Learning Observation Model for Factor Graph Based-State Estimation Using Intrinsic Sensors

Dinh Van Nam and Kim Gon-Woo, *Member, IEEE*

***Abstract*— The navigation system of autonomous mobile robots has appeared challenging when using exteroceptive sensors such as cameras, LiDARs, and radars in textureless and structureless environments. This paper presents a robust state estimation system for holonomic mobile robots using intrinsic sensors based on adaptive factor graph optimization in the degradation scenarios. In particular, the neural networks are employed to learn the observation and noise model using only IMU sensor and wheel encoder data. Investigating the learning model for the holonomic mobile robot is discussed with various neural network architectures. We also explore the neural networks that are far more powerful and have cheaper computing power when using the inertial-wheel encoder sensors. Furthermore, we employ an industrial holonomic robot platform equipped with multiple LiDARs, cameras, IMU, and wheel encoders to conduct the experiments and create the ground truth without a bulky motion capture system. The collected datasets are then implemented to train the neural networks. Finally, the experimental evaluation presents that our solution provides better accuracy and real-time performance than other solutions.**

***Note to Practitioners*—Autonomous mobile robots need to serve in challenging environments robustly that deny extrinsic sensors such as cameras, LiDARs, and radars. In order to operate in this situation, the navigation system shall rely on the intrinsic sensor as inertial sensor and wheel encoders. Existing conventional methods have combined the intrinsic sensors in the form of the recursive Bayesian filtering technique without adapting these models. Besides, deep learning-based solutions have adopted extensive networks like LSTM or CNN to handle the estimation problem. This work aims to develop a state estimation subsystem of the navigation system for the holonomic mobile robots that utilize the intrinsic sensors in adaptive factor graph optimization. In particular, we present how to join the factor graph efficiently with the learning observation model of IMU and wheel encoder factor. Moreover, the neural networks are introduced to learn the observation model with an IMU and wheel encoder data inputs. We recognize that lightweight neural networks can achieve more power than deep learning techniques using the IMU sensor and wheel encoders. Finally, the neural networks are embedded in a factor graph to handle the smoothing state estimation. The proposed system could operate with high accuracy in real time.**

***Index Terms*— State estimation, deep learning based-inertial navigation system, sensor fusion, factor graph optimization.**

Manuscript received 3 June 2022; accepted 18 July 2022. This article was recommended for publication by Associate Editor A. Pietrabissa and Editor C. Seatzu upon evaluation of the reviewers' comments. This work was supported in part by the Ministry of Science and ICT (MSIT), South Korea, under the Grand Information Technology Research Center Support Program Supervised by the Institute for Information and Communications Technology Planning and Evaluation (IITP) under Grant IITP-2021-2020-0-01462, and in part by the Korea Institute for Advancement of Technology (KIAT) Grant through the Korean Government (MOTIE) (HRD Program for Industrial Innovation) under Grant P0020536. *(Corresponding author: Kim Gon-Woo.)*

Dinh Van Nam is with the Intelligent Robotics Laboratory, Department of Intelligent Systems and Robotics, Chungbuk National University, Cheongju 28644, South Korea, and also with the Faculty of School of Engineering and Technology, Vinh University, Vinh, Nghe An 43100, Vietnam (e-mail: namdv@vinhuni.edu.vn).

Kim Gon-Woo is with the Intelligent Robotics Laboratory, Department of Intelligent Systems and Robotics, Chungbuk National University, Cheongju 28644, South Korea (e-mail: gwkim@cbnu.ac.kr).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TASE.2022.3193411.

Digital Object Identifier 10.1109/TASE.2022.3193411

I. Introduction

OVER the years, the increased maturity of autonomous mobile robotics (AMR) has become commercially viable with various industrial applications, such as automated guided vehicles [1], underground autonomous robots [2], and self-driving cars [3]. Positioning plays a fundamental skill that enables AMRs to be reliable [3], [4]. The estimated position is then used for the motion planning and control systems of the autonomous systems [3], [5]. The most advantaged navigation techniques were shown in the DARPA Subterranean (SubT) Challenge [6], which forced robotic researchers to create state-of-the-art technologies to assist operations in complex underground environments, illustrating significant challenges for military and civilians in disaster response [7]. Roboticists employed many complex navigation techniques in exploring and mapping unknown underground sites for the teams of robots in the SubT Challenge [7]. Nevertheless, current state estimation technologies are not flexible and robust sufficiently for what the industry requires in various indoor and outdoor environments. We want to overcome the limitations by developing a multi-sensor fusion system to navigate a holonomic mobile robot using neural networks. This work focuses on implementing a sub-estimator for the whole navigation system using only intrinsic sensors in case of extrinsic sensors failure.

In general, AMRs are equipped with proprioceptive sensors such as inertial sensors and wheel encoders working at high frequency (>100 Hz) for the control systems and short-term navigation [8], [9]. The GPS is the critical sensor for long-term positioning [3]. Nevertheless, GPS accuracy is significantly degraded in such environments as indoor parking lots, under high buildings, or tunnels [2]. Especially, GPS is denied in indoor applications like industrial factories or subterranean environments [10], [11]. Therefore, exteroceptive sensors such as LiDAR, camera, and radar are deployed to estimate the robot state in the GPS-denied scenarios [10]. For

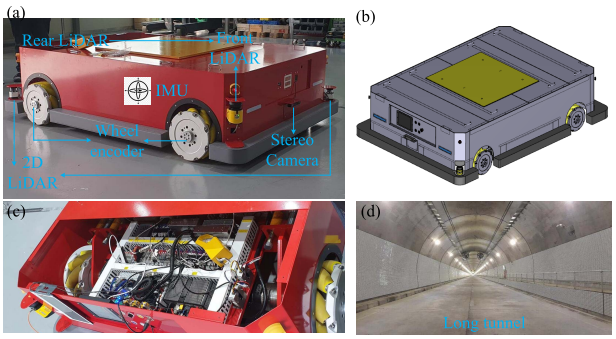

Fig. 1. (**a**) Overview of holonomic robot platform consists of 04 LiDAR, two Zed 2 cameras, wheel encoders, and IMU. (**b**) Mechanical design of the robot. (**c**) Electrical and electronic inside the robot. (**d**) Example of the challenge structureless and textureless environment.

instance, the camera can observe the invariance landmarks in its surroundings to update the pose estimation after long-term navigation [12]. However, the cameras are sensitive to light conditions such as illumination and darkness conditions [13], [14]. The cameras can fail to detect the visual features in textureless scenarios [2], [15]. Furthermore, the radars and LiDARs are continuously challenging to leverage the unstructured environments like free spaces or long corridors [2], [6]. In contrast, the inertial sensors and wheel encoders, which are immune to environmental conditions, can work well in short-term estimation [2], [10]. Accordingly, a fusion of extrinsic and intrinsic sensors can provide a high-performance estimator in GPS-denied conditions [6], [16].

In robotics, the fundamental approach of multi-sensor fusion techniques has commonly relied on the Bayesian filtering (BF) technique [17]. In particular, the BF predicts the state probability with the process model, which is then corrected using the observation model [17], [18]. The fusion schemes can be broadly categorized into loosely coupled and tightly coupled [13], [14]. Although the tightly coupled design can provide more accuracy, the loosely coupled is more straightforward in handling the sensor failure [2], [10]. By contrast, recent works based on deep learning techniques exemplify an alternative strategy to the BF solution [8]. Instead of using Bayes inference, these deep learning solutions replace the prediction and correction stage using just an end-to-end process [19]. We aim to develop an accurate estimation subsystem for a holonomic mobile robot in the lack of visual and structural features, as shown in Fig. 1-d. Here, the robot navigation system could handle the cases when both camera and LiDAR sensors fail to detect the features. The robot shall rely on intrinsic sensors such as wheel encoders and inertial sensors. Nevertheless, the estimated state will inevitably drift over time if not provided with global positioning information. Besides, the use of inertial sensors is quite challenging because of unobservable motions, including x, y, z directions and yaw rotation [13].

In this study, we present a tightly coupled fusion system based on factor graph optimization leveraging the learning technique for the holonomic mobile robot to enhance the estimation performance. The proposed system provides two major advantages. First, the design inherits the Bayesian filtering strategy, which provides a high-performance estimator solution. Second, a lightweight neural network is employed to predict the observation and noise models instead of the heavy deep learning structures. We shall show several surprising properties of neural network architectures for holonomic mobile robots with intrinsic sensors.

The remainder of this paper is organized as follows: Section II reviews the literature and re-categorizes multi-sensor fusion methods via the Bayesian filtering and Artificial intelligence (AI) techniques. Next, section III presents the problem formulation with the mathematics background. Section IV then provides the novel factor graph of the proposed formulation. The detailed implementation and evaluation of the proposed method are presented in section V, followed by a conclusion in section VI.

II. Related Works

This section provides a concise review and classifies the state-of-the-art multi-sensor fusion techniques. In general, extrinsic sensors such as GPS, LiDARs, radars, and cameras and intrinsic sensors like IMUs and wheel encoders are equipped for AMRs [4], [5]. The Bayesian inference is regularly utilized to handle all the sensor information to maximize the state posterior probability [17].

The sensor fusion techniques can be divided into three broad classes: filtering-based, optimization-based, and end-to-end solution [5], [13], [14]. While researchers have formed estimation technologies using extended Kalman filter (EKF) since the 1970s, optimization-based methods have only operated and flourished over the last two decades [17]. The filtering method commonly manipulates the EKF, unscented KF (UKF), and particle filter operating with prediction and correction phases. One of the most high-performance navigation systems is the multi-state constraints EKF, which performs the tightly coupled method for the real-time visual-inertial navigation system [20], [21].

By contrast, optimization-based solutions leverage historical states and sensor information for better accuracy and easy integration of the loop closure detection in long-term navigation [5]. VINS-Mono [22] is a pioneer open-source visual-inertial simultaneous localization and mapping (SLAM) leveraging factor graph optimization (FGO) [18], [23]. LILI-OM [24] and LIO-SAM [25] have been introduced as the state-of-the-art tightly coupled methods of LiDAR-inertial SLAM since 2020. LVI-SAM [26] has lately been proposed for the tightly coupled fusion of 3D LiDAR, camera, and IMU via smoothing estimation.

Recently, more powerful artificial intelligence (AI) techniques have been available, leading to rapidly developing end-to-end learning for the third category state estimation. Overall, AI-based multi-sensor fusion is divided into three primary strategies in which the AI can be employed to learn the sensor noise model, the relative motion model, and the end-to-end solution [11], [19]. The classification overview is concisely represented, as shown in Fig. 2. The AI approach can assist the BF in predicting the observation model or sensor noise model [19]. In particular, AI can be used to predict the relative

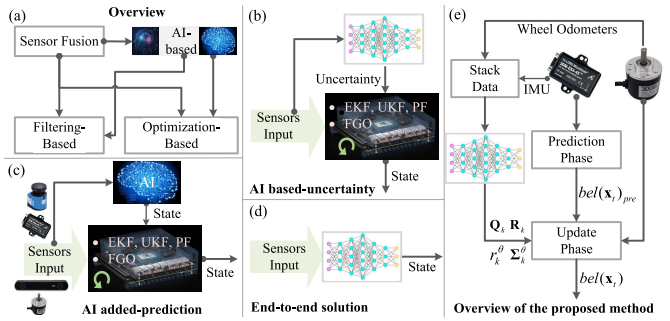

Fig. 2. Overview of the multi-sensor fusion technique. (**a**) Presents the overall multi-sensor fusion divided into three classes; (**b**) AI is used to learn the sensor noise model; (**c**) AI-based relative state model; (d) the end-to-end solution using AI; (**e**) Briefly describes our technique which applied AI to learn both the sensor noise and relative state model.

robot motion between two consecutive poses immediately [8], [19], [27]. Furthermore, learning the sensor noise model can be integrated into the BF process [28], [29].

AI-IMU dead-reckoning [28] and the tight learned inertial odometry (TLIO) [30] are the critical approaches that use the convolution of neural networks (CNNs) to predict the noise model in the correction phase of the EKF. Moreover, the relative state motion is computed to update the belief states of BF using AI directly. For instance, RINS-W proposed an EKF based on a long short-term memory (LSTM) network for motion profile detector on Manifold [31]. The differentiable filters [19], [32] employed deep learning to the relative motion, measurement, or uncertainty models. Backprop KF [33] is a pioneer solution combining the EKF and CNN model for visual odometry and tracking on the KITTI dataset. In 2018, Jonschkowski *et al.* proposed the particle filter with a learning model for state estimation [34]. By contrast, the state estimation can be directly estimated by AI without using the BF technique. The robust neural inertial navigation (RoNIN) [9] and inertial odometry neural network (IONet) [8] performed the recurrent neural networks to predict the relative state motion. An extended method applied LSTM for the neural visual-inertial odometry [35]. Table I describes a summary of the AI-based state estimation problem. The performance of each solution is evaluated by remarked with $\star$ (*star*) score. In general, the end-to-end method is straightforward to implement without the Bayes inference, which needs to improve accuracy. In contrast, the adapting uncertainty model using deep learning in the Bayes filtering technique to correct the uncertainty model provides better results. Moreover, Bayes inference with deep learning in differentiable filtering promises good performance. We implement the AI-based solution to learn the observation and noise models based on the wheel-inertial sensor data, as briefly presented in Fig. 2-e.

We highlight our contributions as follows:

- A link between factor graph optimization and the neural network with a novel learnable model enables us to fuse inertial sensors and wheel encoders efficiently.
- Using IMU sensor and wheel encoder data, simple but powerful neural network architectures are proposed for the observation and noise models. In particular, we compare the neural network architectures using different training methods, including the LSTM model, deep neural network, and multilayer perceptron MLP) network.
- We implement an accurate and robust real-time estimator system to conduct the experimental evaluation with a holonomic robot platform.

TABLE I
OVERVIEW OF THE AI-BASED STATE ESTIMATION

AI-based State Estimation			
Name	*Techniques*	*AI Model*	*Perform*
AI Uncertainty Adaptive	1. AI-IMU [28] 2. TLIO [30] 3. RINS-W [31]	1. CNN 2. CNN 3. LSTM	$\star\star\star\star$
Estimation Differentiable	1. Backprop KF [33] 2. Differentiable filters [19]	1. CNN 2. CNN 2. LSTM	$\star\star\star$ $\star\star\star\star$
AI-based End-to-End	1. RoNIN [9] 2. IONet [8] 3. LSTM [19]	1. LSTM 1. ResNet 2. LSTM 3. LSTM	$\star\star$

III. PROBLEM STATEMENT

An autonomous navigation system is generally designed with three parts: state estimation and mapping, motion planning, and control system [3], [6], as shown in Fig. 3. In this section, we describe how to embed the neural networks into the Bayesian filter for state estimation using the inertial-wheel encoder setup. The complete sensor setup of the holonomic robot is equipped with stereo cameras, LiDARs, IMU, and wheel encoders. We aim to estimate the robot position over time using only the IMU sensor and wheel encoders in the degradation scenarios of LiDARs and cameras.

A. Notations

In this article, we represent matrices by uppercase Roman bold ($\mathbf{A}, \mathbf{B}, \mathbf{C} \ldots$), vectors are lowercase Roman bold ($\mathbf{a}, \mathbf{b}, \mathbf{c} \ldots$), and scalars are donated by lowercase italics ($a, b, c, \ldots$). ${}^{a}\mathbf{X}_{b}$ represents a transformation $\mathbf{X}$ from frame b to frame a. The squared Mahalanobis distance of $\mathbf{X}$ with covariance matrix Σ is $\|\mathbf{X}\|^2_{\Sigma} = \mathbf{X}^{\mathrm{T}}\Sigma^{-1}\mathbf{X}$. Frame b denotes the robot's body coordinate that matches the IMU frame i, and w represents a fixed coordinate to earth. Let $\mathcal{M}$ and $\mathcal{T}_{\mathcal{M}}$ be n-manifold and its tangent space, respectively. For the robotics field, manifold $\mathcal{M}$ is usually considered as a Lie group, and the operator ("boxplus") $\boxplus$ is defined to compute the action $\mathbf{u}$ on tangent space to a Lie group $\mathbf{a}$, and operator ("boxminus") $\boxminus$ is denoted as a differ of Lie group $\mathbf{a}$ and $\mathbf{b}$ on Euclidean space as follows,

$$\boxplus : \mathcal{M} \times \mathbb{R}^n \rightarrow \mathcal{M} \Rightarrow \mathbf{a} \boxplus \mathbf{u} = \mathbf{a} \circ \mathrm{Exp}(\mathbf{u}) \quad (1)$$

$$\boxminus : \mathcal{M} \times \mathcal{M} \rightarrow \mathbb{R}^n \Rightarrow \mathbf{b} \boxminus \mathbf{a} = \mathrm{Log}(\mathbf{a}^{-1} \circ \mathbf{b}), \quad (2)$$

where the Exp and Log are capitalized exponential and logarithm map, and $\circ$ is composite operation [36]. The mapping between $\mathcal{M}$ and $\mathcal{T}_{\mathcal{M}}$ is also computed by using the capitalized Exp and Log explained in [36]. $\mathbb{SE}(3)$ denotes the Special Euclidean Group, and $\mathbb{SO}(3)$ is the Special Orthogonal Group

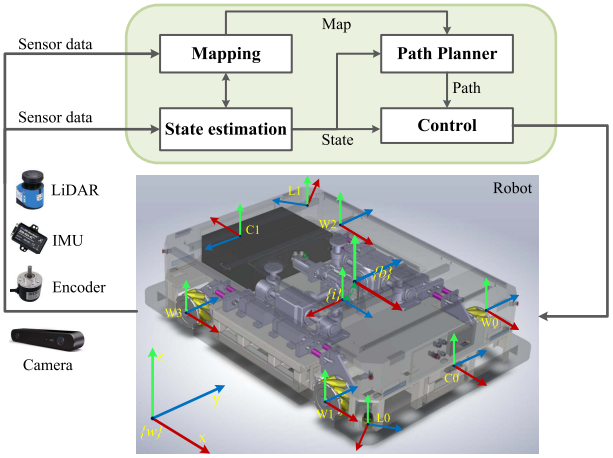

Fig. 3. Autonomous system overview and the sensors coordinate system of the holonomic robot. The world coordinate w is fixed to the earth. Frame i is the IMU frame. b denotes the body frame. W0, W1, W2, W3 are the coordinate of the front left, front right, rear left, and rear right wheel, respectively. C0 and C1 are the front and rear camera frames, respectively. The IMU frame i has coincided with body frame b.

in 3D space [36]. $\mathfrak{se}(3)$ and $\mathfrak{so}(3)$ are the Lie-algebra of $\mathbb{SE}(3)$ and $\mathbb{SO}(3)$, respectively. Similarly, $\mathbb{SE}(2)$ and $\mathbb{SO}(2)$ are denoted on the 2D planar. The skew-symmetric matrix of vector $\boldsymbol{\theta}$ is represented as $[\boldsymbol{\theta}]_\times$. Visit [17], [36] for more detail. $\mathbf{I}_n$ is an n-identity matrix. $\mathbf{e}_1$, $\mathbf{e}_2$ and $\mathbf{e}_3$ represent the 1, 2, and 3 column vectors of $\mathbf{I}_3$, respectively.

B. Problem Formulation

Let us provide the transformation of each sensor frame to the robot center, as shown in Fig. 3. In particular, an IMU is located at frame i, and four wheel encoders are fixed at frames W0, W1, W2, W3. Frame b is the body frame of the robot that coincides with frame i. We assume that the intrinsic and extrinsic calibration parameters of all sensors are computed precisely.

Let us define a robot state at time t_i as follows,

$$\mathbf{x}_i \triangleq [\mathbf{R}_i,\ \mathbf{p}_i,\ \mathbf{v}_i,\ \mathbf{b}_i^a,\ \mathbf{b}_i^\omega] \in \mathbb{SO}(3) \times \mathbb{R}^{12}, \tag{3}$$

where $\mathbf{R}_i \in \mathbb{SO}(3)$, $\mathbf{p}_i,\ \mathbf{v}_i \in \mathbb{R}^3$ are the robot orientation, position and velocity with respect to world frame w, respectively; $\mathbf{b}_i^a,\ \mathbf{b}_i^\omega \in \mathbb{R}^3$ are the IMU bias of accelerometer and gyroscope [13], respectively. A set of robot state up to time step t_k is denoted as,

$$\chi_k \triangleq \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_i, \ldots\}_{i\in\Omega}, \tag{4}$$

where Ω is the sets of robot keyframes up to t_k. The robot utilizes just IMU sensor and wheel encoders, so the entire set of observing sensors data is given as,

$$\mathcal{Z}_k \triangleq \{\mathcal{I}_i,\ \mathcal{W}_i\}_{i\in\mathrm{K}}, \tag{5}$$

where $\mathcal{I}_i$ and $\mathcal{W}_i$ represent the collection data of IMU measurements and wheel encoders up to time t_k, respectively; K is a set of time steps at the observed sensor data. The robot pose at time t_i is predicted by using the kinematic motion model with IMU measurement as [20],

$$\frac{\partial}{\partial t}\mathbf{x}_i = f(\mathbf{x}_i, t), \tag{6}$$

where $f(.)$ is a time-varying vector field function [20].

The objective is to maximize the likelihood of the sensor measurements $\mathcal{Z}_k$ given a set of robot states χ_k. By using the maximum a posterior probability (MAP) inference [17], the problem is turned to,

$$\chi_k^* = \arg\max_{\chi_k} p(\chi_k|\mathcal{Z}_k, \mathcal{H}_0) \propto p(\mathcal{H}_0)p(\mathcal{Z}_k|\chi_k), \tag{7}$$

where $\mathcal{H}_0$ is the set of prior state information computed by the motion model and historical robot state. Given that the Gaussian white noise models all states and measurement noises, the measurement data is also independent. The MAP problem can be transformed to a nonlinear least-square optimization as follows [18],

$$\chi_k^* = \arg\min_{\chi_k}\Big(\|\mathbf{r}_0\|^2_{\Sigma_0} + \sum\|\mathbf{r}_\rho\|^2_{\Sigma_\rho} + \sum\|\mathbf{r}_z\|^2_{\Sigma_z}\Big), \tag{8}$$

where $\mathbf{r_0}$ is the initial pose constraint; $\mathbf{r}_\rho$ denotes the prior residual related to the motion model and historical states; $\mathbf{r}_z$ is the residual constructed by sensor information; $\Sigma_{(.)}$ is the covariance matrices corresponding to its residual. Note that if all the history information is maintained, the size of the optimization problem (8) shall be enormous. We cannot employ this large-scale optimization problem for real-time applications. Therefore, the number of variables is limited to a constant value to reduce the computational cost. To this end, at the beginning of the sliding window, a marginalization constraint of the previous window joins the optimization problem [22]. In particular, in the case of using the inertial and wheel encoders measurement, we indicate the total residual as minimizing the following cost function as follows,

$$\mathbf{r}_M + \sum_{i\in S}\Big(\|\mathbf{r}_{\mathcal{I}_i}\|^2_{\Sigma_{\mathcal{I}_i}} + \|\mathbf{r}_{b_i}\|^2_{\Sigma_{b_i}} + \|\mathbf{r}_{\mathcal{W}_i}\|^2_{\Sigma_{\mathcal{W}_i}} + \|\mathbf{r}_{\psi_i}\|^2_{\Sigma_{\psi_i}}\Big), \tag{9}$$

where $\mathbf{r}_M$ is the residual marginalization [22] that is the residual of the initial pose in the window $\mathcal{S}$; $\mathbf{r}_{\mathcal{I}_i}$ and $\mathbf{r}_{b_i}$ is the IMU preintegration and bias factor between two consecutive keyframes, respectively; $\mathbf{r}_{\mathcal{W}_i}$ is the wheel encoders constraint between two consecutive keyframes; $\mathbf{r}_{\psi_i}$ is the inference constraints including relative velocity, zero velocity, and neural network factor.

Here, the multi-objective non-linear least squares optimization (9) called multi-criterion problem with covariance matrix $\Sigma_{(.)}$ indicated by the weighted sum factor [37]. The covariance matrix $\Sigma_{(.)}$ denotes the optimal trade-off for each factor. The non-linear least square optimization problem (9) is represented as a factor graph, as shown in Fig. 4. All sensor information is observed to compute the factor on the graph and predict the relative motion and noise model using the supervised learning techniques [38]. The detail of this method shall be described in Section IV.

Remark 1: The value of covariance matrix $\Sigma_{(.)}$ significantly influences the optimal solution. A smaller covariance matrix means it has more influence on the optimal answer. Note that $\mathbf{r}_{\psi_i}$ or covariance matrices $\Sigma_{(.)}$ (9) can be directly learned

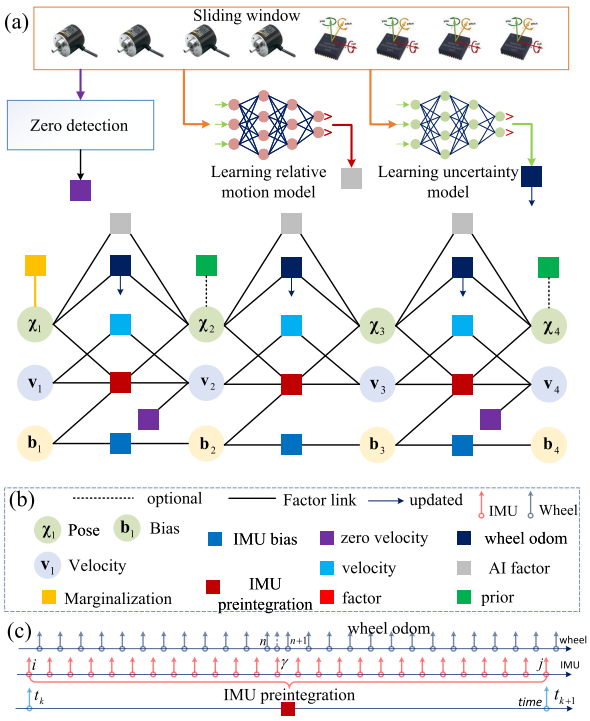

Fig. 4. Overview of the AI-based adaptive factor graph optimization (9). (**a**)-the illustration of a factor graph in which the zero-velocity detection and two neural network based-learning sensor models are calculated within a sliding window. (**b**)- the type of factors and the link between them, where their detailed description is presented in Section 2. (**c**)- the depiction of the wheel encoder clock and IMU sensor clock, where each sensor operates at a different frequency.

using the AI approaches. We can find the influence of covariance on the optimal solution in [37] and [29]. Furthermore, studying convergence properties on the manifold is presented in [17].

IV. Learning-Based Factor Graph Optimization

As we discussed in Section III. In this section, we will explain how to establish the non-linear least-squares optimization (9) as well as the factor graph as shown in Fig. 4. To conduct solving the optimization problem (9), we need to define the IMU preintegration factor, wheel odometer factor, and neural network factor.

A. Preintegration IMU Factors

In general, an IMU sensor can directly provide 6 DoF measurements [13] that are corrupted by noises and biases given as,

$$ {}^{b}\tilde{\boldsymbol{\omega}} = {}^{b}\boldsymbol{\omega} + \mathbf{b}^{g} + \boldsymbol{\eta}_{g} \tag{10} $$

$$ {}^{b}\tilde{\mathbf{a}} = {}^{b}\mathbf{R}_{w}({}^{w}\mathbf{a}_{b} - \mathbf{g}) + \mathbf{b}^{a} + \boldsymbol{\eta}_{a}, \tag{11} $$

where ${}^{b}\tilde{\boldsymbol{\omega}}, {}^{b}\tilde{\mathbf{a}} \in \mathbb{R}^3$ are angular velocity and acceleration vector to body coordinate b, respectively; $\mathbf{b}^{g}, \mathbf{b}^{a} \in \mathbb{R}^3$ are quasi-constant biases, $\boldsymbol{\eta}_{g} \overset{\Delta}{=} \mathrm{N}(\mathbf{0}, \sigma_{g}^2)$, $\boldsymbol{\eta}_{a} \overset{\Delta}{=} \mathrm{N}(\mathbf{0}, \sigma_{a}^2)$ are zero-mean Gaussian noises whose standard deviations are σ_g and σ_a; ${}^{b}\mathbf{R}_{w} \in \mathbb{SO}(3)$ denotes a rotation from frame w to frame b [39]. The biases are computed following the random walk process given as [13],

$$ \dot{\mathbf{b}}^{g} = \boldsymbol{\eta}_{b\omega}, \; \dot{\mathbf{b}}^{a} = \boldsymbol{\eta}_{ba}, \tag{12} $$

where $\boldsymbol{\eta}_{b\omega}$ and $\boldsymbol{\eta}_{ba}$ are white Gaussian noises [13]. The discrete IMU kinematic model can be established as follows [20],

$$ \begin{aligned} {}^{w}\mathbf{R}_{b}^{i+1} &= {}^{w}\mathbf{R}_{b}^{i} \exp(\lfloor \boldsymbol{\omega}_{b} \Delta t_{i} \rfloor_{\times}) \\ {}^{w}\mathbf{v}_{b}^{i+1} &= {}^{w}\mathbf{v}_{b}^{i} + {}^{w}\mathbf{a}_{b}^{i} \Delta t_{i}, \\ {}^{w}\mathbf{p}_{b}^{i+1} &= {}^{w}\mathbf{p}_{b}^{i} + {}^{w}\mathbf{v}_{b}^{i} \Delta t_{i} + \frac{1}{2} {}^{w}\mathbf{a}_{b}^{i} \Delta t_{i}^{2}, \end{aligned} \tag{13} $$

where Δt_i is the sampling time, ${}^{w}\mathbf{a}_{b}^{i} = {}^{w}\mathbf{R}_{b}^{i}\mathbf{a}_{b} + \mathbf{g}$ is the free acceleration in w coordinate, and $\mathbf{g}$ is the gravity vector. The IMU preintegration technique can efficiently free the computational cost, avoiding re-calculating the propagation task during each IMU cycle. The core idea is to detach the velocity and position into the gravity and acceleration components. A set of IMU data $\mathcal{I}_{ij}$ between two consecutive keyframes i and j is perceived, as shown in Fig. 4-c. Let drop the denotation frames b and w (13) for readability. By substituting the IMU measurement (11) into the IMU kinematic model (12) (13). The inertial propagation process with a batch of IMU data from time i to j can draw into the residual of rotation, position, and velocity as,

$$ \mathbf{r}_{\mathcal{I}_{ij}} = \left[\mathbf{r}_{\gamma_{ij}}^{\mathrm{T}}, \; \mathbf{r}_{\beta_{ij}}^{\mathrm{T}}, \; \mathbf{r}_{\alpha_{ij}}^{\mathrm{T}} \right], \tag{14} $$

where each element of residual $\mathbf{r}_{\mathcal{I}_{ij}}$ is computed given as,

$$ \begin{aligned} \mathbf{r}_{\alpha_{ij}} &= \mathbf{R}_{i}^{\mathrm{T}}(\mathbf{p}_{j} - \mathbf{p}_{i} - \mathbf{v}_{i}\Delta t - \frac{1}{2}\mathbf{g}\Delta t^{2}) - \Delta\tilde{\mathbf{p}}_{ij}(\mathbf{b}_{i}^{g}, \mathbf{b}_{i}^{a}) \\ \mathbf{r}_{\beta_{ij}} &= \mathbf{R}_{i}^{\mathrm{T}}(\mathbf{v}_{j} - \mathbf{v}_{i} - \mathbf{g}\Delta t) - \Delta\tilde{\mathbf{v}}_{ij}(\mathbf{b}_{i}^{g}, \mathbf{b}_{i}^{a}) \\ \mathbf{r}_{\gamma_{ij}} &= \mathrm{Log}(\Delta\tilde{\mathbf{R}}_{ij}(\mathbf{b}_{i}^{g}))\mathbf{R}_{i}^{\mathrm{T}}\mathbf{R}_{j}, \end{aligned} $$

where $\Delta t = t_j - t_i$. See [39] for more detail about the explanation.

The pseudo-measurements of rotation $\Delta\tilde{\mathbf{R}}_{ij}$, velocity $\Delta\tilde{\mathbf{v}}_{ij}$ and position $\Delta\tilde{\mathbf{p}}_{ij}$ are implemented as follows [39],

$$ \begin{aligned} \Delta\tilde{\mathbf{R}}_{ij} &= \prod_{\rho=i}^{j-1} \mathrm{Exp}((\tilde{\boldsymbol{\omega}}_{\rho} - \mathbf{b}_{i}^{g})\Delta t) \\ \Delta\tilde{\mathbf{v}}_{ij} &= \sum_{\rho=i}^{j-1} \Delta\tilde{\mathbf{R}}_{i\rho}(\tilde{\mathbf{a}}_{\rho} - \mathbf{b}_{i}^{a})\Delta t \\ \Delta\tilde{\mathbf{p}}_{ij} &= \sum_{\rho=i}^{j-1} \left[\tilde{\mathbf{v}}_{i\rho}\Delta t + \frac{1}{2}\Delta\tilde{\mathbf{R}}_{i\rho}(\tilde{\mathbf{a}}_{\rho} - \mathbf{b}_{i}^{a})\Delta t^{2} \right], \end{aligned} $$

These pseudo-measurements, independent of the gravity effect and state variables, can be directly computed from IMU measurements. Using Eq. 12, following the random walk process, the derivatives acceleration bias and gyroscope bias are Gaussian. Hence, a bias constraint between two consecutive keyframes at time i to j is taken into account by the IMU preintegration factor.

After having the computed preintegration residual, the noises at time step i are iteratively propagated to step j,

as shown in Fig. 4-c. Following [39], the noises propagation process is computed given as,

$$\boldsymbol{\psi}_{i,\rho+1} = \mathbf{A}_\rho \boldsymbol{\psi}_{i,\rho} + \mathbf{B}_\rho \boldsymbol{\eta}_{i,\rho}, \tag{15}$$

where $\boldsymbol{\psi}_{i,\rho} = \begin{bmatrix}\delta\boldsymbol{\phi}_{i,\rho} & \delta\mathbf{v}_{i,\rho} & \delta\mathbf{p}_{i,\rho}\end{bmatrix}^{\mathrm{T}}$; $\boldsymbol{\eta}_{i,\rho} = \begin{bmatrix}\boldsymbol{\eta}^d_{g_i} & \boldsymbol{\eta}^d_{a_i}\end{bmatrix}^{\mathrm{T}}$;

$$\mathbf{A}_\rho = \begin{bmatrix} \Delta\tilde{\mathbf{R}}^{\mathrm{T}}_{\rho,\rho+1} & \mathbf{0} & \mathbf{0} \\ -\Delta\tilde{\mathbf{R}}_{i\rho}(\tilde{\mathbf{a}}_\rho - \mathbf{b}^a_i)^\wedge \Delta t & \mathbf{I} & \mathbf{0} \\ -1/2\Delta\tilde{\mathbf{R}}_{i\rho}(\tilde{\mathbf{a}}_\rho - \mathbf{b}^a_i)^\wedge \Delta t^2 & \mathbf{I}\Delta t & \mathbf{I} \end{bmatrix} \tag{16}$$

$$\mathbf{B}_\rho = \begin{bmatrix} \mathbf{J}^\rho_r \Delta t & \mathbf{0} \\ \mathbf{0} & \Delta\tilde{\mathbf{R}}_{i\rho} \\ \mathbf{0} & 1/2\Delta\tilde{\mathbf{R}}_{i\rho}\Delta t^2 \end{bmatrix}, \tag{17}$$

where $\Delta t = t_{\rho+1} - t_\rho$; $\mathbf{J}^\rho_r \triangleq \mathbf{J}^\rho_r\big((\tilde{\boldsymbol{\omega}}_\rho - \mathbf{b}^g_i)\Delta t\big)$ denotes the right Jacobian [36]. Through the linear noise model (15), the preintegrated measurement covariance matrix is computed as follows,

$$\Sigma_{i,j} = \mathbf{A}_{j-1}\Sigma_{i,j-1}\mathbf{A}^{\mathrm{T}}_{j-1} + \mathbf{B}_{j-1}\Sigma_\eta\mathbf{B}^{\mathrm{T}}_{j-1}. \tag{18}$$

Furthermore, the Jacobians of each element of residual $\mathbf{r}_{\mathcal{I}_{ij}}$ concerning all variables at time step t_i and t_j is computed, as reported in [39].

B. Holonomic Wheel Odometry Factors

Following the kinematic model of the holonomic robot [1], [40], the instance velocity of the holonomic robot is calculated as follows,

$$\begin{bmatrix} {}^o\upsilon_x \\ {}^o\upsilon_y \\ {}^o\omega \end{bmatrix} = \frac{\rho}{4}\begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & 1 & -1 \\ -1/d & 1/d & -1/d & 1/d \end{bmatrix}\begin{bmatrix} {}^E\omega_0 \\ {}^E\omega_1 \\ {}^E\omega_2 \\ {}^E\omega_3 \end{bmatrix}, \tag{19}$$

where ${}^E\omega_i$ is the angular velocity of wheel i, ρ is the wheel's radius, and d is the mechanical size of the robot [1], [40]. Therefore, the general 3D velocity is computed as follows [41],

$${}^o\boldsymbol{\omega}_m = {}^o\omega\mathbf{e}_3 + \mathbf{n}_\omega \tag{20}$$

$${}^o\mathbf{v}_m = \begin{bmatrix}\mathbf{e}_1 & \mathbf{e}_2 & \mathbf{0}_3\end{bmatrix}\begin{bmatrix}{}^o\upsilon_x & {}^o\upsilon_y & 0\end{bmatrix}^{\mathrm{T}} + \mathbf{n}_\upsilon, \tag{21}$$

where $\mathbf{n}_\omega$ and $\mathbf{n}_\upsilon$ are zero-mean Gaussian noises of the angular and linear velocity, respectively. Like Eq. 13, the robot kinematic model using wheel odometer is given as,

$${}^w\mathbf{R}^{i+1}_o = {}^w\mathbf{R}^i_o \mathrm{Exp}({}^o\boldsymbol{\omega}^i_m \Delta t) \tag{22}$$

$${}^w\mathbf{v}^{i+1}_o = {}^w\mathbf{R}^{i+1}_o {}^o\mathbf{v}^{i+1}_m. \tag{23}$$

$${}^w\mathbf{p}^{i+1}_o = {}^w\mathbf{p}^i_o + {}^w\mathbf{v}^i_o\Delta t + \frac{1}{2}\big({}^w\mathbf{v}^{i+1}_o - {}^w\mathbf{v}^i_o\big)\Delta t \tag{24}$$

Therefore, we can calculate the robot state at the wheel encoders clock as follows,

$${}^w\mathbf{N}^i_b = \big[{}^w\mathbf{R}^i_o {}^o\mathbf{R}_b;\ {}^o\mathbf{R}_b {}^w\mathbf{p}^i_o + {}^o\mathbf{t}_b;\ {}^o\mathbf{R}_b {}^w\mathbf{v}^i_o\big], \tag{25}$$

where ${}^o\mathbf{T}_b = \begin{bmatrix} {}^o\mathbf{R}_b & {}^o\mathbf{t}_b \\ \mathbf{0} & 1 \end{bmatrix} \in \mathbb{SE}(3)$ is the transformation from wheel odometry frame to body frame, is known. Let us assume that body frame b coincides with wheel odometry frame o, so ${}^o\mathbf{T}_b = \begin{bmatrix}\mathbf{I}_3 & \mathbf{0} \\ \mathbf{0} & 1\end{bmatrix}$. The relative motion factor of the robot is expressed as follows,

$${}^w\mathbf{r}_{p_{ij}} = \tilde{\mathbf{T}}_{p_{ij}} \boxminus \mathbf{T}_{p_{ij}} = \tilde{\mathbf{T}}_{p_{ij}} \boxminus \mathrm{Exp}({}^w\mathbf{T}_{p_i} \boxminus {}^w\mathbf{T}_{p_j}), \tag{26}$$

where $\tilde{\mathbf{T}}_{p_{ij}}$ is the observed relative motion in (25); ${}^w\mathbf{T}_{p_i} \in \mathbb{SE}(3)$ is the robot pose at time step i. We note that the wheel odometer factor (26) is calculated in the global coordinate instead of the temporary local coordinate [42]. We note that the global wheel coordinate is the world coordinate that coincides with the first position of the robot. Next, the velocity factor in the global coordinate can be established as follows,

$${}^w\mathbf{r}_{v_i} = {}^w\mathbf{v}^i_b - {}^w\tilde{\mathbf{v}}^i_b = {}^w\mathbf{v}^i_b - {}^w\mathbf{R}^i_b {}^o\mathbf{R}^{\mathrm{T}}_b {}^o\tilde{\mathbf{v}}^i_m. \tag{27}$$

Similarly, we determine the zero velocity factor as follows,

$${}^w\mathbf{r}^{zero}_{v_i} = {}^w\mathbf{v}^i_b - \mathbf{0}. \tag{28}$$

To minimize the cost function (9) using the iterative optimization process [17], we need to calculated the Jacobians. Let us calculate the Jacobian of relative motion residual ${}^w\mathbf{r}_{p_{ij}}$ with respect to each variable,

$$\mathbf{J}^{{}^w\mathbf{r}_{p_{ij}}}_{{}^w\mathbf{T}_{p_j}} = \mathbf{J}_l^{-1}({}^w\mathbf{r}_{p_{ij}}), \tag{29}$$

$$\mathbf{J}^{{}^w\mathbf{r}_{p_{ij}}}_{{}^w\mathbf{T}_{p_i}} = -\mathbf{J}_r^{-1}(\boldsymbol{\tau}_{p_{ij}}). \tag{30}$$

Similarly, we can compute the Jacobian matrices for the velocity and zero velocity factors,

$$\mathbf{J}^{{}^w\mathbf{r}_{v_i}}_{{}^w\mathbf{v}^i_b} = \mathbf{I}, \tag{31}$$

$$\mathbf{J}^{{}^w\mathbf{r}_{v_i}}_{{}^w\mathbf{R}^i_b} = {}^w\mathbf{R}^i_b \left\lfloor {}^o\tilde{\mathbf{v}}^i_m \right\rfloor_\times, \tag{32}$$

$$\mathbf{J}^{{}^w\mathbf{r}^{zero}_{v_i}}_{{}^w\mathbf{v}^i_b} = \mathbf{I}. \tag{33}$$

See Appendix A for the proof.

The global velocity (23) is formed as follows,

$$\begin{aligned} {}^w\mathbf{v}^j_o &= {}^w\mathbf{R}^j_o {}^o\hat{\mathbf{v}}^j_m = {}^w\mathbf{R}^j_o({}^o\mathbf{v}^j_m - \mathbf{n}_\upsilon) \\ &= {}^w\mathbf{R}^j_o {}^o\mathbf{v}^j_m - {}^w\mathbf{R}^j_o\mathbf{n}_\upsilon = {}^w\hat{\mathbf{v}}^j_o - \delta\mathbf{v}^j_o. \end{aligned} \tag{34}$$

Hence, the velocity noise is expressed as,

$$\delta\mathbf{v}^j_o = {}^w\mathbf{R}^j_o\mathbf{n}_\upsilon. \tag{35}$$

From Eq. (20), on 2D planar, the robot orientation is determined as,

$$\phi_{i+1} + \delta\phi_{i+1} = \phi_i + \delta\phi_i + ({}^o\omega + \eta_\omega)\Delta t, \tag{36}$$

where ϕ_i is the yaw angle at time step i. Hence, the yaw angle noise is denoted as,

$$\delta\phi_{i+1} = \delta\phi_i + \eta^i_\omega\Delta t. \tag{37}$$

The robot orientation noise from time step i to j is also computed as follows,

$$\delta\phi_{ij} = \delta\phi_i + \sum_{k=1}^{j-i}\eta^k_\omega\Delta t. \tag{38}$$

Accordingly, we can solve the orientation noise as given,

$$\delta\mathbf{R}_{ij} = \delta\phi_{ij}\mathbf{e}_3. \tag{39}$$

From Eq. (24), the relative measurement noise is propagated following the accumulative encoder errors as yields,

$$\Delta\mathbf{p}_{k,k+1} = {}^{b}\mathbf{R}_{w}^{k}({}^{w}\mathbf{p}_{o}^{k+1} - {}^{w}\mathbf{p}_{o}^{k}) = \Delta\hat{\mathbf{p}}_{k,k+1} - \delta\mathbf{p}_{k,k+1}. \quad (40)$$

Therefore, the relative motion noise is propagated given as,

$$\delta\mathbf{p}_{k,k+1} = \frac{1}{2}{}^{k}\mathbf{R}_{k+1}\mathbf{n}_{v}\Delta t + \frac{1}{2}\mathbf{n}_{v}\Delta t = \frac{1}{2}({}^{k}\mathbf{R}_{k+1} + \mathbf{I})\mathbf{n}_{v}\Delta t,$$

where $\mathbf{n}_v$ remains constant. The proof is present in Appendix B. So, the relative noise from time i to j is given as,

$$\delta\mathbf{p}_{ij} = \frac{1}{2}\sum_{k=i}^{j-1}({}^{k}\mathbf{R}_{k+1} + \mathbf{I})\mathbf{n}_{v}\Delta t. \quad (41)$$

Finally, the measurement uncertainties are updated following Eq. (35), (39) and (41).

C. Learning Observation and Noise Model Based on Neural Network

The AI techniques are used to learn the relative state motion and noise model updated in the BF solution, as shown in Section II. Here, the AI based-estimation problem can be represented as,

$$\mathcal{Y} = \mathcal{G}_{\text{AI}}(\mathcal{I}, \theta), \quad (42)$$

where $\mathcal{G}(.)$ indicates the AI model such as deep neural networks or recurrent neural networks (RNN), et cetera; θ is the trainable parameters of the AI model; $\mathcal{Y}$ is the BF parameters as shown in Eq. 9; $\mathcal{I}$ is the input of the AI technique as the sensor data or historical states.

In particular, in order to feed into the neural network (NN), the sensor measurements in a sliding window S are stacked as follows,

$$\mathcal{I}_{\alpha}^{S} = \mathcal{G}_{RNN}(\alpha\{\boldsymbol{\omega}_{b}^{\rho}, \mathbf{a}_{b}^{\rho}, \mathbf{v}_{m}^{\rho}, \boldsymbol{\omega}_{m}^{\rho}\}\big|_{\rho\triangleq i:j}) \quad (43)$$

$$\mathcal{I}_{\beta}^{S} = \mathcal{G}_{NN}(\beta\{\boldsymbol{\omega}_{b}^{\rho}, \mathbf{a}_{b}^{\rho}, \mathbf{v}_{m}^{\rho}, \boldsymbol{\omega}_{m}^{\rho}\}\big|_{\rho\triangleq i:j}), \quad (44)$$

where $\mathcal{G}_{RNN}$ and $\mathcal{G}_{NN}$ are the RNN function and neural network function, respectively; α is a function to merge the sensor data into the sequential vectors; β denotes the vectorization function to combine the sensor outputs into a unique vector; $\mathcal{I}_{\alpha}^{S}$ and $\mathcal{I}_{\beta}^{S}$ are the output of the neural networks utilized to predict the observation model or noise model. The overall solution is described as shown in Fig. 5. Here, $\mathcal{G}_{RNN}$ or $\mathcal{G}_{NN}$ is employed to calculate the AI factor $\mathbf{r}_{\psi}$ (9).

Remark 2: Two dominant approaches are using the neural network to assist the BF technique. On the one hand, the relative state motion can be estimated given a batch of sensor measurements called the learning observation model. The residual is designed similarly to (26). On the other hand, the noise model (41) can be predicted and used as the uncertainty information in BF. The experiment result of each NN solution is evaluated in the next section.

V. Experiments

In the following experiments, we will show the implementation as well as the performance of the proposed method. First, in the implementation part, we will describe the system setup and deal with sensor synchronization, zero velocity detection,

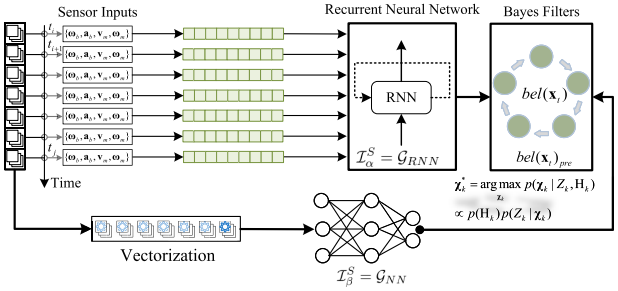

Fig. 5. Overview of the neural network-based state estimation. There are two approaches consisting of the NN solution and the RNN method. The sensor's data input is converted to a unique vector, which feeds into a neural network. The sensor's data input can also be transformed sequentially to an RNN model. The neural network's output is then employed to the Bayesian filters such as EKF or FGO, as shown in Fig. 2.

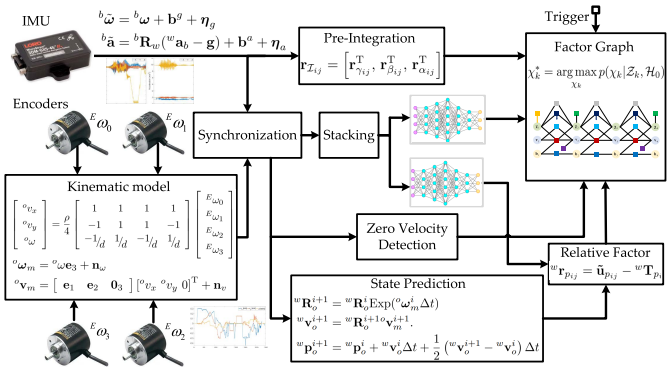

Fig. 6. Pipeline of the proposed method uses an IMU sensor and wheel encoders. The neural networks are applied to predict the relative motion and noise model adopted to a factor graph.

and algorithm design. Second, the neural networks for the predicting problem of relative state motion and noise model are presented. Finally, we report the experimental evaluations in which the performance of the proposed method is compared to other methods.

A. Implementation

The architecture of the proposed estimator design is described as shown in Fig. 6, which consists of four parts: sensor synchronization, IMU preintegration and wheel encoder factor, neural network prediction, and factor graph solver. The system operates within three parallel threads, including wheel encoder handling, IMU processing, and neural prediction and factor graph optimization.

1) Zero Velocity Detection Factor: Zero velocity factor (28) can limit the drift and exactly reset the IMU sensor bias. The robot detect stationary when zero translation and rotation are observed, then a zero velocity factor (28) is immediately added to the factor graph (9). In particular, the stationary motion can be figured out using the wheel encoder measurements in which the sensor data reported zero velocities with a small threshold of 0.1 mm translation and 0.5 degree rotation.

2) Sensor Synchronization: In practice, the IMU and wheel encoder is configured at the same frequency of 100 Hz. However, the IMU sensor data and wheel velocity are not

observed simultaneously, as shown in Fig. 4-c. The inertial sensor and wheel velocity need to synchronize to deploy the neural network and factor graph, as shown in Fig. 6. Herein, the IMU clocks are set as the references. Let observing the wheel velocities $[v_x^n\ v_y^n\ \omega_x^n]$ and $[v_x^{n+1}\ v_y^{n+1}\ \omega_x^{n+1}]$ at two consecutive time steps n and $n+1$, as shown in Fig. 4. Let us assume that the acceleration had remained constant during time t_n to t_{n+1}. Therefore, the acceleration is reckoned as follows,

$$[a_x^n,\ a_y^n,\ \zeta_x^n] = \left[\frac{v_x^{n+1}-v_x^n}{\Delta t},\ \frac{v_y^{n+1}-v_y^n}{\Delta t},\ \frac{\omega^{n+1}-\omega^n}{\Delta t}\right],$$

where $\Delta t = t_{n+1} - t_n$. Let t_γ is an IMU time step that falls between time t_n and t_{n+1} as shown in Fig. 4. Hence, the velocity at time t_γ is calculated as follows,

$$[v_x^\gamma\ v_y^\gamma\ \omega_x^\gamma]^{\mathrm{T}} = [v_x^n\ v_y^n\ \omega_x^n]^{\mathrm{T}} + [a_x^n\, a_y^n\, \zeta_x^n]^{\mathrm{T}}(t_\gamma - t_n).$$

Therefore, the velocity is synchronized at the IMU clock computed as follows,

$$\mathbf{v}_\gamma = [\mathbf{e}_1\, \mathbf{e}_2\, \mathbf{0}][v_x^\gamma\ v_y^\gamma\ 0]^{\mathrm{T}} \tag{45}$$

$$\boldsymbol{\omega}_\gamma = \omega_x^\gamma \mathbf{e}_3. \tag{46}$$

3) Algorithm: We design Algorithm 1 following the pipeline, as shown in Fig. 6. The program is implemented with three parallel threads in which thread $\mathbf{\Omega}_W$ and $\mathbf{\Omega}_{IMU}$ are employed to observe wheel encoder and IMU measurement, respectively. Unlike other open sources [22], [25], [26], [43], the central computing thread is processed by using a fast timer checking the thread to observe the size of the global buffers. Whenever the size reaches 20, the optimization trigger is computed in this thread. The technique helps us separate the IMU and wheel recording thread, which can continue registering data while optimizing.

Algorithm 1 Inertial-Wheel Fusion Algorithm

Input: IMU sensor and wheel encoder data
Output: Robot state

```
Initialization system: IMU, wheel encoder sensor;
Init IMU thread Ω_IMU and wheel encoder thread Ω_W;
Init checking timer thread (1ms) Ω_T;
while wheel encoder obtained (Ω_W) do
    Calculate wheel velocity 19;
    Add to global wheel buffer;
    Integration and generate temporary estimator;
end
while IMU data observed (Ω_IMU) do
    Convert to body coordinate;
    Add to IMU global buffer;
end
while checking time (Ω_T) do
    if the size of IMU global buffer is 20 then
        Synchronization and stacking data V-A2;
        Compute neural network factorIV-C;
        Compute and add IMU factor IV-A;
        Compute and add wheel factor IV-B;
        if Zero velocity V-A1 then
            add zero factor 28;
        end
        Solve the graph optimization (9);
        Update bias for IMU preintegration;
        if the size of the graph is maximum then
            Reset the factor graph;
            Add the marginalization factor r_M;
        end
        Generate optimal state;
    end
end
```

B. Experimental Setup and Dataset

1) Experimental Setup: A holonomic mobile robot platform with four omnidirectional wheels was used to conduct the experiment evaluation, as shown in Fig. 1. The robot was equipped with an IMU, four Kamotek wheel-encoders,[1] two Sick 2D LiDAR sensors, and two Zed 2 cameras. The specification of the sensors is described in Table. II. An embedded computer-NVIDIA Jetson AGX Xavier[2] with Linux-based operating system and ROS Melodic[3] using C++ language was employed to implement Algorithm 1.

2) Evaluation Dataset and Preprocessing Data: The holonomic robot was employed to collect the datasets in an industrial environment. The ground truth was created by using the multi-sensor fusion of LiDAR, IMU, and camera, which provided a superior accuracy trajectory in the structured environment with localization mode [44]. Raw inertial data, including angular velocity and acceleration of one experiment, is shown in Fig. 7. Following Eq. (19), we can calculate the wheel velocity, which is filtered with a low-pass filter [45] and transformed to the global coordinate. A sample wheel velocity is described in Fig. 8. The zero velocities are straightforward to recognize in the sample dataset, as shown in Fig. 8.

TABLE II
SENSOR SPECIFICATION OF THE EXPERIMENTAL SETUP

Sensor	Model	Hz	Specification
IMU *01*	MicroStrain 3DM-GX5-GNSS/INS	100	*Bias Stab:* 0.004 mg; 8°/hr *Init Bias:* 2 mg; 0.04°/s
2D LiDAR *02*	Sick LiDAR S30B-3011BA	12.5	*Meas Range:* 0-30m *Angular Res:* 0.5°
Camera *02*	Zed 2 Stereo Camera	30	*Res:* 2560x720 px *FoV:* 110° (H) x 70° (V)
Encoder *04*	Kamotek Motor	100	2500 Pulse/Round 15 wires

The data smoothing and outlier removal were employed to eliminate the noise of the estimated trajectory. Here, the datasets were smoothed by applying the Gaussian-weighted moving average [45]. The smoothed datasets had then cleaned the outlier using the Modified Akima cubic interpolation [46]. A sample dataset was treated, as shown in Fig. 9.

C. Learning Relative State Motion and Noise Model

As shown in Subsection IV-C, the neural networks were used to predict the factor parameters in graph optimization (9).

[1] http://komotek.com/
[2] http://developer.nvidia.com/embedded/jetson-agx-xavier-developer-kit
[3] http://wiki.ros.org/melodic

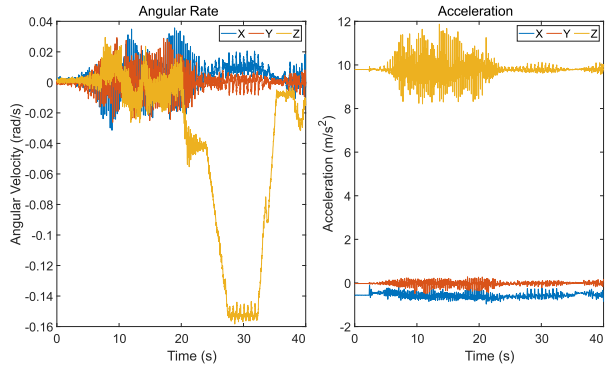

Fig. 7. Angular velocity and acceleration were recorded from one of the datasets.

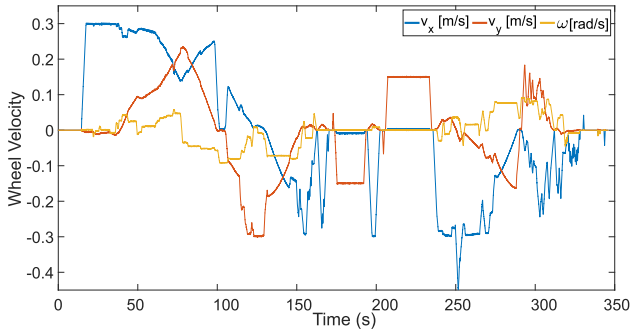

Fig. 8. Sample dataset of the wheel velocity in global coordinate was recorded.

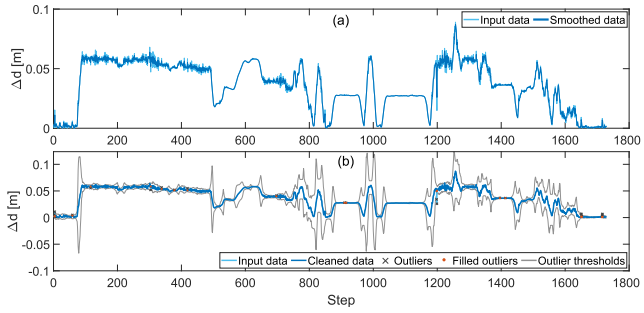

Fig. 9. Estimated trajectory was preprocessed by using the data smoothing and outlier removal.

Herein, the neural network's input is the IMU and wheel velocity data, and the neural network's output is the relative state motion or sensor noise model of two consecutive robot poses. The input of the recurrent neural network is fed continuously at each time step as Eq. (43). Besides, the IMU and wheel velocity information can also be assembled as a unique vector (44) to be supplied to the neural network once.

Remark 3: In practice, the rotation is accurately computed using the angular velocity of inertial data and wheel encoder. Therefore, the estimation output only need to handle the relative translation values, which are two dimensional $[\Delta x, \Delta y]$ *(*$y = 2$*) or one dimensional* $\Delta d = \sqrt{\Delta x^2 + \Delta y^2}$ *(*$y = 1$*). Later, the NN relative motion is built as a relative motion factor indicated as Eq. 26. Moreover, the noise model of the wheel odometer can be learned with the same input.*

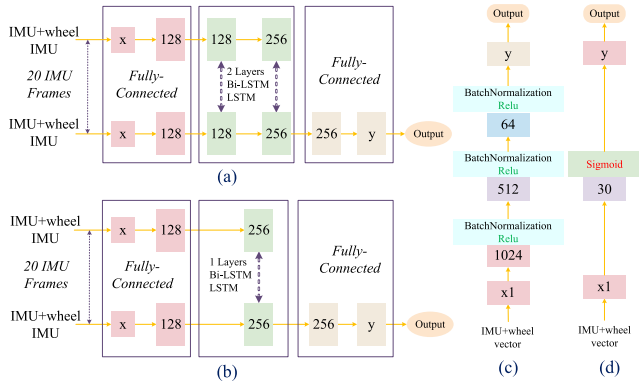

Fig. 10. Pipeline of neural networks has been studied. The network's input is constructed by an IMU sensor and wheel encoder data. Here, $x = 9$ and $x1 = 180$ if using IMU sensor and wheel encoders, $x = 6$ and $x1 = 120$ using only IMU sensor; The output predicts the relative state motion (y = 2 or y = 1) or noise model (y = 1). (**a**)-the neural network uses two layers of Bi-LSTM or LSTM; (**b**)-the network with one layer of Bi-LSTM or LSTM; (**c**)-a deep neural network structure; (**d**)-an MLP network with two layers.

The lost function for two dimensional ($y = 2$) is simply the mean square error (MSE) between the mean of the prediction state motion and actual relative state at each timestep as,

$$L_2 = \frac{1}{n}\sum_{i=1}^{n}\left(\Delta\mathbf{x}_i^p - \Delta\hat{\mathbf{x}}_i\right)^{\mathrm{T}}\left(\Delta\mathbf{x}_i^p - \Delta\hat{\mathbf{x}}_i\right), \tag{47}$$

where $\Delta\mathbf{x}_i^p = [\Delta x_i, \Delta y_i]$ is the prediction state motion; $\Delta\hat{\mathbf{x}}_i = [\Delta\hat{x}_i, \Delta\hat{y}_i]$ is the true relative state; n is the number of timestep.

Similarly, The lost function for one dimensional ($y = 1$) is denoted by the MSE of the relative distance of belief and true state at each timestep as,

$$L_1 = \sum_{i=1}^{n}\frac{1}{n}\left(\Delta d_i^p - \Delta\hat{d}_i\right)^2, \tag{48}$$

where Δd_i^p and $\Delta\hat{d}_i$ are the relative distance of prediction and actual state.

1) Learning Observation Model: We investigate the neural network architectures for the end-to-end learning relative state motion. The network structures are designed following three types of neural network architectures, as shown in Fig. 10. The first scheme employs the recurrent neural network (RNN) designed following the IONet architecture [8]. Instead of using only IMU data as IONet [8] ($x = 6$), we add the wheel velocities into the input ($x = 9$), as shown in Subsection IV-C. The system can utilize two Bi-LSTM or LSTM, as shown in Fig. 10-a. The two layers network also can employ one-layer Bi-LSTM or LSTM to reduce the learnable weights, as shown in Fig. 10-b. Moreover, a deep neural network can conducts to learn the relative state model [47], as shown in Fig. 10-c. Finally, a multilayer neural network (MLP) with two layers is applied for the prediction problem, as shown in Fig. 10-d. The total number of weights and activation functions for each neural network with $x = 9, y = 1$ is shown in Table III. Here, the total number of weights of two layers of Bi-LSTM is

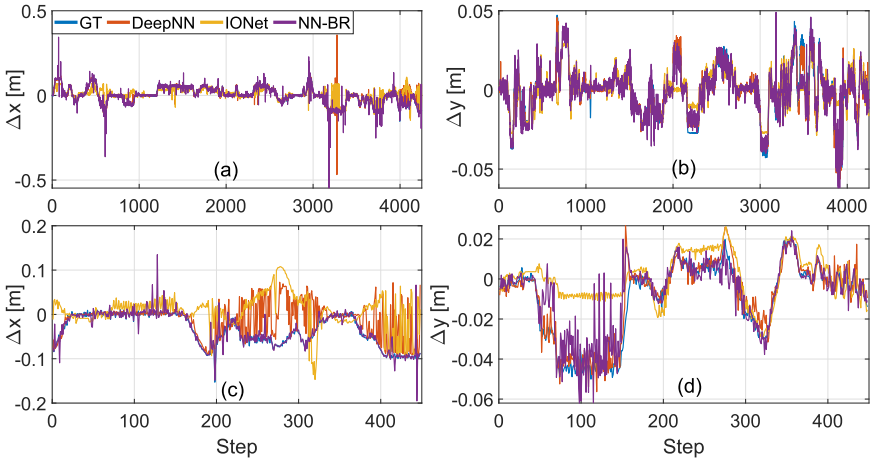

Fig. 11. Performances of each neural network handling two outputs $[\Delta x, \Delta y]$. (**a**) Prediction of Δx. (**b**) Prediction of Δy. (**c**) and (**d**) Enlarged from one part of (**a**) and (**b**), respectively.

TABLE III
PROPERTIES OF THE NEURAL NETWORK ARCHITECTURES

Neural Network	No. Layer	No. Activation Function	No. Learnable Parameters
IONet- 2 Bi-LSTM	5	1163	1446657
IONet- 2 LSTM	5	779	593117
IONet- 1 Bi-LSTM	4	907	921345
IONet- 1 LSTM	4	651	461569
Deep NN	7	4982	746241
MLP	2	31	5461

approximately double the deep NN and is more than 260 times the shallow MLP, as presented in Table III.

The neural networks were trained on a desktop computer using an 8GB NVIDIA RTX 3070TI GPU. For the LSTM and deep NN, the dataset was randomly shuffled during each epoch. We employed a batch size of 2048 and an ADAM optimizer. The initial learning rate was practically selected at 0.001. The Levenberg-Marquardt (LM) [48] and Bayesian Regularization (BR) algorithm [49] were used to train the MLP. Although the LM algorithm could quickly train the neural network, the BR method was more accurate and suitable for the noise dataset [49]. The evaluation of each neural network is specified by each training method, as shown in Table IV. For $y = 2$, the MLP accuracy achieved 0.02, which was 1.5 times better than the Bi-LSTM but worse than Deep NN, approximately two times. For $y = 1$, the multilayer neural network accuracy was just about 0.0014, and the Bi-LSTM and deep NN were around 0.009 and 0.008, respectively. The MLP with the LM and BR algorithm could be obtained with 8 to 9 times better accuracy than Bi-LSTM and Deep NN. The results of each neural network with two outputs ($y = 2$) are illustrated, as shown in Fig. 11. Although the response of Δx to the ground truth of each neural network was suitable, the accuracy of Δy was fair, as shown in Fig. 11. The evaluation results of each neural network with one output ($y = 1$) are presented, as shown in Fig. 12. In this case, the MLP provided the best results compared with Bi-LSTM and deep NN. The MLP using the BR algorithm could smoothly match the ground truth. Although the MLP adopting the LM method provided good results, the results were noise. The

TABLE IV
EXPERIMENTAL RESULTS OF EACH NEURAL NETWORK ARCHITECTURE

Network	Training RMSE (m)	Validation RMSE (m)	Training Time (s)	Specs
IONet 2 Bi-LSTM	0.035	0.05	860	*adam*
(y=2 AND y=1)	0.009	0.01	600	*GPU*
IONet 1 Bi-LSTM	0.05	0.06	510	*adam*
(y=2 AND y=1)	0.01	0.009	320	*GPU*
IONet 2 LSTM	0.05	0.07	530	*adam*
(y=2 AND y=1)	0.01	0.008	200	*GPU*
IONet 1 LSTM	0.05	0.06	330	*adam*
(y=2 AND y=1)	0.01	0.012	180	*GPU*
Deep NN	0.01	0.07	1080	*adam*
(y=2 AND y=1)	0.008	0.04	1050	*GPU*
MLP (H=30)	0.03	0.08	10	*LM*
(y=2 AND y=1)	0.0058	0.0045	10	*GPU*
MLP (H=60)	0.03	0.08	20	*LM*
(y=2 AND y=1)	0.0066	0.0063	20	*GPU*
MLP (H=30)	0.02	0.07	200	*LM*
(y=2 AND y=1)	0.0032	0.0045	200	*CPU*
MLP (H=30)	0.02	0.07	1500	*BR*
(y=2 AND y=1)	0.0014	0.0015	1500	*CPU*
IONet 2 Bi-LSTM (x=6, y=1)	0.03	0.03	620	*adam* *GPU*
Deep NN (x=6, y=1)	0.0046	0.01	520	*adam* *GPU*
MLP (H=30) (x=6, y=1)	0.0074	0.01	125	*LM* *CPU*
MLP (H=30) (x=6, y=1)	0.0052	0.01	820	*BR* *CPU*
MLP (H=60) (x=6, y=1)	0.0006	0.005	3800	*BR* *CPU*

Bi-LSTM and deep NN could track the ground truth with a gap. The last five rows in Table IV indicated the predicting results using only IMU input ($x = 6$). Here, the deep NN achieved 0.0046, which was better than MLP with H = 30. When the number of neural increased to H = 60, the accuracy of MLP was better than Deep NN. However, the results of MLP were pretty sensitive to the noise of IMU, as shown in Fig. 13. Here, IONet got poor results compared to the deep NN and MLP.

Remark 4: Using only IMU, the MLP is sensitive to sensor noises. In this case, the experimental results confirmed that the deep neural network has more potential than other neural networks.

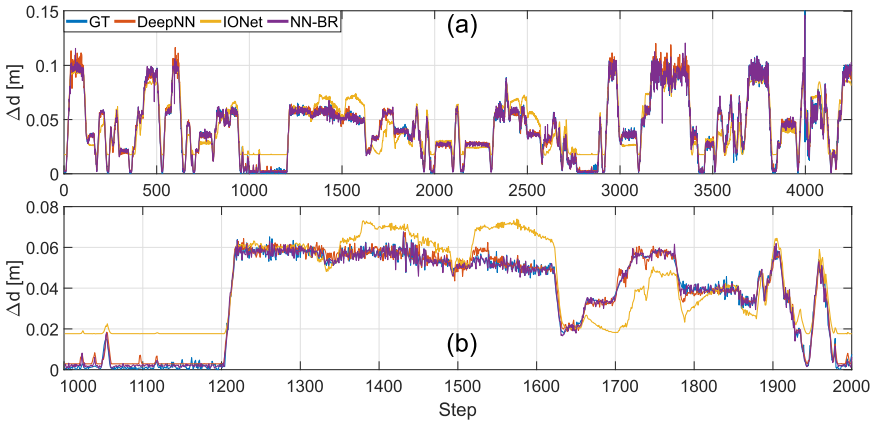

Fig. 12. Accuracy of each neural network with one output Δd. (**a**) Comparison of each neural network estimation. (**b**) Result is enlarged from step 1000 to 2000.

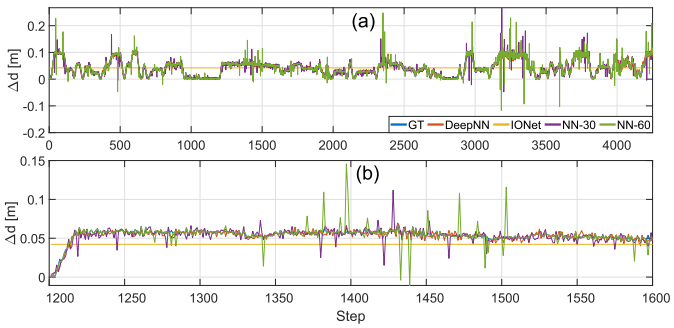

Fig. 13. Accuracy of each neural network using only IMU as input $x = 6$ and $x1 = 120$. (**a**) Comparison of each neural network estimation. (**b**) Result is enlarged from step 1200 to 1600.

Remark 5: The MLP is deployed to learn the relative state motion, as shown in Fig. 6.

2) Neural Networks for Predicting Noise Model: Similarly, the NN architectures are used to learn the noise model of wheel relative motion factor (26), as shown in Fig. 10. The loss function for the noise model is defined as,

$$L_e = \sqrt{\sum_{i=1}^{n} \frac{1}{n} e_k^2}, \tag{49}$$

where Δe_k is the output created by a difference of the relative motion of wheel odometer prediction and ground truth, as follows,

$$\Delta e_k = \sqrt{(x_{k+1}^w - x_k^w)^2 + (y_{k+1}^w - y_k^w)^2} - \sqrt{(x_{k+1}^{gt} - x_k^{gt})^2 + (y_{k+1}^{gt} - y_k^{gt})^2}, \tag{50}$$

where $[x_k^w, y_k^w]$ and $[x_k^{gt}, y_k^{gt}]$ are the 2D position of the robot computed by wheel encoders and ground truth, respectively. The performance of each neural network is shown in Table V. In this case, the training accuracy of MLP with the BR algorithm was the best at only 0.0017. MLP achieved the accuracy 1.8 times better than a deep neural network of 0.003 and 5.3 times than IONet. Here, IONet provided the worst results compared with MLP and deep NN. The response results of the neural networks are shown in Fig. 14.

TABLE V
NOISE MODEL PREDICTION OF EACH NEURAL NETWORK ARCHITECTURE

Network	Training RMSE (m)	Validation RMSE (m)	Training Time (s)	Specs
IONet 2 Bi-LSTM	0.009	0.008	600	*'adam' GPU*
Deep NN	0.003	0.04	1050	*'adam' GPU*
MLP (H=30)	0.0058	0.0049	10	*'LM' GPU*
MLP (H=30)	0.003	0.0032	80	*'LM" CPU*
MLP (H=30)	0.0017	0.0028	1500	*'BR' CPU*

Remark 6: The learning noise model using an MLP is applied in the factor graph optimization, as shown in Fig. 6. Therefore, the observation error covariance at keyframe k is then defined as,

$$\mathbf{Q}_k \triangleq \varphi_e^k diag\{\sigma_{\vartheta_1}^2, \sigma_{\vartheta_2}^2, \ldots, \sigma_{\vartheta_6}^2\}, \tag{51}$$

where $\sigma_{\vartheta_1}^2, \sigma_{\vartheta_2}^2, \ldots, \sigma_{\vartheta_6}^2$ are empirically determined; φ_e^k is the output of the neural network for predicting noise model.

D. State Estimation Comparison

The experimental results are evaluated following two standard metrics: the absolute trajectory error (ATE) and the relative trajectory error (RTE) [8]. The ATE is defined by average the root-mean-square error (RMSE) of predicted trajectory and ground truth as follows,

$$\text{ATE} = \sqrt{\frac{1}{n}\sum_{k=1}^{n} \left\| \mathbf{p}_k - \mathbf{p}_k^{gt} \right\|^2 (m)} \tag{52}$$

where $\mathbf{p}_k$ and $\mathbf{p}_k^{gt}$ are the current position and ground truth at time step k; n is the number of sampling. The process of ATE rotation is similar.

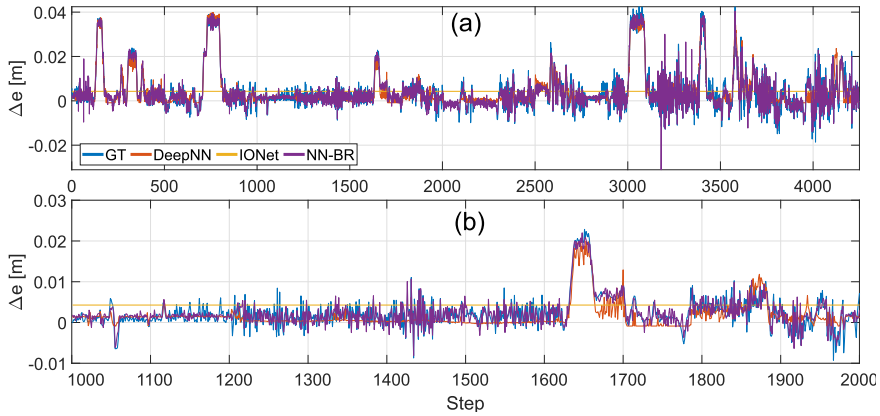

Fig. 14. (**a**) Results of each neural network architecture to learn the sensor noise model. (**b**) Same results were enlarged from time step 1000 to 2000.

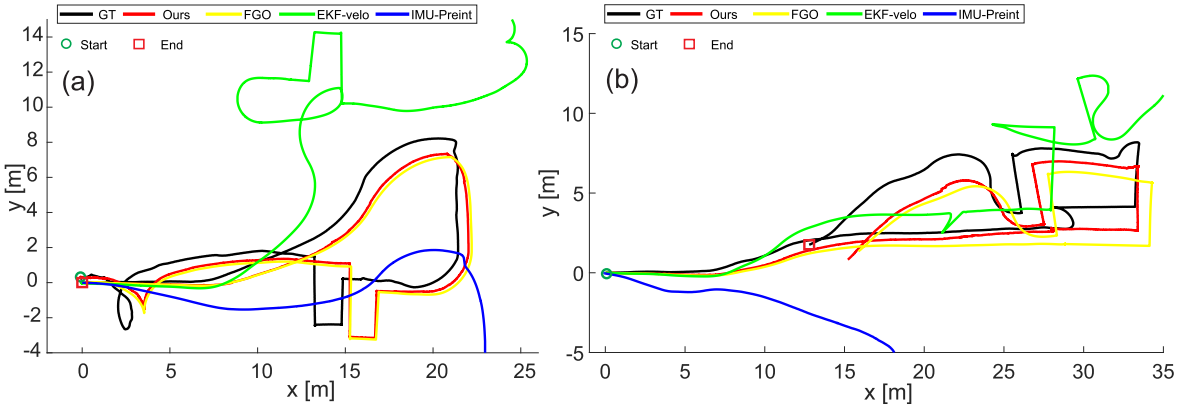

Fig. 15. Comparison of the proposed method to other solutions is shown. (**a**) Represents the trajectories of each solution in a trained dataset. (**b**) Illustrates the estimated trajectories in an untrained dataset.

The RTE denotes the average RMSE of the estimated and ground truth trajectory in a constant period of 0.2 ms as,

$$\mathrm{RTE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} \left\| \Delta \mathbf{p}_i - \Delta \mathbf{p}_i^{gt} \right\|^2} \tag{53}$$

where $\Delta \mathbf{p}_i$ and $\Delta \mathbf{p}_i^{gt}$ are the relative motion of the robot pose and its ground truth.

The proposed algorithm is compared with the other methods as follows,

- **EKF:** The EKF state estimation enabled fuse inertial data and wheel velocity for the robot [43]. The ROS configuration and source code are available in the robot localization toolbox.[4]
- **IMU:** The IMU-preintegration estimation with updated bias each optimization period as shown in Subsection IV-A. This technique is a cumulative relative robot motion computed by the IMU preintegration.
- **FGO:** The proposed factor graph optimization is operated without using neural network prediction.

Fig. 15-a,b illustrate the predicted trajectories of each solution using two datasets. The dataset in Fig. 15-a is trained, and Fig. 15-b is not trained. It was seen that the proposed approach provides the estimated trajectories more closely to ground truth. The proposed method outperformed other approaches in global coordinate. The EKF-Velo and IMU-preintegration results were worse in that their predicted trajectories ran outside of scope. Fig. 15-a represented the estimated trajectory of FGO that was also close to the proposed system with slight differences. However, the proposed solution was much better than FGO, which was far from the ground truth, as shown in Fig. 15-b. The ATE and RTE quantitative results are reported in Table VI. The proposed approach achieved the best performance compared with other solutions. The overall ATE errors were approximately 1.3 m in translation and 0.02 rad in rotation. The mean RTE error was only 0.02 m for translation and 0.001 rad for rotation. Compared to FGO, the results of ATE and RTE in rotation were pretty similar. However, for ATE translation, our approach was much better than FGO. EKF and IMU-preintegration presented poor results for ATE in which they ran out of scope.

Remark 7: In Fig. 15, the estimated trajectories demonstrate that the relative accuracy can be learned accurately when the robot moves straight following the x and y axis. The estimation is not too efficient when the robot moves and turns simultaneously. Therefore, the robot should travel parallel to the x or y axis or rotate without moving to achieve the best estimating performance.

[4]https://github.com/cra-ros-pkg/robot_localization

TABLE VI
ATE AND RTE ACCURACY OF EACH ESTIMATION APPROACH

Data1 (66.27 m)	**EKF**	**IMU**	**FGO**	**Ours**
Trans. [m]	14.1111	12.0707	1.6740	***1.3450***
Trans. [RTE]	0.0225	0.0189	***0.0162***	***0.0162***
Rots. [m]	0.3823	0.1572	**0.0210**	0.0280
Rots. [RTE]	0.0053	0.0016	7.33e-04	***5.26e-04***
Data2 (78.64 m)	**EKF**	**IMU**	**FGO**	**Ours**
Trans. [m]	34.1809	15.240	2.5237	***1.3272***
Trans. [RTE]	0.0383	0.0250	0.0241	***0.0223***
Rots. [m]	0.3687	0.1443	0.0545	***0.0330***
Rots. [RTE]	0.0139	0.0013	***0.0011***	***0.0011***

TABLE VII
THE TIMING CONSUMPTION OF THE PROPOSED METHOD

Module Task	**Time Consuming [μs]**	**Freq. [Hz]**
Wheel	5	100
IMU	10	100
Checking Timer	1	1000
Preprocessing	250	5
Neural Network	25	5
FGO	900	5
Total	1500	5

E. Timing Consumption Analysis

Computational time is the most critical task in dealing with real-time applications. The computation time of the proposed method running on an 8-core ARM 64-bit CPU of JETSON AGX XAVIER is summarized in Table VII. Here, the IMU and wheel encoder operated at 100 Hz. A software timer was created at 1000 Hz to check the size of sensor data. These tasks consumed a short time of about 1-10 μs. The neural network using MLP demanded only 25 μs, and FGO was needed about 900 μs for a sliding window size of 100. The overall process was averagely required only 1.5 ms.

VI. CONCLUSION

We present a comprehensive report on the intrinsic sensors fusion of IMU and wheel encoder with neural networks based-adaptive factor graph optimization. First, we surveyed the state-of-the-art multi-sensor fusion techniques using Bayesian filtering and learning approaches. Then, a detailed explanation for state estimating problems using inertial and wheel encoder sensors is presented. At the core of the estimation system, we analyzed the different neural network architectures for learning the relative and uncertainty model. The results confirmed that a simple MLP technique could enhance the performance. We implemented a sensor system with multiple sensors of LiDAR, Camera, IMU, and wheel encoders on a holonomic mobile robot. The experimental results reported that the proposed technique can achieve the most accuracy and outperform other strategies. The researchers can quickly implement the proposed method in the multi-sensor fusion of intrinsic and extrinsic state estimation.

APPENDIX
ODOMETRY FACTOR

A. The Jacobians of Relative Motion

The Jacobians of relative motion factor is computed as follows,

$$\begin{aligned}
\mathbf{J}^{{}^{w}\mathbf{r}_{p_{ij}}}_{{}^{w}\mathbf{T}_{p_j}} &= \mathbf{J}^{{}^{w}\mathbf{r}_{p_{ij}}}_{\mathrm{Log}({}^{w}\mathbf{T}_{p_i}\boxminus{}^{w}\mathbf{T}_{p_j})}\mathbf{J}^{\mathrm{Log}({}^{w}\mathbf{T}_{p_i}\boxminus{}^{w}\mathbf{T}_{p_j})}_{{}^{w}\mathbf{T}_{p_i}\boxminus{}^{w}\mathbf{T}_{p_j}}\mathbf{J}^{{}^{w}\mathbf{T}_{p_i}\boxminus{}^{w}\mathbf{T}_{p_j}}_{{}^{w}\mathbf{T}_{p_j}^{-1}}\mathbf{J}^{{}^{w}\mathbf{T}_{p_j}^{-1}}_{{}^{w}\mathbf{T}_{p_j}} \\
&= (-\mathbf{I})\mathbf{J}_r^{-1}(\tau_{p_{ij}})\mathbf{Ad}^{-1}_{{}^{w}\mathbf{T}_{p_i}}(-\mathbf{Ad}_{{}^{w}\mathbf{T}_{p_j}}) \\
&= \mathbf{J}_r^{-1}({}^{w}\mathbf{r}_{p_{ij}})\mathbf{Ad}_{{}^{w}\mathbf{T}_{p_j}^{-1}{}^{w}\mathbf{T}_{p_i}} \\
&= \mathbf{J}_r^{-1}({}^{w}\mathbf{r}_{p_{ij}})\mathbf{Ad}^{-1}_{\mathrm{Exp}(\boldsymbol{\tau}_{p_{ij}})} = \mathbf{J}_l^{-1}({}^{w}\mathbf{r}_{p_{ij}}) \\
\mathbf{J}^{{}^{w}\mathbf{r}_{p_{ij}}}_{{}^{w}\mathbf{T}_{p_i}} &= \mathbf{J}^{{}^{w}\mathbf{r}_{p_{ij}}}_{\mathrm{Log}({}^{w}\mathbf{T}_{p_i}\boxminus{}^{w}\mathbf{T}_{p_j})}\mathbf{J}^{\mathrm{Log}({}^{w}\mathbf{T}_{p_i}-{}^{w}\mathbf{T}_{p_j})}_{{}^{w}\mathbf{T}_{p_i}\boxminus{}^{w}\mathbf{T}_{p_j}}\mathbf{J}^{{}^{w}\mathbf{T}_{p_i}\boxminus{}^{w}\mathbf{T}_{p_j}}_{{}^{w}\mathbf{T}_{p_i}} \\
&= -\mathbf{J}_r^{-1}(\boldsymbol{\tau}_{p_{ij}})
\end{aligned}$$

The Jacobians of the velocity constraint is calculated as,

$$\mathbf{J}^{{}^{w}\mathbf{r}_{v_i}}_{{}^{w}\mathbf{R}_b^i} = \mathbf{J}^{{}^{w}\mathbf{r}_{v_i}}_{{}^{w}\mathbf{R}_b^{i\,o}\mathbf{R}_b^{\mathrm{T}\,o}\tilde{\mathbf{v}}_m^i}\mathbf{J}^{{}^{w}\mathbf{R}_b^{i\,o}\mathbf{R}_b^{\mathrm{T}\,o}\tilde{\mathbf{v}}_m^i}_{{}^{w}\mathbf{R}_b^i} = {}^{w}\mathbf{R}_b^i\left\lfloor{}^{o}\tilde{\mathbf{v}}_m^i\right\rfloor_\times$$

B. The Relative Motion Noise

The relative motion noise is computed as,

$$\begin{aligned}
\Delta\mathbf{p}_{k,k+1} &= {}^{b}\mathbf{R}_w^k({}^{w}\mathbf{p}_o^{k+1} - {}^{w}\mathbf{p}_o^k) \\
&= \frac{1}{2}{}^{b}\mathbf{R}_w^k\left({}^{w}\hat{\mathbf{v}}_o^{k+1} + {}^{w}\hat{\mathbf{v}}_o^k\right)\Delta t \\
&= \frac{1}{2}{}^{b}\mathbf{R}_w^k\left({}^{w}\mathbf{R}_b^{k+1}({}^{o}\mathbf{v}_m^{k+1} - \mathbf{n}_v) + {}^{w}\mathbf{R}_b^k({}^{o}\mathbf{v}_m^k - \mathbf{n}_v)\right)\Delta t \\
&= \frac{1}{2}{}^{k}\mathbf{R}_{k+1}{}^{o}\mathbf{v}_m^{k+1}\Delta t + \frac{1}{2}{}^{o}\mathbf{v}_m^k\Delta t - \frac{1}{2}{}^{k}\mathbf{R}_{k+1}\mathbf{n}_v\Delta t - \frac{1}{2}\mathbf{n}_v\Delta t \\
&= \Delta\hat{\mathbf{p}}_{k,k+1} - \delta\mathbf{p}_{k,k+1}
\end{aligned}$$

REFERENCES

[1] G. Ullrich *et al.*, *Automated Guided Vehicle Systems*, vol. 10. Berlin, Germany: Springer-Verlag, 2015, p. 978.

[2] S. Zhao, H. Zhang, P. Wang, L. Nogueira, and S. Scherer, "Super odometry: IMU-centric LiDAR-Visual-Inertial estimator for challenging environments," 2021, *arXiv:2104.14938*.

[3] N. D. Van, M. Sualeh, D. Kim, and G.-W. Kim, "A hierarchical control system for autonomous driving towards urban challenges," *Appl. Sci.*, vol. 10, no. 10, p. 3543, May 2020.

[4] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to Autonomous Mobile Robots*. Cambridge, MA, USA: MIT Press, 2011.

[5] C. Cadena *et al.*, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Trans. Robot.*, vol. 32, no. 6, pp. 1309–1332, Dec. 2016.

[6] M. Tranzatto *et al.*, "CERBERUS in the DARPA subterranean challenge," *Sci. Robot.*, vol. 7, no. 66, May 2022, eabp9742.

[7] M. Tranzatto *et al.*, "CERBERUS: Autonomous legged and aerial robotic exploration in the tunnel and urban circuits of the DARPA subterranean challenge," 2022, *arXiv:2201.07067*.

[8] C. Chen, X. Lu, A. Markham, and N. Trigoni, "IONet: Learning to cure the curse of drift in inertial odometry," in *Proc. AAAI Conf. Artif. Intell.*, 2018, vol. 32, no. 1, pp. 1–9.

[9] S. Herath, H. Yan, and Y. Furukawa, "RoNIN: Robust neural inertial navigation in the wild: Benchmark, evaluations, & new methods," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2020, pp. 3146–3152.

[10] A. Reinke *et al.*, "LOCUS 2.0: Robust and computationally efficient lidar odometry for real-time underground 3D mapping," 2022, *arXiv:2205.11784*.

[11] D. Van Nam and K. Gon-Woo, "Deep learning based-state estimation for holonomic mobile robots using intrinsic sensors," in *Proc. 21st Int. Conf. Control, Autom. Syst. (ICCAS)*, Oct. 2021, pp. 12–16.

[12] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An open-source slam system for monocular, stereo, and RGB-D cameras," *IEEE Trans. Robot.*, vol. 33, no. 5, pp. 1255–1262, Oct. 2017.

[13] G. Huang, "Visual-inertial navigation: A concise review," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 9572–9582.

[14] N. V. Dinh and G.-W. Kim, "Multi-sensor fusion towards VINS: A concise tutorial, survey, framework and challenges," in *Proc. IEEE Int. Conf. Big Data Smart Comput. (BigComp)*, Feb. 2020, pp. 459–462.

[15] M. Sizintsev, A. Rajvanshi, H.-P. Chiu, K. Kaighn, S. Samarasekera, and D. P. Snyder, "Multi-sensor fusion for motion estimation in visually-degraded environments," in *Proc. IEEE Int. Symp. Saf., Secur., Rescue Robot. (SSRR)*, Sep. 2019, pp. 7–14.

[16] D. Wisth, M. Camurri, and M. Fallon, "VILENS: Visual, inertial, lidar, and leg odometry for all-terrain legged robots," 2021, *arXiv:2107.07243*.

[17] T. D. Barfoot, *State Estimation for Robotics*. Cambridge, U.K.: Cambridge Univ. Press, 2017.

[18] F. Dellaert and M. Kaess, "Factor graphs for robot perception," *Found. Trends Robot.*, vol. 6, nos. 1–2, pp. 1–139, 2017.

[19] M. A. Lee, B. Yi, R. Martin-Martin, S. Savarese, and J. Bohg, "Multimodal sensor fusion with differentiable filters," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 10444–10451.

[20] P. Geneva, K. Eckenhoff, W. Lee, Y. Yang, and G. Huang, "OpenVINS: A research platform for visual-inertial estimation," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 4666–4672.

[21] D. V. Nam and K. Gon-Woo, "Robust stereo visual inertial navigation system based on multi-stage outlier removal in dynamic environments," *Sensors*, vol. 20, no. 10, p. 2922, May 2020.

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

[23] V. Indelman, S. Williams, M. Kaess, and F. Dellaert, "Information fusion in navigation systems via factor graph based incremental smoothing," *Robot. Auto. Syst.*, vol. 61, no. 8, pp. 721–738, Aug. 2013.

[24] K. Li, M. Li, and U. D. Hanebeck, "Towards high-performance solid-state-LiDAR-inertial odometry and mapping," *IEEE Robot. Autom. Lett.*, vol. 6, no. 3, pp. 5167–5174, Jul. 2021.

[25] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus, "LIO-SAM: Tightly-coupled lidar inertial odometry via smoothing and mapping," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 5135–5142.

[26] T. Shan, B. Englot, C. Ratti, and D. Rus, "LVI-SAM: Tightly-coupled lidar-visual-inertial odometry via smoothing and mapping," 2021, *arXiv:2104.10831*.

[27] X. Zhao, C. Deng, X. Kong, J. Xu, and Y. Liu, "Learning to compensate for the drift and error of gyroscope in vehicle localization," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Oct. 2020, pp. 852–857.

[28] M. Brossard, A. Barrau, and S. Bonnabel, "AI-IMU dead-reckoning," *IEEE Trans. Intell. Vehicles*, vol. 5, no. 4, pp. 585–595, Dec. 2020.

[29] D. V. Nam and G.-W. Kim, "Online self-calibration of multiple 2D LiDARs using line features with fuzzy adaptive covariance," *IEEE Sensors J.*, vol. 21, no. 12, pp. 13714–13726, Jun. 2021.

[30] W. Liu *et al.*, "TLIO: Tight learned inertial odometry," *IEEE Robot. Autom. Lett.*, vol. 5, no. 4, pp. 5653–5660, Oct. 2020.

[31] M. Brossard, A. Barrau, and S. Bonnabel, "RINS-W: Robust inertial navigation system on wheels," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Nov. 2019, pp. 2068–2075.

[32] B. Yi, M. A. Lee, A. Kloss, R. Martín-Martín, and J. Bohg, "Differentiable factor graph optimization for learning smoothers," 2021, *arXiv:2105.08257*.

[33] T. Haarnoja, A. Ajay, S. Levine, and P. Abbeel, "Backprop KF: Learning discriminative deterministic state estimators," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 4376–4384.

[34] R. Jonschkowski, D. Rastogi, and O. Brock, "Differentiable particle filters: End-to-end learning with algorithmic priors," 2018, *arXiv:1805.11122*.

[35] C. Chen *et al.*, "Selective sensor fusion for neural visual-inertial odometry," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 10542–10551.

[36] J. Solà, J. Deray, and D. Atchuthan, "A micro lie theory for state estimation in robotics," 2018, *arXiv:1812.01537*.

[37] S. Boyd and L. Vandenberghe, *Introduction to Applied Linear Algebra: Vectors, Matrices, and Least Squares*. Cambridge, U.K.: Cambridge Univ. Press, 2018.

[38] C. M. Bishop and N. M. Nasrabadi, *Pattern Recognition and Machine Learning*, vol. 4, no. 4. New York, NY, USA: Springer, 2006.

[39] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration for real-time visual–inertial odometry," *IEEE Trans. Robot.*, vol. 33, no. 1, pp. 1–21, Feb. 2017.

[40] H. Taheri, B. Qiao, and N. Ghaeminezhad, "Kinematic model of a four Mecanum wheeled mobile robot," *Int. J. Comput. Appl.*, vol. 113, no. 3, pp. 6–9, Mar. 2015.

[41] P. Geneva, N. Merrill, Y. Yang, C. Chen, W. Lee, and G. Huang, "Versatile 3D multi-sensor fusion for lightweight 2D localization," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 4513–4520.

[42] M. Quan, S. Piao, M. Tan, and S.-S. Huang, "Tightly-coupled monocular visual-odometric SLAM using wheels and a MEMS gyroscope," *IEEE Access*, vol. 7, pp. 97374–97389, 2019.

[43] T. Moore and D. Stouch, "A generalized extended Kalman filter implementation for the robot operating system," in *Intelligent Autonomous Systems 13*, E. Menegatti, N. Michael, K. Berns, and H. Yamaguchi, Eds. Cham, Switzerland: Springer, 2016, pp. 335–348.

[44] M. Labbé and F. Michaud, "RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation," *J. Field Robot.*, vol. 36, no. 2, pp. 416–446, 2019.

[45] P. R. Kumar and P. Varaiya, *Stochastic Systems: Estimation, Identification, and Adaptive Control*. Philadelphia, PA, USA: SIAM, 2015.

[46] H. Akima, "A new method of interpolation and smooth curve fitting based on local procedures," *J. ACM*, vol. 17, no. 4, pp. 589–602, Oct. 1970.

[47] Y. Wang, H. Cheng, C. Wanga, and M. Q.-H. Meng, "Pose-invariant inertial odometry for pedestrian localization," *IEEE Trans. Instrum. Meas.*, vol. 70, 2021, Art. no. 8503512.

[48] L. Saini and M. Soni, "Artificial neural network based peak load forecasting using Levenberg–Marquardt and quasi-Newton methods," *IEE Proc.-Gener., Transmiss. Distrib.*, vol. 149, no. 5, pp. 578–584, 2002.

[49] F. Burden and D. Winkler, "Bayesian regularization of neural networks," in *Artificial Neural Networks*. 2008, pp. 23–42.

Dinh Van Nam received the B.Eng. degree in control and automation engineering from the Hanoi University of Science and Technology (HUST), Hanoi, Vietnam, in 2012, and the Ph.D. degree in control and robot engineering from Chungbuk National University, South Korea, in February 2022. Since 2022, he has been a Post-Doctoral Research Fellow with Chungbuk National University. He is a Lecturer with Vinh University, Vietnam. His research interests include AI-robotics, multi-sensor fusion for SLAM, optimization, and control systems.

Kim Gon-Woo (Member, IEEE) received the M.S. and Ph.D. degrees from Seoul National University, South Korea, in 2002 and 2006, respectively. He is currently a Professor with the Department of Electronics Engineering, Chungbuk National University, South Korea. His research interests include navigation, localization and SLAM for mobile robots, and autonomous vehicles.