Learning Type-2 Fuzzy Logic for Factor Graph Based-Robust Pose Estimation With Multi-Sensor Fusion

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

***Abstract*—Although a wide variety of high-performance state estimation techniques have been introduced recently, the robustness and extension to actual conditions of the estimation systems have been challenging. This paper presents a robust adaptive state estimation framework based on the Type-2 fuzzy inference system and factor graph optimization for autonomous mobile robots. We use the hybrid solution to connect the advantages of the tightly and loosely coupled technique by providing an inertial sensor and other extrinsic sensors such as LiDARs and cameras. In order to tackle the uncertainty input covariance and sensor failures problems, a learnable observation model is introduced by joining the Type-2 FIS and factor graph optimization. In particular, the use of Type-2 Takagi-Sugeno FIS can learn the uncertainty by using particle swarm optimization before adding the observation model to the factor graph. The proposed design consists of four parts: sensor odometry, up-sampling, FIS based-learning observation model, and factor graph-based smoothing. We evaluate our system by using a mobile robot platform equipped with a sensor setup of multiple stereo cameras, an IMU, and a LiDAR sensor. We imitate the LiDAR odometry in structure environments without needing other bulky motion capture systems to learn the observation model of the visual-inertial estimators. The experimental results are deployed in real-world environments to present the accuracy and robustness of the algorithm.**

***Index Terms*—Multi-sensor fusion, state estimation, learning fuzzy inference systems, factor graph optimization.**

I. Introduction

STATE estimation has been a core component of navigation and control systems for autonomous mobile robots and has been revolutionized by pioneering various technologies for many years [1], [2]. However, robust estimation solutions for self-driving cars, automated guided vehicles, or mining robots in GPS-denied environments have faced significant challenges such as darkness, dust, or fog scenarios [2], [3]. Extrinsic sensors such as LiDARs, cameras, and radars can perceive their surrounding environments as invariance landmarks to estimate the robot poses and map the environments. However, the LiDAR sensor cannot distinguish features in structureless environments like long straight corridors or tunnels. Moreover, the vision sensors are sensitive to light conditions and fail to track visual landmarks in texture-less scenarios such as insufficient illumination or changing light conditions [4], [5]. In contrast, the inertial sensors or wheel encoders with immunity against the surrounding conditions [3], [6]. Although the intrinsic sensors can work well in short-term navigation, intrinsic sensor-based state estimation drifted over time [1]. Therefore, incorporating multi-sensor data enables the estimation system to operate more accurately and robustly [2]. Many sensor fusion frameworks have recently made available open sources for the research community [7], [8], [9], [10], [11], [12].

Generally, state estimation can divide into two primary techniques: loosely coupled and tightly coupled fusion [13]. Regularly, the tightly coupled methods have shown advantages for accuracy, which scheme handles all sensor data in a central process [4]. In contrast, the loosely coupled technique independently deploys each sensor data to estimate the individual state. Then the predicted states are joined to get the best result [5]. Nevertheless, the tightly coupled systems have been challenging to integrate more sensor data and inherit from the different former methods, demanding more computational cost [11], [12]. Moreover, the tightly coupled techniques are sensitive to sensor failures because of running in one central process [6]. This method is also hard to compute separately in distributed computing systems [3], [11]. Compared to the tightly coupled method, the loosely coupled approaches allow more simplicity and extendibility. The methods can easily handle sensor failure by monitoring individual estimation engines [6]. To leverage the advantages of both techniques, we propose a hybrid-couple solution that handles the tightly coupled method for different sensor types. The estimated states are then combined using the loosely coupled design to get the optimal estimation [3].

Artificial intelligence (AI) in robotics has recently attracted many researchers [14]. However, AI solutions need to be tailored to meet the real-time performance of embedded computers. Recently, although deep learning has been a research trend topic, it usually needs much data for training and heavy

Manuscript received 14 September 2021; revised 6 August 2022 and 14 November 2022; accepted 29 December 2022. 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 through the Institute for Information & Communications Technology Planning & Evaluation (IITP) under Grant IITP-2023-2020-0-01462; and in part by the Korea Institute of Planning and Evaluation for Technology in Food, Agriculture, Forestry (IPET) through the Open Field Smart Agriculture Technology Short-Term Advancement Program funded by the Ministry of Agriculture, Food and Rural Affairs (MAFRA) under Grant 322039-3. The Associate Editor for this article was N. Bekiaris-Liberis. *(Corresponding author: Kim Gon-Woo.)*

Dinh Van Nam is with the 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 System and Robotics, Chungbuk National University, Cheongju-si 28644, South Korea (e-mail: gwkim@cbnu.ac.kr).

Digital Object Identifier 10.1109/TITS.2023.3234595

computational cost [14], [15]. In contrast, the fuzzy inference system (FIS) has effectively employed control systems to handle complex models for more than four decades [16]. FIS is also an intelligent technique with easy-to-learn parameters [17]. Consequently, we introduce an efficient learning method to handle observation model uncertainties and sensor failures in multi-sensor fusion problems. The proposed method deploys Type-2 TS-FIS to improve the observation model of sensor estimation before adding the information matrices to the factor graph [18].

This paper is organized as follows: after presenting the related works in Section II, Section III describes the proposed framework in detail, including problem formulation, system architecture, and training Type-2 FIS for adapting the observation model. Next, Section IV offers experimental results and comparison with state-of-the-art works, followed by a conclusion in Section V.

II. Related Works

State estimation for robotics by using multi-sensor fusion has been studied thoroughly in various literature [1], [2], [19], [20], [21]. Generally, the sensor fusion techniques apply the Bayes filtering inference that can be divided into two classes: filtering-based and optimization-based [4], [5], [19]. These fusion methods can further be categorized into loosely coupled and tightly coupled techniques [4], [5].

The filtering based-solution applies the Kalman filter (KF) and its variations, such as extended KF (EKF) and unscented KF (UKF) [20]. The filtering-based systems can quickly analyze the observability, which is one significant advantage [4], [22]. Lynen et al. introduced a loosely coupled EKF-based multi-sensor fusion framework to combine IMU, GPS, and camera [23]. This framework could manage sensor data with different frequencies and assist sensor outages. Shen et al. [24] performed a UKF-based odometry framework to handle the camera, LiDAR, and GPS sensor. The technique used global information and visual landmarks to update the state estimation. Although this technique could perform with high accuracy due to precision linearization, the computational cost was increased depending on the variable's size. A filtering-based ROS open-source enabled the estimation to fuse odometry data such as inertial sensor, GPS, and LiDAR sensor [25]. The ROS package could also decentralize the process with a weak, tightly coupled fusion. MaRS [26] implemented the recursive filtering strategy by updating sensor information sequentially. The tightly coupled approaches for the visual-inertial systems employed multi-state constraints EKF within a sliding window to achieve high accuracy with an efficient computation [9], [27]. Although the filtering solution could achieve high performance when providing global information, it could fail to the suboptimal state [28].

The optimization-based systems leverage all state information and sensor data to estimate the complete robot trajectory [4], [19]. The visual-inertial navigation system (VINS), such as VIN-Fusion [7] and GOMSF [29], was introduced with high performance using the factor graph optimization (FGO) in a sliding window [21]. Con-Fusion was an open-source for multi-sensor data fusion using FGO [8]. In 2021, LiLiOM [12] enabled the directly combining 3D solid-state LiDAR and IMU to provide real-time applications with superior accuracy. Tixiao et al. proposed the LVI-SAM, a tightly-coupled 3D LiDAR, camera, and IMU odometry method via smoothing and mapping [21]. David et al. suggested a unified multi-sensor fusion framework for tightly coupled LiDAR-visual-inertial odometry using point and line features [30]. The LIC-Fusion performed the sliding-window edge and plane features tracking using 3D Lidar-inertial-camera odometry [31], [32]. Nevertheless, the above estimation techniques used 3D LiDAR without considering the sensor's degradation and predicting observation model uncertainty.

Recently, AI-IMU dead-reckoning applied the convolutional neural network (CNN) to predict the noise model using the invariant EKF for IMU odometry [33]. The TLIO [34] was a tight learned inertial odometry, which applied deep learning to join the correction phase of the EKF directly. Sodhi et al. proposed a factor graph-based estimation for learning tactile models and ground encoding using CNN [35], [36]. Kloss et al. presented an excellent overview and showed how to train the differentiable filtering methods for state estimation [37]. A differentiable factor graph optimization was introduced using an end-to-end learning method [38]. This method uses backpropagation to adjust the system models using the CNN model. Nam et al. [18] used Type-2 FIS without training to solve the online calibration problem by adapting the sensor noise model in a factor graph. Recent work exploited factor graph optimization with the learning observation model using a neural network to adapt the covariance and model for intrinsic sensors [39]. In July 2022, Facebook released an exciting toolbox called Theseus for differentiable nonlinear optimization with custom layers written in Pytorch [40]. SymForce is a symbolic computation and code generation for Robotics developed and maintained by Skydio. The toolbox can auto-generated high-performance optimized code in C++ [41]. Theseus and SymForce can combine to become the perfect toolboxes that expect to help boost the application of estimation techniques in robotics.

SIMSF [42] proposed an optimization-based solution with fault detection using the Chi-square test. LOCUS [6] presented a simple solution with a non-smoothing switching estimation. An efficient lidar odometry technique for real-time subterranean 3D localization called LOCUS 2.0 has been released for heterogeneous sensors framework using the GICP formula [43]. In recent work, Super Odometry [3] introduced a hybrid fusion technique of multiple tightly-coupled odometry using an IMU-centric estimation. However, SIMSF, LOCUS, and Super Odometry techniques are not open sources. A tightly-coupled Sparse-Direct technique called FAST-LIVO using 3D LiDAR-Inertial-Visual Odometry [44]. FAST-LIVO is an open source that can provide good accuracy with a simple sensor degradation technique suitable for 3D solid-state LiDAR.

We believe that there are no solutions using intuitive and interpretable Type-2 Fuzzy logic to handle the uncertainty problem with sensor fusion of 2D LiDAR and more than one stereo camera fusion. We note that the earlier methods do not correct the noise model before registering to the factor graph using Type-2 TS FIS. Also, all the methods were just applied for 3D liDAR, while we used 2D LiDAR, which is very tough to perform the tight-coupled technique.

In contrast with previous works, we highlight our contributions as follows,

- We propose a novel learnable observation noise model for robust pose estimation by joining Type-2 TS FIS and factor graph optimization. The optimization guarantee of the learnable observation model is also discussed.
- We perform a system framework using the visual-LiDAR-inertial sensor based on a lightweight but practical Type-2 FIS to handle uncertainty sensor noise models and sensor failure problems with a highly accurate and low-latency estimation. The proposed system can achieve real-time performance and verify robustness in various perceptual degradation scenarios.

III. Methodology

This section describes the proposed multi-sensor fusion framework in detail. Firstly, we present several notations and problem formulation. Secondly, the techniques consisting of the IMU preintegration, up-sampling, and system architecture are described. Next, we exhibit the proposed system architecture using factor graph optimization. Finally, an adaptive observation noise model strategy based on learning Type-2 Takagi-Sugeno FIS is shown.

A. Notations

In this paper, the primary notations are defined as follows,

1) $(\mathbf{X}, \mathbf{Y}, \mathbf{Z}\ldots)$ denote matrices, $(\mathbf{x}, \mathbf{y}, \mathbf{z}\ldots)$ represent vectors, and lowercase italics $(x, y, z, \ldots)$ are scalars.
2) I denotes the IMU-frame that coincides with the body coordinate B, and O represents the odometry's initial coordinate. ${}^{O}\mathbf{T}_B$ is the transformation of frame B to frame O.
3) The p-norm of an n-vector $\mathbf{x} = (x_1, \ldots, x_n)$ is defined as, $\|\mathbf{x}\|_p = (\sum_{i=1}^{n} |x_i|^p)^{1/p}$, where $p \geq 1$ is a real number. $p = 2$ denotes the Euclidean norm, and $\|\mathbf{x}\|_\infty \triangleq \max_i |x_i|$ denotes the max norm.
4) Max-norm of matrix $\mathbf{X}_{n\times m}$ is defined by the maximum absolute column sum of $\mathbf{X}$ as $\|\mathbf{X}\|_\infty = \max_{1\leq j\leq n} \sum_{i=1}^{m} |x_{ij}|$.
5) $\|\mathbf{x}\|^2_{\Omega} = \mathbf{x}^{\mathrm{T}}\mathbf{\Omega}\mathbf{x}$ denotes the squared Mahalanobis distance.
6) $\mathbf{I}_n$ is an n-by-n identity matrix, $\mathbf{I}_n^{\alpha} \triangleq \alpha \times \mathbf{I}_n$ and $ae{-}\alpha \triangleq a \times 10^{-\alpha}$, where $a, \alpha \in \mathbb{R}$.
7) $\mathcal{M}$ denotes an n-manifold. A tangent space $\mathcal{T}_{\mathcal{M}}$ of $\mathcal{M}$ is locally homeomorphic to $\mathbb{R}^n$.
8) Capitalized Exp and Log are the exponential and logarithm maps between tangent space and manifold [45].
9) Lowercased exp and log present exponential and logarithm maps between lie algebra and manifold [45].
10) In this work, manifold $\mathcal{M}$ is a Lie group, then the local map can establish via operator ("oplus") $\oplus$ and operator ("ominus") $\ominus$ as,
$$\oplus : \mathcal{M} \times \mathbb{R}^n \to \mathcal{M} \Rightarrow \mathbf{x} \oplus \mathbf{u} = \mathbf{x} \circ \mathrm{Exp}(\mathbf{u})$$
$$\ominus : \mathcal{M} \times \mathcal{M} \to \mathbb{R}^n \Rightarrow \mathbf{y} \ominus \mathbf{x} = \mathrm{Log}(\mathbf{x}^{-1} \circ \mathbf{y}), \quad (1)$$
where $\circ$ is a composition operation [45].
11) $\mathbb{SO}(3)$ is the Special Orthogonal Group,
$$\mathbb{SO}(3) = \left\{\mathbf{C} \in \mathbb{R}^{3\times 3} |\det(\mathbf{C}) = 1, \mathbf{C}^{\mathrm{T}}\mathbf{C} = \mathbf{I}\right\}.$$
12) $\mathbb{SE}(3)$ is the Special Euclidean Group,
$$\mathbb{SE}(3) = \left\{\begin{bmatrix} \mathbf{C} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \in \mathbb{R}^{4\times 4} | \mathbf{C} \in \mathbb{SO}(3), \mathbf{t} \in \mathbb{R}^3\right\}.$$
13) The Lie-algebra of $\mathbb{SO}(3)$ and $\mathbb{SE}(3)$ are $\mathfrak{so}(3)$ and $\mathfrak{se}(3)$, respectively [45].
14) $[\boldsymbol{\theta}]_\times$ is a skew-symmetric matrix of vector $\boldsymbol{\theta}$ [45], given as
$$\lfloor \boldsymbol{\omega} \rfloor_\times = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}.$$
15) A similar definition for $\mathbb{SO}(2)$ and $\mathbb{SE}(2)$ Lie group, for more detail on $\mathbb{SO}(n)$, $\mathbb{SE}(n)|_{n=2,3}$, visit [19], [45].

B. Problem Formulation

Let us define the robot state at time step t_k embedded in a vector $\mathbf{x}_k$ on 15-dimensional manifold $\mathcal{M}$ as,
$$\mathbf{x}_k = \left(\mathbf{R}_k,\ \mathbf{p}_k,\ \mathbf{v}_k,\ \mathbf{b}_k^a,\ \mathbf{b}_k^\omega\right), \quad (2)$$
where $\mathbf{R}_k \in \mathbb{SO}(3)$, $\mathbf{p}_k$, $\mathbf{v}_k \in \mathbb{R}^3$ are the robot's orientation, position, and velocity with respect to navigation coordinate O, respectively; $\mathbf{b}_k^a$, $\mathbf{b}_k^\omega \in \mathbb{R}^3$ are the accelerometer and gyroscope biases [4]. Here, the scope of the proposed sensor setup uses two stereo cameras, a LiDAR, and an IMU. We assume that all extrinsic calibration parameters of each sensor to the body frame B are known precisely. The LiDAR sensor can provide the relationship between two consequence poses [46]. Moreover, by using the stereo-visual-inertial odometry techniques [9], [27], both stereo cameras can also compute the relative motion between two consecutive poses.

Hence, the stereo-LiDAR-inertial fusion problem is written as a nonlinear least-square optimization with a cost function given as [19],
$$J(\mathbf{x}) = \sum_{m,n\in C} \|e_r(\mathbf{x}_m, \mathbf{x}_n)\|^2_{\Omega_r} + \sum_{k\in W} \left\|\mathbf{r}_I^k\right\|^2_{\Omega_I} + \|e_M(\mathbf{x}_s)\|^2_{\Omega_s} + \sum_{p\in P} \left\|e_p^k(\mathbf{x}_p)\right\|^2_{\Omega_p} \quad (3)$$
where $e_r(\mathbf{x}_m, \mathbf{x}_n)$ and $\mathbf{\Omega}_r$ are the residual and information matrix of the relative motion between pose m and pose n; $\mathcal{C}$ is a set of relative motion factors in the sliding window $\mathcal{W}$ and is limited to a fixed number for efficient computing called fixed-lag smoothing solution [21]; $\mathbf{r}_I^k$ and $\mathbf{\Omega}_I$ are the residuals and information matrix of the IMU that will be discussed in the next Subsection; $e_M(\mathbf{x}_s)$ is the marginalization residual [8], [47] that uses to limit the size of sliding window and maintain the information of the first pose $\mathbf{x}_s$ in the sliding window $\mathcal{W}$; $e_p^k(\mathbf{x}_p)$ and $\mathbf{\Omega}_p$ are the prior factor and information matrix of pose k in prior space $\mathcal{P}$. The prior

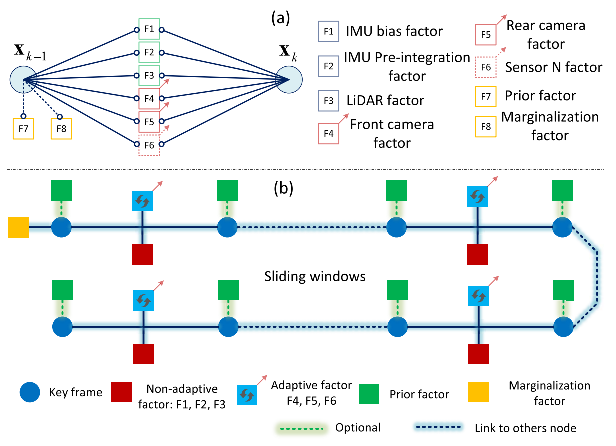

Fig. 1. An illustration of the factor graph framework pipeline. The factors are divided into adaptive and non-adaptive factors. We can integrate another sensor like radar (sensor N) into the framework. (a)- represents the factor graph between two consecutive poses. (b)- describes the factor graph in a sliding window.

factors are added, indicating the prior information of the robot poses to deal with the general problem. Usually, in this case, we add a prior factor for the initial pose. However, suppose we know prior pose information, such as the visual fiducial system AprilTag [48], during robot operation. In that case, that can be added directly to the factor graph. Equation 3 indicates a factor graph optimization problem [21] where the constrained variables are described as the factors above.

We note that the covariance matrices above are usually constant that possess several limits to achieve robust performance. This paper brings an AI approach to update these covariance matrices. We design a factor graph architecture to model the optimization problem (3), as shown in Fig. 1. We note that the belief of the observation model is denoted by the inverse of the covariance matrix of each sensor model. A factor with a smaller covariance matrix contributes more influence to the optimal solution [49]. Here, the results of the MAP problem are strongly affected by the belief in the observation model [19].

C. IMU Preintegration and Upsampling Estimation

We show how to compute inertial residual $\mathbf{e}_I^k$ (3). The measurement outputs of an IMU are given as [4],

$$\begin{aligned}\tilde{\boldsymbol{\omega}} &= \boldsymbol{\omega} + \mathbf{b}_g + \mathbf{n}_g\\ \tilde{\mathbf{a}} &= \mathbf{R}({}^{w}\mathbf{a} - \mathbf{g}) + \mathbf{b}_a + \mathbf{n}_a\end{aligned} \tag{4}$$

where $\tilde{\boldsymbol{\omega}}$ and $\boldsymbol{\omega}$ are the measured values and actual output of the instantaneous angular velocity expressed in the body frame, respectively; $\tilde{\mathbf{a}}$ is the measurement acceleration in the body frame, and ${}^{w}\mathbf{a}$ is the actual acceleration in world coordinate; $\mathbf{R}$ is the rotation of the body frame to the world frame; $\mathbf{g}$ is the gravity vector in the world coordinate; $\mathbf{b}_g$ is angular velocity bias and $\mathbf{b}_a$ is the accelerator bias; $\mathbf{n}_g$ and $\mathbf{n}_a$ are the zero-mean Gaussian noises of the angular velocity and acceleration, respectively. The continuous differential equations in the world coordinate are computed as [9],

$$\begin{aligned}\dot{\mathbf{R}}(t) &= \mathbf{R}(t)\lfloor \boldsymbol{\omega}\Delta t\rfloor_{\times}\\ \dot{\mathbf{p}}(t) &= \mathbf{v}(t)\\ \dot{\mathbf{v}}(t) &= {}^{w}\mathbf{a}(t) = \mathbf{g} + \mathbf{R}^{\mathrm{T}}(t)\left(\tilde{\mathbf{a}}(t) - \mathbf{b}_a\right)\\ \dot{\mathbf{b}}_a(t) &= \mathbf{0},\ \dot{\mathbf{b}}_g(t) = \mathbf{0}\end{aligned} \tag{5}$$

To avoid re-computing the inertial propagation over time (5), the IMU preintegration enables a high-frequency propagation process with an efficient computational cost [50]. The IMU factor (3) predicts the navigation state $\mathbf{x}_n$ from the current state $\mathbf{x}_m$ given a batch of IMU measurements. The key idea is that the velocity and position are separated into the gravity and acceleration components [50]. Here, the preintegrated measurements are turned into pseudo-measurements of position, velocity, and rotation. The preintegration is computed up to time Δt given as,

$$\mathbf{r}_{\mathcal{I}_{mn}} = \left[\mathbf{r}_{\gamma_{mn}}^{\mathrm{T}},\ \mathbf{r}_{\beta_{mn}}^{\mathrm{T}},\ \mathbf{r}_{\alpha_{mn}}^{\mathrm{T}}\right], \tag{6}$$

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

$$\begin{aligned}\mathbf{r}_{\alpha_{mn}} &= \mathbf{R}_m^{\mathrm{T}}(\mathbf{p}_n - \mathbf{p}_m - \mathbf{v}_m\Delta t - \frac{1}{2}\mathbf{g}\Delta t^2)\\ &\quad -\Delta\tilde{\mathbf{p}}_{mn}(\mathbf{b}_{g_m},\ \mathbf{b}_{a_m}),\\ \mathbf{r}_{\beta_{mn}} &= \mathbf{R}_m^{\mathrm{T}}(\mathbf{v}_n - \mathbf{v}_m - \mathbf{g}\Delta t) - \Delta\tilde{\mathbf{v}}_{mn}(\mathbf{b}_{g_m},\ \mathbf{b}_{a_m}),\\ \mathbf{r}_{\gamma_{mn}} &= \mathrm{Log}(\Delta\tilde{\mathbf{R}}_{mn}(\mathbf{b}_{g_m}))\mathbf{R}_m^{\mathrm{T}}\mathbf{R}_n,\end{aligned}$$

where $\Delta t = t_n - t_m$. The pseudo-measurements of rotation $\Delta\tilde{\mathbf{R}}_{mn}$, velocity $\Delta\tilde{\mathbf{v}}_{mn}$ and position $\Delta\tilde{\mathbf{p}}_{mn}$ are implemented as follows [50],

$$\begin{aligned}\Delta\tilde{\mathbf{R}}_{mn} &= \prod_{k=m}^{n-1}\mathrm{Exp}((\tilde{\boldsymbol{\omega}}_k - \mathbf{b}_m^g)\Delta t)\\ \Delta\tilde{\mathbf{v}}_{mn} &= \sum_{k=m}^{n-1}\Delta\tilde{\mathbf{R}}_{mk}(\tilde{\mathbf{a}}_k - \mathbf{b}_m^a)\Delta t\\ \Delta\tilde{\mathbf{p}}_{mn} &= \sum_{k=m}^{n-1}\left[\tilde{\mathbf{v}}_{mk}\Delta t + \frac{1}{2}\Delta\tilde{\mathbf{R}}_{mk}(\tilde{\mathbf{a}}_k - \mathbf{b}_m^a)\Delta t^2\right].\end{aligned} \tag{7}$$

The preintegration noise can be iteratively propagated as reported in [50].

We note that each sensor estimation operates at a different frequency in practice, such as camera odometry at 30 Hz and LiDAR odometry at 15 Hz. So, a high-speed IMU with a frequency of 400 Hz is used to synchronize all the estimations. An overview of the up-sampling process is represented in Fig. 2. Using the discrete IMU kinetic model (5), robot state $\mathcal{Y}$ to world coordinate can be upsampled to IMU frequency as,

$$\mathcal{Y}(t) = \begin{pmatrix}\mathbf{R}^{i+1}\\ \mathbf{v}^{i+1}\\ \mathbf{p}^{i+1}\end{pmatrix} = \begin{pmatrix}\mathbf{R}^i\exp(\lfloor\boldsymbol{\omega}\Delta t\rfloor_{\times})\\ \mathbf{v}^i + {}^{w}\mathbf{a}^i\Delta t\\ \mathbf{p}^i + \mathbf{v}^i\Delta t + \frac{1}{2}{}^{w}\mathbf{a}^i\Delta t^2\end{pmatrix}, \tag{8}$$

where $\Delta t = t_{i+1} - t_i$, ${}^{w}\mathbf{a}^i = \mathbf{R}^{i\mathrm{T}}\left(\tilde{\mathbf{a}}^i - \mathbf{b}_a^i\right) + \mathbf{g}$. The preintegration is updated to compensate for the bias velocity and rotated gravity vector whenever other sensor data is obtained [47].

To sum up, Algorithm 1 is implemented to perform the up-sampling process.

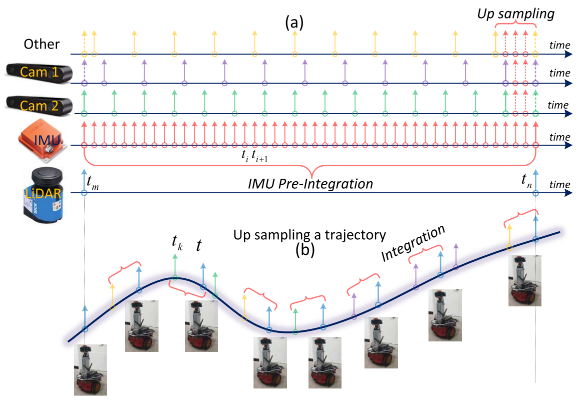

Fig. 2. The up-sampling and synchronization process using multi-stereo cameras, LiDAR, IMU. (a)- represents the time clock of each sensor. (b)- illustrates the upsampling process of the robot trajectory using IMU.

```
Algorithm 1 Preintegration and Upsampling
   Input: IMU data, estimation observing at time $t_k$
   Output: estimation $\mathcal{Y}_t$ at time $t$
 1 Initialization preintegration $\mathcal{P}$;
 2 if new IMU data come then
 3     Integrated preintegration $\mathcal{P}$ at $t$ (7);
 4 end
 5 if new estimation observed then
 6     Update bias of preintegration $\mathcal{P}$ at $t_k$ [47];
 7 end
 8 Generate estimation $\mathcal{Y}_t$ at time $t$ (8)
```

Remark 1: In this work, the adaptive covariance Σ_a in which its format is diagonal matrix as,

$$\Sigma_a = diag(\sigma_1, \sigma_2, \ldots, \sigma_n), \tag{9}$$

where $0 < \sigma_1, \sigma_2, \ldots, \sigma_n < \xi$ are scalars that are learned and bounded by AI technique. So, the covariances of non-adaptive and adaptive factors are symmetric positive definite matrix and bounded. Therefore, given the estimation variable $\hat{\mathbf{x}}$, the optimization problem is convergence using the Levenberg–Marquardt algorithm toward robustness [19].

D. A Simulation for Adaptive Factor Graph Optimization

Let us provide an example of state estimation using a factor graph with an adaptive element, as shown in Fig. 3-a. A robot moves on a planner over five steps from pose $\mathbf{x}_1$ to pose $\mathbf{x}_5$. Each robot pose is represented on $\mathbb{SE}(2)$ given as,

$$\mathbf{x}_i = \begin{bmatrix} \mathbf{R}_i & \mathbf{t}_i \\ \mathbf{0} & 1 \end{bmatrix} \in \mathbb{SE}(2). \tag{10}$$

The relative motion between two consecutive poses is computed as follows,

$$\mathbf{r}_i = \mathrm{Log}(\mathbf{x}_{i+1} \ominus \mathbf{x}_i) - \mathbf{u}_i, \tag{11}$$

where $\mathbf{u}_i = \begin{bmatrix} \delta x \\ \delta y \\ \delta\theta \end{bmatrix} + \begin{bmatrix} n_x \\ n_y \\ n_\theta \end{bmatrix}$ is the odometry on $\mathfrak{se}(2)$ of x, y and θ; $n_x \sim \mathcal{N}(0, \sigma_x)$, $n_y \sim \mathcal{N}(0, \sigma_y)$ and $n_\theta \sim \mathcal{N}(0, \sigma_\theta)$

TABLE I
THE PARAMETERS OF THE FACTOR GRAPH

Factor	Node From	Node To	Relative Motion	Noise Model
F1	x1	x2	(2, 1, 0)	(0.1, 0.1, 0.1)
F2	x2	x3	(2, 0, pi/2)	(0.1, 0.1, 0.1)
F3	x3	x4	(2, 0, pi/2)	(0.1, 0.1, 0.1)
F4	x4	x5	(2, 0, pi/2)	(0.1, 0.1, 0.1)
F5	x1	x2	(2.5, 1.5, 0.2)	$(\sigma_x, \sigma_y, \sigma_\theta)$
F6	Prior	x1	(0, 0, 0)	(0.3, 0.3, 0.1)

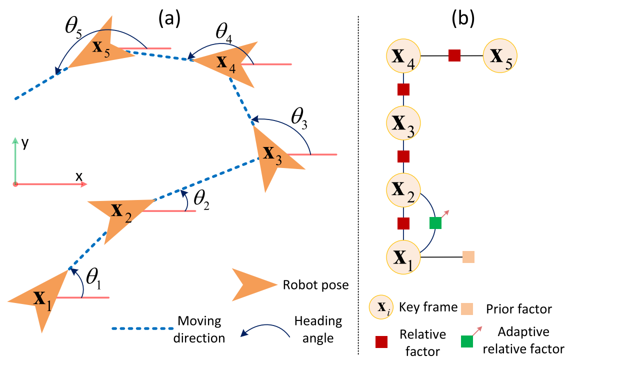

Fig. 3. The experiment setup for the simulation. (a)-the visualization of the robot's trajectory. (b)-the factor graph represents the robot motion as shown in (a).

are the uncertainties modelled by zero mean Gaussian noise. To tackle the state estimation problem, we build a factor graph to solve the state estimation problem, as shown in Fig. 3-b. All parameters of the relative odometry factors are presented in Table I. Here, the relative factors F1, F2, F3, F4, and prior factor F6 are indicated with constant parameters. Only the noise model of factor F5 is manually adjusted as follows,

$$1e-3 \leq \sigma_x \leq 1, \quad 2e-3 \leq \sigma_y \leq 2, \quad 1e-2 \leq \sigma_\theta \leq 2. \tag{12}$$

We note that factor F5 is added to mimic the uncertainty information of the relative odometry from pose 1 to pose 2.

Next, the Levenberg–Marquardt algorithm is applied to solve the optimization problem. The results of the factor graph optimization are illustrated as shown in Fig. 4. The estimated trajectories show that the uncertainty model of factor F5 decided the accuracy of the state estimation. In particular, the results are better when the covariance values of factor F5 have increased, as shown in Fig. 4-a. Moreover, the variation of the relative odometry of factor F5 is present as in Fig. 4-b. The above results confirmed that increasing the values $(\sigma_x, \sigma_y, \sigma_\theta)$ produced better estimation outcomes.

We also increase the value of $(\sigma_x, \sigma_y, \sigma_\theta)$ to 1000, the estimated trajectory almost coincides with the ground truth, which validated Remark 1.

E. System Architecture

The proposed system is designed as shown in Fig. 5, which consists of four modules: sensor estimation, up-sampling and

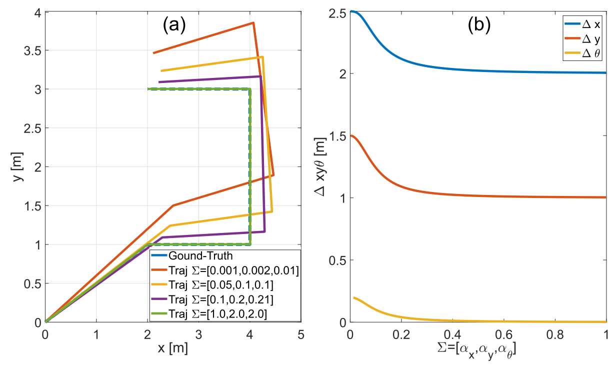

Fig. 4. The simulation results are illustrated when the covariance is manually changed. (a)- estimated robot states corresponding with each covariance matrix. (b)- The relation of the covariance scale factor to the difference between end-pose estimated odometry with ground truth.

synchronization, Type-2 FIS covariance prediction, and local factor graph.

1) Sensor Estimators: The sensor estimators can leverage various fusion approaches to compute the robot pose. In particular, the sensor setup is equipped with two stereo cameras, a 9-DoF IMU, and a LiDAR sensor rigidly attached to the robot frame, as shown in Fig. 8. This sensor system can effortlessly add more external sensors such as wheel encoders and radars.

Firstly, the point cloud data generated from the LiDAR sensor is filtered by downsampling and normalizing [46]. Then, the geometric registration utilizes S2M of the new point cloud to the local point cloud submap [10], [46]. The registration process then provides the transformation ${}^{O}\mathbf{T}_L$ by solving a nonlinear least-square optimization problem [46],

$$ {}^{O}\mathbf{T}_L^{*} = \underset{{}^{O}\mathbf{T}_L}{\arg\min} \left(e \left({}^{O}\mathbf{T}_L(\mathcal{P}_L), \mathcal{P}_O \right) \right), \tag{13} $$

where $e(.)$ denotes the error function calculated by comparing the transformed reading point cloud $\mathcal{P}_L$ to the local map point cloud $\mathcal{P}_O$. The point cloud registration is done using the iterative closest point (ICP) method [46]. To overcome human-made environments with various plane structures, the point-to-plane minimizer with IMU prediction search range [51] is applied to solve Eq. 13. Here, LiDAR-IMU fusion provides the robot poses from the laser frame L to Odom frame O. Therefore, the trajectory information provided by LiDAR-IMU needs to be transformed from frame B to frame O as follows,

$$ {}^{O}\mathcal{Y}_B^{L}(t_k) = {}^{O}\mathbf{T}_L(t_k) {}^{B}\mathbf{T}_L^{-1} \in \mathbb{SE}(3), \quad \mathbf{\Omega}_L \in \mathbb{R}^{6\times 6}, \tag{14} $$

where t_k is the time step obtained by LiDAR measurement, and $\mathbf{\Omega}_L$ is the information matrix of LiDAR odometry.

Secondly, the VINS using a stereo camera and IMU is handled to compute the state estimation. The proposed system with two stereo cameras employs two VINS estimations. We note that all open sources based on VINS methods [7], [9], [52] fail when the camera loses track and cannot recover. In contrast, Zed-VINS[1] provides high accuracy and can recover after failures. Nevertheless, the Zed VINS covariance matrix cannot reflect correct information in several cases like lost feature tracking. Therefore, we provide an AI technique to solve the uncertainty problem for the system. The Zed VINS estimations of both stereo cameras represent their odometry information in frame B as,

$$ {}^{O}\mathcal{Y}_B^{C1}(t_m) = {}^{O}\mathbf{T}_{C1}(t_m) {}^{B}\mathbf{T}_{C1}^{-1} \in \mathbb{SE}(3), \mathbf{\Omega}_{C1} \in \mathbb{R}^{6\times 6}, \tag{15} $$

$$ {}^{O}\mathcal{Y}_B^{C2}(t_p) = {}^{O}\mathbf{T}_{C2}(t_p) {}^{B}\mathbf{T}_{C2}^{-1} \in \mathbb{SE}(3), \mathbf{\Omega}_{C2} \in \mathbb{R}^{6\times 6}, \tag{16} $$

where t_m and t_p are the time step observed by the front and rear camera, respectively; $\mathbf{\Omega}_{C1}$ and $\mathbf{\Omega}_{C2}$ are the front and rear camera odometry information matrix, respectively. A similar process is deployed if we have extra other sensors.

2) Up-Sampling and Synchronization: We note that each sensor has a different frequency. So, the sensor estimators provide their trajectory information at different times. Herein, trajectories generated from LiDAR ${}^{O}\mathcal{Y}_B^{L}(t_k)$ (14), front camera ${}^{O}\mathcal{Y}_B^{C1}(t_m)$ (15) and rear camera ${}^{O}\mathcal{Y}_B^{C2}(t_p)$ (16) are obtained at different time steps at t_k, t_m, and t_p, respectively. The sensor estimations can obtain at the IMU time step by using the up-sampling technique in Subsection III-C. In practice, LiDAR frequency is smaller than the camera frequency. Therefore, the LiDAR clock is utilized to create the triggers for the synchronization of all sensor estimators. Finally, the estimation information of the front camera ${}^{B}\mathcal{Y}_{C1}(t_k)$ and rear camera ${}^{B}\mathcal{Y}_{C2}(t_k)$ are simultaneously observable at time t_k.

3) Local Factor Graph Optimization: Once all sensor estimators are synchronized, their residual is added to the factor graph (3). Whenever a laser estimation is computed, its observed time step is selected as a local keyframe [47]. The factors are divided into two categories: adaptive and non-adaptive factors, as shown in Fig. 1. The non-adaptive factor consists of F1, F2, F7, and F8, as shown in Fig. 1. Factor F1 is the IMU bias factor that remains a constant value between two keyframes. Factor F2 denotes the IMU preintegration as present in Subsection III-C. Note that Factor F1 and F2 are combined to correct the bias error of the IMU preintegration. Factor F7 is optional if proving prior information of a keyframe such as GPS [42] and UWB [53]. Factor F8 is the marginalization factor deployed in the front of the sliding window described in Subsection III-B [7], [8]. The adaptive factors, including LiDAR factor F3, front camera factor F4 and rear camera factor F5, will be updated using Type-2 FIS, as shown in Subsection III-F. Finally, the cost function $J(\mathbf{x})$ (3) of the factor graph is solved to find the optimal states.

F. Type-2 TS FIS for Observation Noise Model

There are three FIS types: Mamdani, Takagi-Sugeno (TS), and singleton-type. Mamdani FIS uses IF-THEN rules registered by linguistic variables [17]. Although Mamdani FIS is an intuitive and interpretable method, TS FIS is more computationally efficient and suitable for optimization and adaptive solutions [16], [17]. Besides, the TS type can quickly learn the rules and guarantee the output surface's continuity [16]. In particular, the Type-2 membership functions can handle the uncertainty problem of the inputs [17]. Furthermore, Type-2 TS-FIS is an efficient solution for uncertainty data, with impressive results in the control system [16].

[1]https://github.com/stereolabs/zed-ros-wrapper

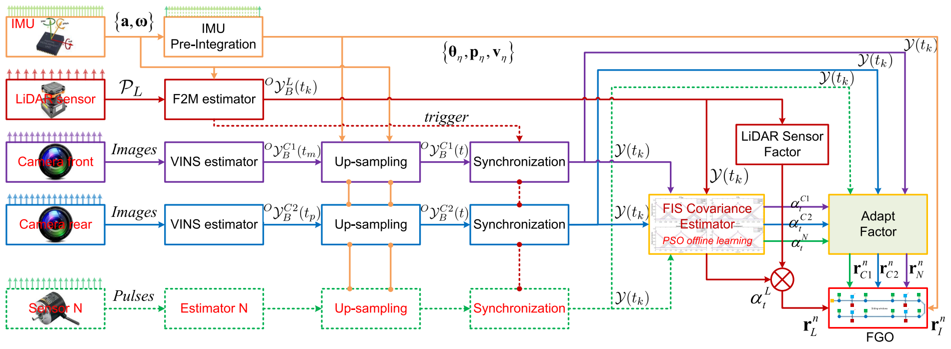

Fig. 5. The pipeline of the proposed system architecture is equipped with a LiDAR sensor, two Zed 2 cameras, an IMU sensor, and an option using wheel encoders. LiDAR odometry uses the scan-to-map (S2M) method [46]; The camera estimation leverages the visual-inertial navigation system (VINS). These estimators are up-sampled and synchronized, then added to a factor graph with FIS-updated covariance.

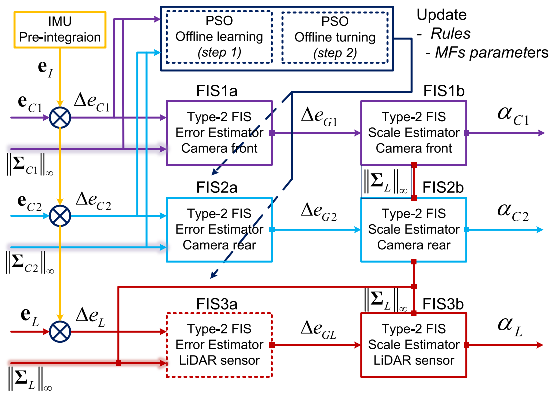
IMU Pre-integraion
PSO Offline learning (step 1)
PSO Offline turning (step 2)
Update
- Rules
- MFs parameters
FIS1a
Type-2 FIS Error Estimator Camera front
FIS1b
Type-2 FIS Scale Estimator Camera front
FIS2a
Type-2 FIS Error Estimator Camera rear
FIS2b
Type-2 FIS Scale Estimator Camera rear
FIS3a
Type-2 FIS Error Estimator LiDAR sensor
FIS3b
Type-2 FIS Scale Estimator LiDAR sensor

Fig. 6. The proposed architecture using Type-2 FIS. The inputs are the error of estimations and IMU preintegration and the norm of covariance matrices. First, block FIS1a, FIS2a and FIS3a use to estimate the error to ground truth. Then, block FIS1b, FIS2b, and FIS3b are performed to calculate the covariance scales, which are optional depending on the experimental system.

We design a FIS framework for adapting noise parameters, as shown in Fig. 6. Here, block FIS1a, FIS2a and FIS3a are Type-2 TS FIS are utilized to predict the error of the front camera, rear camera, and LiDAR sensor to the ground truth. Block FIS1a is defined as following TS fuzzy model [16], [17],

$$\begin{aligned}\text{RULE } \mathcal{R}_i \ &: \ \text{IF } \Delta e_{C1} \text{ is } \mathcal{S}_i^{C1} \text{ and } \|\Sigma_{C1}\|_\infty \text{is } \mathcal{S}_i^{\Sigma}\\ \text{THEN } \Delta e_{G1} &= a_i \Delta e_{C1} + b_i \|\Sigma_{C1}\|_\infty + c_i,\end{aligned}$$

where $\mathcal{R}_i$ represents the ith rule of FIS1a; $\|\Sigma_{C1}\|_\infty$ is the max-norm of front camera covariance matrix; $\Delta \mathbf{e}_{C1} = \|\text{Log}(\mathbf{e}_I \ominus \mathbf{e}_{C1})\|_2$; $\mathbf{e}_I$ is the prediction using IMU preintegration, $\mathbf{e}_{C1}$ is the front camera estimation; Δe_{G1} is the prediction error of the front VINS-Zed to ground truth. $\mathcal{S}_i^C$ and $\mathcal{S}_i^\Sigma$ are the fuzzy sets to handle the error of the front camera estimation and its covariance matrix [16], respectively. Block FIS2a and FIS3a are designed for the rear camera and LiDAR error estimation similar to block FIS1a.

Next, block FIS1b, FIS2b, and FIS3b are deployed to predict the scale coefficients of the covariance matrix of the front camera, rear camera, and LiDAR sensor. Block FIS1b is

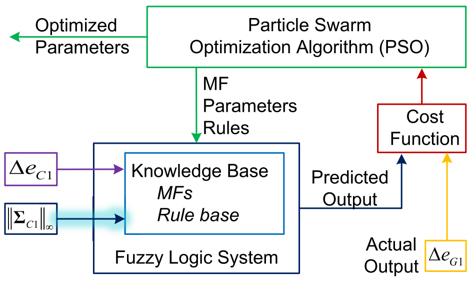

Fig. 7. The training process of Type-2 FIS for FIS1a. The FIS first learned the rule, then tuned the MF parameter of the input and output. The PSO algorithm can learn the FIS parameter during training using the input and output datasets.

designed as,

$$\begin{aligned}\text{RULE } \mathcal{R}_i \ &: \ \text{IF } \Delta e_{G1} \text{ is } \mathcal{S}_i^{G1} \text{ and } \|\Sigma_L\|_\infty \text{is } \mathcal{S}_i^{L}\\ \text{THEN } \alpha_{C1} &= a_i \Delta e_{G1} + b_i \|\Sigma_L\|_\infty + c_i,\end{aligned}$$

where Δe_{G1} is the output of FIS1a, α_{C1} is a coefficient of the covariance matrix of the front camera factor F4 in the factor graph as, $\Sigma_{C1}^{\text{F4}} \triangleq \alpha_{C1}\mathbf{I}_6^{1e-6}$. Block FIS2b and block FIS3b are designed like block FIS1a. α_{C2} and α_L are the coefficients of the rear camera factor F5 and LiDAR factor F3 as, $\Sigma_L^{\text{F3}} \triangleq \alpha_L\mathbf{I}_6^{1e-6}$; $\Sigma_{C2}^{\text{F5}} \triangleq \alpha_{C2}\mathbf{I}_6^{1e-6}$. $\mathcal{S}_i^{G1}$ and $\mathcal{S}_i^{L}$ are the fuzzy sets of Δe_{G1} and $\|\Sigma_L\|_\infty$, respectively.

An overview of the training process for FIS is shown in Fig. 7. Since we have input/output training data, FIS1a is trained following three steps,

- Step 1 (*learning*): keep the input MF $\mathcal{S}_i^{C1}$, $\|\Sigma_{C1}\|_\infty$ and output MF, and learn rules $\mathcal{R}_i$ of Type-2 TS fuzzy logic.
- Step 2 (*turning*): retain the learned rules $\mathcal{R}_i$ and lower parameters of input MF $\mathcal{S}_i^{C1}$, and turn the output MF and upper parameters of input MF $\mathcal{S}_i^{C1}$.
- Step 3 (*turning*): hold the rules $\mathcal{R}_i$, output MF and upper parameters of input MF $\mathcal{S}_i^{C1}$, and turn the lower parameters of the input MF $\mathcal{S}_i^{C1}$.

Many global optimization solutions are inspired by wildlife, such as flocks of birds or insects swarming. The genetic algorithms and particle swarm optimization perform better in dealing with the large ranges parameter of FIS. Similar to the

Algorithm 2 Multi-Sensor Fusion Algorithm

```
Input: Heterogeneous sensor data
Output: State estimation at time $t$
1  Init the odometry system: IMU, cameras, LiDAR;
2  Init the front and rear camera up-sampling (Algorithm 1);
3  while LiDAR estimation obtained do
4      Synchronization of all sensor estimators ;
5      Feed all estimation information to FIS (III-F);
6      Add factor with adaptive covariance matrices;
7      Solve the graph optimization (Fig. 1);
8      Update bias for IMU preintegration;
9      if size of the graph is maximum then
10         Reset the factor graph;
11         Update the marginalization factor;
12     end
13 end
14 while IMU data observed do
15     Add to global buffer;
16     Integration and generate estimation $\mathcal{Y}_t$;
17 end
```

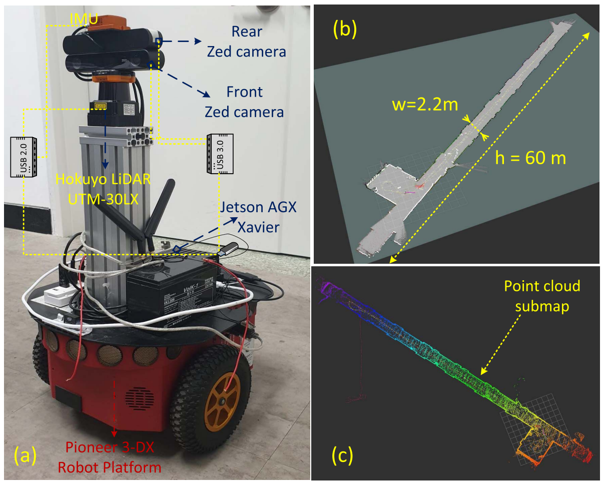

Fig. 8. (a)-The experimental structure consists of LiDAR, two Zed 2 cameras, and an IMU that is connected to an embedded computer-Jetson NVIDIA Xavier by USB wire; (b)-The map environment is to evaluate the investigations; (c)- The local point cloud submap for S2M method with LiDAR sensor.

genetic algorithm, the Particle swarm is a population-based technique in which a set of particles follow in steps throughout a search area [54]. The objective function evaluates the process at each step to determine the best velocity of each particle. After updating the new particle position, the algorithm will be re-evaluated. We perform the above process until the algorithm achieves a stopping criterion. Here, the particle swarm optimization (PSO) algorithm with the MATLAB-Global Optimization Toolbox[2] is used to learn the rules and tune the membership functions (MFs) of the FIS system. FIS2a is trained similarly to FIS1a. The training dataset is collected in a realistic environment, then offline training the FIS structure on a Desktop computer. Finally, we deploy the FIS framework in C++ embedded in the whole system, as shown in Fig. 5.

IV. Experiments

This section describes the implementation and evaluation of the proposed system in sensor failure scenarios and uncertainty input covariance.

A. Implementation and Experiment Setup

We implemented Algorithm 2 to run the proposed system with four parallel threads running on the CPU resources. The datasets were collected with a sensor system, as shown in Fig. 8.

We built a sensor setup consisting of two stereo cameras, a LiDAR and an IMU equipped on a mobile robot platform which utilized the mobile robot Pioneer P3DX,[3] as shown in Fig. 8a. In order to expand the field of view of visual information, the first Zed 2 camera[4] was placed in the robot's front, and the second Zed 2 camera was fixed in the robot's rear, as shown in Fig. 8a. Both cameras utilized the 720p resolution image 2560×720 pixels at 30 Hz for real-time performance. LiDAR sensor used Hokuyo UTM-30LX[5] operating at 40 Hz, and an MTi-100 IMU[6] sampled at 400 Hz ($dt = 2.5$ ms). We employed an embedded computer-NVIDIA Jetson AGX Xavier,[7] to operate the proposed system. Zed 2 cameras were connected to the embedded computer using a USB 3.0 hub. Hokuyo LiDAR and IMU were linked to the Xavier board with wire USB 2.0, as shown in Fig. 8a. A C++ software framework on the Linux-based operating system and ROS Melodic[8] is used to implement Algorithm 2. The FIS algorithm was trained on MATLAB with a desktop PC with an Intel 4-core i7-7700 processor CPU. We then implemented Algorithm 2 with four parallel threads on the CPU resources of the Jetson Xavier.

In this work, we calibrated the extrinsic parameters of the sensor setup step by step. First, the front camera and IMU were calibrated by using the kalibr toolbox.[9] Then, we determined the front and rear cameras using the ROS-based-package TagSLAM.[10] Next, 2D LiDAR and the front camera were computed by using ROS package.[11] Finally, we combine all the extrinsic parameters concerning the front camera frame. Table II shows the setting parameters of the proposed method as described in Subsection III-E.

B. Learning and Analysis Type-2 FIS

The mobile robot was manually controlled to operate in a human-made environment of around 2.5 m×60 m, as shown in Fig. 8b. Then, we collected a realistic dataset to train the Type 2-FIS structure and parameters as explained in

[2]http://mathworks.com/products/global-optimization.html
[3]http://generationrobots.com/402395-robot-mobile-pioneer-3-dx.html
[4]http://stereolabs.com/zed-2/
[5]http://hokuyo-usa.com/products/lidar-obstacle-detection/utm-30lx
[6]http://xsens.com/products/mti-100-series
[7]http://developer.nvidia.com/embedded/jetson-agx-xavier-developer-kit
[8]http://wiki.ros.org/melodic
[9]https://github.com/ethz-asl/kalibr/wiki/camera-imu-calibration
[10]https://berndpfrommer.github.io/tagslam_web/
[11]https://github.com/TurtleZhong/camera_lidar_calibration

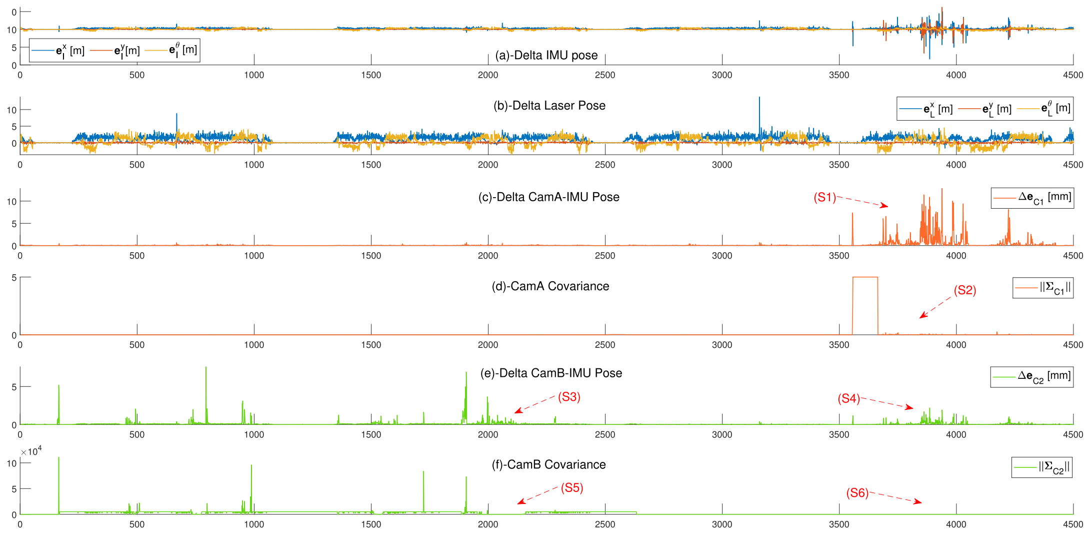

Fig. 9. The evaluation of the training dataset collected using inertial sensor, IMU preintegration, LiDAR, and stereo cameras estimation.

TABLE II
AN OVERVIEW OF THE SETTING OF THE EXPERIMENT

Sensor	**Sensor Freq (Hz)**	**Odometry Freq (Hz)**	**Odometry Method**
Front Camera	*30*	*30*	*VINS-Zed*
Rear Camera	*30*	*30*	*VINS-Zed*
2D LiDAR	*40*	*15*	*S2M*
IMU MTi-100	*400*	*400*	*Propagation*
Ours	**Odometry Freq (Hz)**	**Update Freq (Hz)**	**Method**
LiDAR-Visual-Inertial	*400*	*10-15*	*Factor Graph*

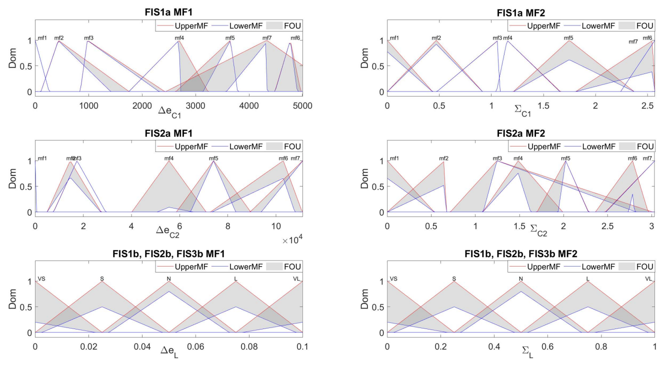

Fig. 10. The membership functions of the Type-2 FIS scheme (as shown in Fig. 6) after learning. Here, Dom represents the degree of membership which describe in MATLAB- Fuzzy Logic Toolbox.

Subsection III-F. We note that the S2M method with LiDAR sensor supports superior accuracy for the odometry in the environments mentioned earlier. Therefore, the proposed method used laser-based S2M to create the ground truth in the structure environments.

After collecting the dataset, the relative transition errors between the two keyframes were computed. The max-norm of the covariance matrices of VINS-Zed estimators were multiplied by 10^3 for handling efficiency. Fig 9 shows the training data after preprocessing. We note that the smaller the value of the max-norm of the covariance matrix, the more likely the information is valuable. However, in several regions, the VINS-Zed provided the wrong covariance matrices. Specifically, although arrow S1 shows a failure estimation of the front camera, arrow S2 indicates the trustable area simultaneously, as shown in Fig. 9. Furthermore, arrows S3 and S4 present the rear camera estimation failure regions, but arrows S5 and S6 indicate the small covariance at the same time, as shown in Fig. 9. Therefore, we design the blocks FIS1a and FIS2a to detect these uncertainty input covariances. Block FIS1a and FIS2a are trained with collected datasets mimicking laser-based S2M odometry. After training, the results of rules and MFs of block FIS1a and FIS2a are shown in Fig. 10. It is noted that the laser covariance matrix accurately reflects estimation information. So FIS3a is not used in this test. We do not train blocks FIS1b, FIS2b, and FIS3b which are only manually designed with five MFs of the inputs and outputs. The rules for block FIS3b differ from block FIS1b and block FIS2b. In general, the output after block FIS1b and block FIS2b gets larger when laser covariance increases. In contrast, the output of block FIS3b is grown if the laser covariance turns in no small value. Instead of directly updating the factor covariances, we leverage naive logic to only keep the smallest coefficient after block FIS1b, FIS2b, and FIS3b. Then we increase other factor scales by 10^3. Researchers can design another configuration depending on the experimental results or other sensor setups.

The parameters of the FIS design are presented in Table III. Table IV shows that we also compare the learned Type-2 FIS with other state-of-the-art supervised learning methods [45] by using MATLAB Statistics and Machine Learning Toolbox.[12] Although all other regression methods using local optimization can achieve faster training, Type-2 FIS achieves better accuracy and robustness. The neural networks (NNs) with a fully connected layer with 25 neural indicate the best result in all the local optimization approaches, as shown in Fig. 12. Nevertheless, the results of NNs indicate a small error,

[12]https://www.mathworks.com/products/statistics.html

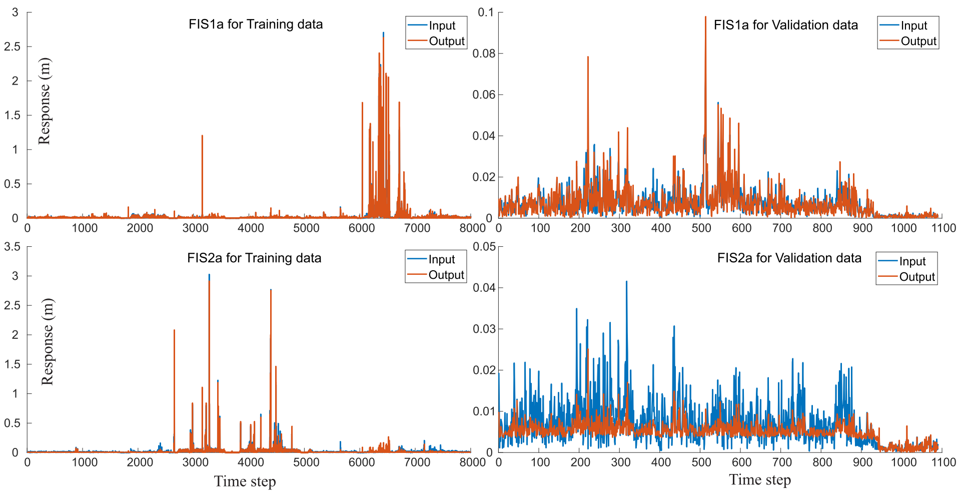

Fig. 11. The result of trained Type-2 FIS structure. On the left is the training process, and on the right is the validation plot.

TABLE III
THE DESIGN AND CHARACTERISTICS OF THE FIS STRUCTURE

FIS Name	Number of Input	Number of Input MFs	Number of Output	Number of Rules
FIS1a	*2*	*7*	*81*	*45*
FIS2a	*2*	*7*	*81*	*50*
FIS1b	*2*	*5*	*5*	*25*
FIS2b	*2*	*5*	*5*	*25*
FIS3b	*2*	*5*	*5*	*25*
And Method	**TypeReduction Method**	**Implication Method**	**Aggregation Method**	**De-fuzzy Method**
prod	*karnikmendel [55]*	*prod*	*sum*	*wtaver*

TABLE IV
THE COMPARATIVE RESULTS OF THE SUPERVISED LEARNING METHOD

Method	RMSE (m)	Training Time (s)	Optimization Method
Robust Linear Regression	*0.032*	*5*	*local*
Fine Tree	*0.035*	*5*	*local*
Support Vector Machine	*0.033*	*1000*	*local*
Ensemble	*0.030*	*35*	*local*
Gaussian process regression	*0.032*	*670*	*local*
Neural Network (1 layer)	*0.025*	*950*	*local*
Neural Network (2 layers)	*0.027*	*1010*	*local*
Neural Network (3 layers)	*0.026*	*1032*	*local*
Type 2- FIS	***0.018***	*15000*	*global*

which is not consistent with uncertainty compared with Type-2 FIS, as shown in Fig. 11.

C. Performance Evaluation and Comparison

1) Real-Time Performance: We analyze the processing time to run Algorithm 2 on an embedded computer under 30W - Jetson Xavier in this part. We benchmarked the proposed technique on an onboard system using an 8-core ARM v8.2 64-bit CPU running Ubuntu 18.04 LTS. The S2M method using 2D LiDAR has achieved the estimation frequencies of around 15 Hz. VINS-front and VINS-rear implementing Zed-VINS odometry obtained their state estimation at 30 Hz with a bit of time drift between the two estimators. The overall

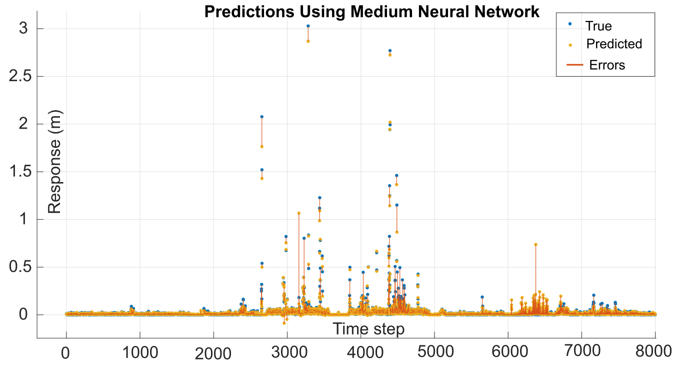

Fig. 12. The prediction results of the medium neural networks (NNs) after training indicate much error in the higher response.

processing time is shown in Table II. The preprocessing task consumed only 0.2 ms, and the FIS structure only demanded 0.8 ms, as explained in Algorithm 2. The optimization period of the proposed method is similar to the S2M solution of about 15 Hz. In particular, the optimization time depends on the sliding window size of around 5-10 ms when the number of keyframes is limited to 1000. Furthermore, the estimation output could be upsampled up to 400 Hz as the IMU clock.

2) Performance Evaluation: The proposed algorithm is evaluated in two sessions using the testing datasets,

1) In the first round, the noise model and the accuracy of VINS-Zed estimations were incorrect in several regions where cameras defected.
2) The second round imitated an actual situation by creating an error in LiDAR scan data. Although the S2M estimation method failed, the proposed system was robustly operating with a small error.

In the first scenario, the stereo cameras were masked with sheets of paper to examine the robustness of the proposed method. After moving for approximately one minute, the front camera was first covered, then the rear camera followed. Here, we masked the front and rear cameras for

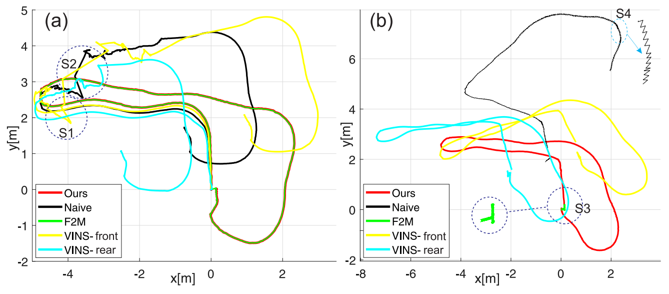

Fig. 13. (a)-The experimental results in round 1 when the camera fails, (b)-when the laser fails in round 2.

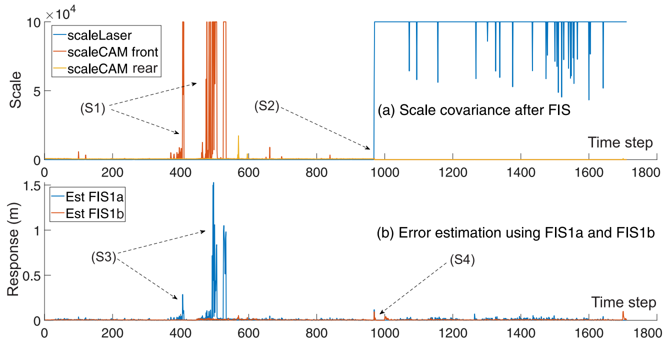

Fig. 14. The output of the FIS scaling factor is indicated in the experiments in Fig. 13.

about 60 and 20 seconds, respectively. The VINS-front and VINS-rear estimations failed, but they could recover when the cameras' masks were removed. We used the same scale factors without updating covariance information, called the naive fusion approach. The first case result is presented in Fig. 13a. We used the absolute trajectory error (ATE) [56] to calculate the accuracy. Here, the ATE is represented by the average root-mean-square error (RMSE) of the estimated trajectory and ground truth given as,

$$e_{ATE} = \sqrt{\frac{1}{K}\sum_{i=1}^{K}\left\|\mathbf{p}_i - \hat{\mathbf{p}}_i\right\|^2 (m)} \tag{17}$$

where $\mathbf{p}_i$ and $\hat{\mathbf{p}}_i$ are the current position and ground-truth at time step i; K is the number of samplings.

In particular, arrow S1 indicated the lost tracking area of the front camera, and the front VINS-Zed estimator began failure, as shown in Fig. 13-a. Region S2 indicated the lost tracking area of the rear camera. However, the influence of region S2 was much smaller than region S1, as shown in Fig. 13-a. In this case, the accuracy of LiDAR was the most precise, followed by the rear camera, and the worst result was the front camera. Hence, the prediction outcomes of the scale factor of the LiDAR sensor were about 5, the rear VINS round 1000, and the front VINS round 10^5, as shown in area S1 of Fig. 14-a. The error estimation of the front-VINS and rear-VINS is the output of FIS1a and FIS2a, as shown in area S3 of Fig. 14-b.

The second scenario began when the robot passed the initial pose after finishing the first round. The laser-based

TABLE V

The Evaluation Results of Ours and Other Methods. The Best Score Is in Bold, 'x' Denotes a Failed Experiment, and 'o' Indicates the Wrong Estimation but Not Failure. D2E Is the Distance-to-End Point Following the KITTI Evaluation Critical [57]

Technique	ATE (m)	D2E (m)	D2E (%)	Note
VINS-ZED front camera	0.022	1.363	3.78	recovery after failure
VINS-ZED rear camera	0.019	1.136	3.16	recovery after failure
VINS-fusion OpenVINS ORB-SLAM	x	x	x	not recovery after failure
S2M	o	o	o	recovery after failure
Fusion without FIS (Naive)	0.016	2.90	8.06	no failure
Ours	**0.005**	**0.074**	**0.2**	no failure

S2M method provided poor results as represented by a green trajectory denoted by region S3, as shown in Fig. 13-b. Note that we removed the previous masks to recover the field of view of both stereo cameras. In contrast, we narrowed the laser scan data with a mask to mimic the geometrically degenerate cases.

Table V presents our evaluation results and other methods. Although the VINS-front and VINS-rear failed in section 1, the system used their estimation information for the data fusion process. We used a remote control device to drive the robot to return to the initial pose. The first visualization realized that our system could follow the ground truth with the same draw. Here, the ATE was less than 0.005m, calculated similarly to our previous work [27]. The VINS-ZED of the front and rear cameras can recover after failure. The total error of both VINS depends on the time of failure tracking. Table V showed that the error of ZED-VINS in ATE is more than four times higher than ours, much far away from the initial point indicated by the D2E score. Then we examined the error using the KITTI criterion [27], [57] of about 0.2%. Although the fusion without FIS got poor results with a fluctuated estimation indicated in area S4, as shown in Fig. 13-b, the ATE was smaller than VINS-ZED because it did not fail. In this situation, the covariance scale of the laser factor was enormous, and the front and rear camera scales were small, as shown in regions S3 and S4 of Fig. 14.

The result of other VINS methods such as ORB-SLAM [52], [58], VINS-fusion [7], and OpenVINS [9] failed in this scenario because of covered the field of view of the cameras. The main reason is that the methods can not recover from failure. Also, the naive fusion approach did not guarantee accuracy with much drift compared to the ground truth, as shown in Fig. 13-a.

V. Conclusion

This paper proposed a fundamental method for a robust multi-sensor fusion framework based on AI-assisted factor graph optimization. Our system could handle challenging scenarios such as uncertainty sensor observation models or sensor failures. We deployed Type-2 FIS to learn the uncertainty with

PSO global optimization to update the information matrices in the factor graph. The integration of the learned Type-2 FIS enabled adapting the covariance for each visual sensor factor. The outcomes indicated that the proposed method presented great accuracy and robustness for uncertainty input covariances and sensor failures. Our system could be quickly applied to various sensors for diverse applications.

REFERENCES

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

[2] 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.

[3] 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*.

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

[5] D. Scaramuzza and Z. Zhang, "Visual-inertial odometry of aerial robots," 2019, *arXiv:1906.03289*.

[6] M. Palieri et al., "LOCUS: A multi-sensor LiDAR-centric solution for high-precision odometry and 3D mapping in real-time," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 421–428, Apr. 2021.

[7] T. Qin, S. Cao, J. Pan, and S. Shen, "A general optimization-based framework for global pose estimation with multiple sensors," 2019, *arXiv:1901.03642*.

[8] T. Sandy, L. Stadelmann, S. Kerscher, and J. Buchli, "ConFusion: Sensor fusion for complex robotic systems using nonlinear optimization," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 1093–1100, Apr. 2019.

[9] 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.

[10] 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.

[11] 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*.

[12] 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.

[13] P. Corke, J. Lobo, and J. Dias, "An introduction to inertial and visual sensing," *Int. J. Robot. Res.*, vol. 26, no. 6, pp. 519–535, 2007, doi: 10.1177/0278364907079279.

[14] N. Sünderhauf et al., "The limits and potentials of deep learning for robotics," *Int. J. Robot. Res.*, vol. 37, nos. 4–5, pp. 405–420, 2018.

[15] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep Learning*, vol. 1, no. 2. Cambridge, U.K.: MIT Press, 2016.

[16] A.-T. Nguyen, T. Taniguchi, L. Eciolaza, V. Campos, R. Palhares, and M. Sugeno, "Fuzzy control systems: Past, present and future," *IEEE Comput. Intell. Mag.*, vol. 14, no. 1, pp. 56–68, Feb. 2019.

[17] J. Mendel et al., *Introduction to Type-2 Fuzzy Logic Control: Theory and Applications*. Hoboken, NJ, USA: Wiley, 2014.

[18] 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.

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

[20] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. Cambridge, MA, USA: MIT Press, 2005.

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

[22] M. Wang and A. Tayebi, "Observers design for inertial navigation systems: A brief tutorial," in *Proc. 59th IEEE Conf. Decis. Control (CDC)*, Dec. 2020, pp. 1320–1327.

[23] S. Lynen et al., "A robust and modular multi-sensor fusion approach applied to MAV navigation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Nov. 2013, pp. 3923–3929.

[24] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar, "Multi-sensor fusion for robust autonomous flight in indoor and outdoor environments with a rotorcraft MAV," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2014, pp. 4974–4981.

[25] T. Moore and D. Stouch, "A generalized extended Kalman filter implementation for the robot operating system," in *Intelligent Autonomous Systems 13* (Advances in Intelligent Systems and Computing), vol. 302, E. Menegatti, N. Michael, K. Berns, and H. Yamaguchi, Eds. Cham, Switzerland: Springer, 2016, doi: 10.1007/978-3-319-08338-4_25.

[26] C. Brommer, R. Jung, J. Steinbrener, and S. Weiss, "MaRS: A modular and robust sensor-fusion framework," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 359–366, Apr. 2021.

[27] 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.

[28] H. Strasdat, J. M. M. Montiel, and A. J. Davison, "Visual SLAM: Why filter?" *Image Vis. Comput.*, vol. 30, no. 2, pp. 65–77, 2012.

[29] R. Mascaro, L. Teixeira, T. Hinzmann, R. Siegwart, and M. Chli, "GOMSF: Graph-optimization based multi-sensor fusion for robust UAV pose estimation," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 1421–1428.

[30] D. Wisth, M. Camurri, S. Das, and M. Fallon, "Unified multi-modal landmark tracking for tightly coupled LiDAR-visual-inertial odometry," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 1004–1011, Apr. 2021.

[31] X. Zuo et al., "LIC-fusion 2.0: LiDAR-inertial-camera odometry with sliding-window plane-feature tracking," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 5112–5119.

[32] X. Zuo, P. Geneva, W. Lee, Y. Liu, and G. Huang, "LIC-fusion: LiDAR-inertial-camera odometry," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Nov. 2019, pp. 5848–5854.

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

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

[35] P. Sodhi, M. Kaess, M. Mukadam, and S. Anderson, "Learning tactile models for factor graph-based estimation," 2020, *arXiv:2012.03768*.

[36] A. Baikovitz, P. Sodhi, M. Dille, and M. Kaess, "Ground encoding: Learned factor graph-based models for localizing ground penetrating radar," 2021, *arXiv:2103.15317*.

[37] A. Kloss, G. Martius, and J. Bohg, "How to train your differentiable filter," *Auto. Robots*, vol. 45, pp. 1–18, Jun. 2021.

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

[39] D. V. Nam and K. Gon-Woo, "Learning observation model for factor graph based-state estimation using intrinsic sensors," *IEEE Trans. Autom. Sci. Eng.*, early access, Jun. 28, 2022, doi: 10.1109/TASE.2022.3193411.

[40] L. Pineda et al., "Theseus: A library for differentiable nonlinear optimization," 2022, *arXiv:2207.09442*.

[41] H. Martiros et al., "SymForce: Symbolic computation and code generation for robotics," 2022, *arXiv:2204.07889*.

[42] B. Dai, Y. He, L. Yang, Y. Su, Y. Yue, and W. Xu, "SIMSF: A scale insensitive multi-sensor fusion framework for unmanned aerial vehicles based on graph optimization," *IEEE Access*, vol. 8, pp. 118273–118284, 2020.

[43] A. Reinke et al., "LOCUS 2.0: Robust and computationally efficient lidar odometry for real-time 3D mapping," *IEEE Robot. Autom. Lett.*, vol. 7, no. 4, pp. 9043–9050, Oct. 2022.

[44] C. Zheng, Q. Zhu, W. Xu, X. Liu, Q. Guo, and F. Zhang, "FAST-LIVO: Fast and tightly-coupled sparse-direct LiDAR-inertial-visual odometry," 2022, *arXiv:2203.00893*.

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

[46] F. Pomerleau, F. Colas, R. Siegwart, and S. Magnenat, "Comparing ICP variants on real-world data sets," *Auto. Robots*, vol. 34, no. 3, pp. 133–148, 2013.

[47] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual–inertial odometry using nonlinear optimization," *Int. J. Robot. Res.*, vol. 34, no. 3, pp. 314–334, 2015.

[48] E. Olson, "AprilTag: A robust and flexible visual fiducial system," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 3400–3407.

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

[50] 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.
[51] A. Censi, "An ICP variant using a point-to-line metric," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2008, pp. 19–25.
[52] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM3: An accurate open-source library for visual, visual-inertial and multi-map SLAM," 2020, *arXiv:2007.11898*.
[53] J. Li et al., "Accurate 3D localization for MAV swarms by UWB and IMU fusion," in *Proc. IEEE 14th Int. Conf. Control Autom. (ICCA)*, Jun. 2018, pp. 100–105.
[54] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization," *Swarm Intell.*, vol. 1, no. 1, pp. 33–57, Jun. 2007.
[55] D. Wu and M. Nie, "Comparison and practical implementation of type-reduction algorithms for type-2 fuzzy sets and systems," in *Proc. IEEE Int. Conf. Fuzzy Syst. (FUZZ-IEEE)*, Jun. 2011, pp. 2131–2138.
[56] Z. Zhang and D. Scaramuzza, "A tutorial on quantitative trajectory evaluation for visual(-inertial) odometry," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 7244–7251.
[57] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1231–1237, 2013.
[58] 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.

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. He has been a Lecturer at the School of Engineering and Technology, Vinh University, Vietnam, since 2013. His research interests include AI-robotics, SLAM, motion planning, 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.