \mathbf{I})$, $\delta\mathbf{v}_i = \sum_{j=0}^{i-1}\delta\mathbf{R}_j\mathbf{a}_j\Delta t/r$, and $\Delta\mathbf{p}_n = \sum_{i=0}^{r-1}\delta\mathbf{v}_i\Delta t/r + \delta\mathbf{R}_i\mathbf{a}_i\Delta t^2/(2r^2)$, and thus (11), are computed without the need for any state estimates.

B. Extended Kalman filter (EKF) Based on Corrected Models

Model (5) with observation (9) yield a system with state $\mathbf{x}_n$ and observation $\mathbf{y}_n$ of the form

$$\mathbf{x}_{n+1} = \mathbf{x}_n \boxplus f\left(\mathbf{u}_n,\ \mathbf{w}_n\right), \tag{15}$$

$$\mathbf{y}_n = h\left(\mathbf{x}_n,\ \mathbf{e}_n\right). \tag{16}$$

Using measurements $\mathbf{y}_n$ of (11), corrections may be learned for the observation model following the methodology of Section III-A, we may want to use the corrected model for real time (online) state estimation of system (15)-(16) when no ground truth is available, that is, on test data. To do so, we suggest to design an Extended Kalman Filter (EKF) using as model the mean of the posterior Gaussian process models, that is, $\bar{f}(\cdot) = \mathbb{E}\left[f(\cdot)|D_f\right]$, and $\bar{h}(\cdot) = \mathbb{E}\left[h(\cdot)|D_h\right]$. One could also use posterior covariance for EKF tuning, but this increases the execution time, and we decided to tune the corrected EKF with the propagation noise covariance matrix $\mathbf{Q}_n$ and measurement noise covariance matrix $\mathbf{J}_n$ that we have set for the uncorrected EKF.

The standard EKF methodology is based on Jacobian computation. Those used for propagating and updating the covariance matrix are computed by adding the Jacobian of the model (2)-(11) to the Jacobian of the corrections. The latter are computed through automatic differentiation in PyTorch [20].

C. Extended Kalman Filter Implementation

Our filter follows the state delayed EKF of [18]: it uses a differential-drive process model to integrate measurements from the wheel encoders, the FoG gyro are used for propagating heading, a constant velocity model for the roll and pitch is assumed, and we adapt the IMU measurements in [18] into the form of pre-integrated measurements [33].

We consider two EKFs. The first is a standard EKF for non-corrected dynamical system (5) with sub-increment (2) and observations (11). The second one is a corrected model based EKF using our correction methodology of Section III. We implement these error-state extended Kalman filters yielding state and error covariance estimates $\hat{\mathbf{x}}_{n|n}$, $\mathbf{P}_{n|n}$, where we use the small angle quaternion error [34] for representing the error on the Euler angles, enabling proper representation of the angle uncertainties. Since measurement $\mathbf{y}_n$ in (11) requires state estimates at time $n-1$ and n [see (12)-(14)], we use stochastic cloning [35] such that the matrix covariance error $\mathbf{P}_{n|n-1}$ contains the required error at both time $n-1$ and n as in, e.g., [5]. Finally, Jacobian required for updating the state are computed after [33], see equations (42)-(43) therein. The remaining Jacobian w.r.t. the corrections are computed by automatic differentiation [20].

D. Corrected Model Based EKF Evaluation

We now compare in this section the performances of the filters based on the corrected (expected) model and the filters based on the physical model on test data. We set the same noise standard-deviation parameters for both physical and corrected filters for each dataset as: $0.7\,\mathrm{m/s}$ for wheel velocities, $0.005\,\mathrm{deg/s}$ for FoG gyro, $1\,\mathrm{deg/s}$ for

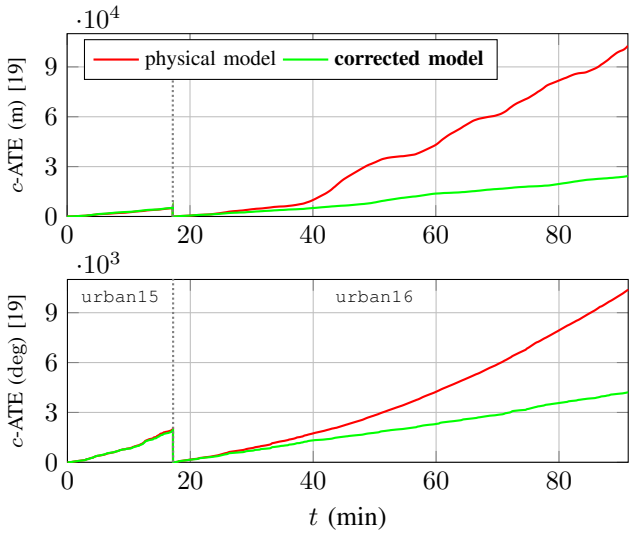

Fig. 5. Translation (above) and rotation (bottom) cumulative Absolute Trajectory Error (c-ATE) as function of time for physical and corrected models for sequences `urban15-16` of the car dataset [19]. Errors are set to zero when starting a new sequence. Plots show that the proposed corrected model accumulates lower error.

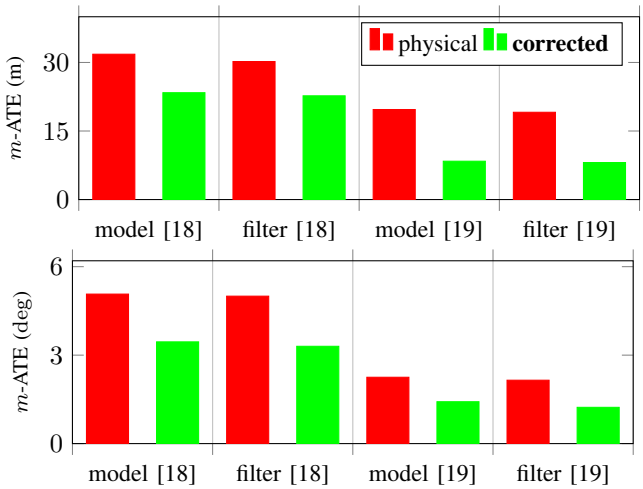

Fig. 6. The translation and rotation mean Absolute Trajectory Error (m-ATE) computed between position estimates and ground truth on the datasets [18,19]. The proposed corrected models and filters (EKF) clearly outperform their physical counterpart for each dataset.

angular rate, $1\,\mathrm{deg}/\mathrm{s}^2$ for angular acceleration, and $2\,\mathrm{deg}/\mathrm{s}$, $0.05\,\mathrm{m}/\mathrm{s}^2$ for the IMU noise standard deviation. Results are displayed on Figure 6. An example of the trajectory estimates in dataset [18] is displayed in Figure 7. The c-ATE is displayed over time on Figure 8. The results show a very clear improvement over the use of physical model (2)-(11) both for wheel odometry based dead-reckoning, and for EKF-based navigation using additional IMU data.

VI. Conclusion

This paper introduces a methodology for Gaussian processes combined with neural networks and stochastic variational inference to improve both the propagation and measurement functions of a state-space dynamical model by

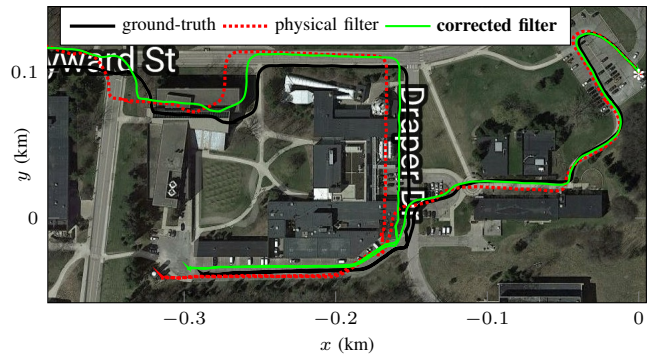

Fig. 7. Ground truth and estimated trajectories of the Segway, from the physical and corrected filters on the validation sequence `2013-01-10` of dataset [18]. The corrected filter improves its physical counterpart both for translation and rotation rotation mean Absolute Trajectory Error (m-ATE).

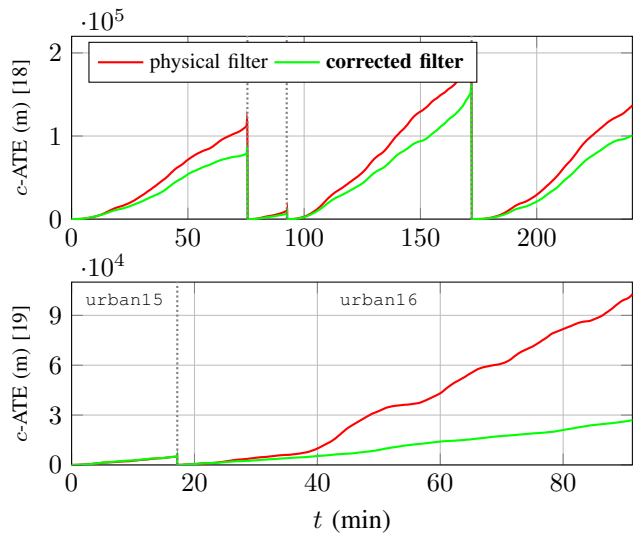

Fig. 8. Translation cumulative Absolute Trajectory Error (c-ATE) as function of time for physical and corrected filters for datasets [18,19]. Errors are set to zero when starting a new sequence. The proposed corrected filters obtains better results than the physical filter. Trend are similar for the rotational error.

learning error residuals between physical prediction and ground truth data. These corrections are also shown to be usable for EKF design. The applications on publicly available datasets for consumer car and Segway navigation systems based on wheel encoders, gyro and IMU clearly reveals the gains of performances of the proposed approach.

Future work includes the evaluation and generalization of what the algorithm is actually learning. This concerns testing the training Gaussian processes on different datasets. Another works concerns on-line learning with standard devices and inaccurate, but many, ground truth data, since the proposed method are independent w.r.t. data size once training is done.

Acknowledgment

We thank the authors of [18] and especially Arash Ushani for sharing their wheel encoder data log.

REFERENCES

[1] J. Hidalgo-Carrió, D. Hennes, J. Schwendner, *et al.*, "Gaussian Process Estimation of Odometry Errors for Localization and Mapping," in *International Conference on Robotics and Automation*, IEEE, 2017, pp. 5696–5701.

[2] R. Jazar, *Vehicle Dynamics: Theory and Application*, ser. Springer. 2017.

[3] R. Wang and J. Wang, "Fault-Tolerant Control With Active Fault Diagnosis for Four-Wheel Independently Driven Electric Ground Vehicles," *Transactions on Vehicular Technology*, IEEE, vol. 60, no. 9, pp. 4276–4287, 2011.

[4] S. Leutenegger, S. Lynen, M. Bosse, *et al.*, "Keyframe-based Visual-Inertial Odometry using Nonlinear Optimization," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2015.

[5] K. Sun, K. Mohta, B. Pfrommer, *et al.*, "Robust Stereo Visual Inertial Odometry for Fast Autonomous Flight," *Robotics and Automation Letters*, IEEE, vol. 3, no. 2, pp. 965–972, 2018.

[6] R. Mur-Artal and J. D. Tardos, "Visual-Inertial Monocular SLAM With Map Reuse," *Robotics and Automation Letters*, IEEE, vol. 2, no. 2, pp. 796–803, 2017.

[7] M. Quan, S. Piao, M. Tan, *et al.*, "Tightly-Coupled Monocular Visual-Odometric SLAM Using Wheels and a MEMS Gyroscope," 2018.

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

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

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

[11] D. Blei, A. Kucukelbir, and J. McAuliffe, "Variational Inference: A Review for Statisticians," *Journal of the American Statistical Association*, Taylor & Francis, vol. 112, no. 518, pp. 859–877, 2017.

[12] A. Damianou and N. Lawrence, "Deep Gaussian Processes," AISTATS, 2013.

[13] J. Ko, D. J. Klein, D. Fox, *et al.*, "Gaussian Processes and Reinforcement Learning for Identification and Control of an Autonomous Blimp," in *International Conference on Robotics and Automation*, IEEE, 2007, pp. 742–747.

[14] J. Ko, D. Kleint, D. Fox, *et al.*, "GP-UKF: Unscented Kalman Filters with Gaussian Process Prediction and Observation Models," in *International Conference on Intelligent Robots and Systems*, IEEE, 2007, pp. 1901–1907.

[15] M. Hoffman, "Stochastic Variational Inference," *Journal of Machine Learning Research*, vol. 14, no. 1, pp. 1303–1347, 2013.

[16] A. Wilson, Z. Hu, R. Salakhutdinov, *et al.*, "Stochastic Variational Deep Kernel Learning," in *Advances in Neural Information Processing Systems*, 2016, pp. 2586–2594.

[17] A. Wilson, Z. Hu, R. Salakhutdinov, *et al.*, "Deep Kernel Learning," in *International Conference on Artificial Intelligence and Statistics*, Curran Associates, Inc., 2016, pp. 370–378.

[18] N. Carlevaris-Bianco, A. Ushani, and R. Eustice, "The University of Michigan North Campus Long-Term Vision and Lidar Dataset," *The International Journal of Robotics Research*, vol. 35, no. 9, pp. 1023–1035, 2015.

[19] J. Jeong, Y. Cho, Y.-S. Shin, *et al.*, "Complex Urban LiDAR Data Set," in *International Conference on Robotics and Automation*, IEEE, 2018.

[20] A. Paszke, S. Gross, S. Chintala, *et al.*, "Automatic Differentiation in PyTorch," in *Advances in Neural Information Processing Systems*, 2017.

[21] C. E. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*, ser. Adaptive computation and machine learning. Cambridge, Mass: MIT Press, 2006.

[22] C. Williams, S. Klanke, S. Vijayakumar, *et al.*, "Multi-task Gaussian Process Learning of Robot Inverse Dynamics," in *Advances in Neural Information Processing Systems*, 2009, pp. 265–272.

[23] T. Tang, D. Yoon, F. Pomerleau, *et al.*, "Learning a Bias Correction for Lidar-Only Motion Estimation," 2018.

[24] J. Toledo, J. Piñeiro, R. Arnay, *et al.*, "Improving Odometric Accuracy for an Autonomous Electric Cart," *Sensors*, vol. 18, no. 2, 2018.

[25] V. Peretroukhin and J. Kelly, "DPC-Net: Deep Pose Correction for Visual Localization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2424–2431, 2018.

[26] 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.

[27] S. Wang, R. Clark, H. Wen, *et al.*, "End-to-End, Sequence-to-Sequence Probabilistic Visual Odometry through Deep Neural Networks," *The International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 513–542, 2017.

[28] I. Belhajem, Y. Ben Maissa, and A. Tamtaoui, "Improving low cost sensor based vehicle positioning with Machine Learning," *Control Engineering Practice*, vol. 74, pp. 168–176, 2018.

[29] A. Punjani and P. Abbeel, "Deep Learning Helicopter Dynamics Models," in *International Conference on Robotics and Automation*, IEEE, 2015.

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

[31] M. A. Álvarez, L. Rosasco, and N. D. Lawrence, "Kernels for Vector-Valued Functions: A Review," *Foundations and Trends® in Machine Learning*, vol. 4, no. 3, pp. 195–266, 2012.

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

[33] C. Forster, L. Carlone, F. Dellaert, *et al.*, "On-Manifold Preintegration for Real-Time Visual-Inertial Odometry," *Transactions on Robotics*, IEEE, vol. 33, no. 1, pp. 1–21, 2017.

[34] J. Solà, "Quaternion Kinematics for the Error-State Kalman Filter," 2015.

[35] S. Roumeliotis and J. Burdick, "Stochastic Cloning: A Generalized Framework for Processing Relative State Measurements," in *International Conference on Robotics and Automation*, vol. 2, IEEE, 2002, pp. 1788–1795.