Received July 8, 2021, accepted August 2, 2021, date of publication August 24, 2021, date of current version September 7, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3106864

Dynamics-Based Modified Fast Simultaneous Localization and Mapping for Unmanned Aerial Vehicles With Joint Inertial Sensor Bias and Drift Estimation

NARGESS SADEGHZADEH-NOKHODBERIZ[1], AYDIN CAN[2], RUSTAM STOLKIN[3], AND ALLAHYAR MONTAZERI[2]

[1]Department of Electrical and Computer Engineering, Qom University of Technology, Qom 3718146645, Iran
[2]Department of Engineering, Lancaster University, Lancaster, Lancashire LA1 4YW, U.K.
[3]Extreme Robotics Laboratory (ERL), School of Metallurgy and Materials, University of Birmingham, Birmingham B15 2TT, U.K.

Corresponding author: Allahyar Montazeri (a.montazeri@lancaster.ac.uk)

This work was supported in part by the Engineering and Physical Sciences Research Council (EPSRC) under Grant EP/R02572X/1, in part by the National Centre for Nuclear Robotics (NCNR), and in part by the Centre for Innovative Nuclear Decommissioning (CINDe), which is led by the National Nuclear Laboratory, in partnership with Sellafield Ltd. and a network of universities that includes The University of Manchester, Lancaster University, the University of Liverpool, and the University of Cumbria.

ABSTRACT In this paper, the problem of simultaneous localization and mapping (SLAM) using a modified Rao Blackwellized Particle Filter (RBPF) (a modified FastSLAM) is developed for a quadcopter system. It is intended to overcome the problem of inaccurate localization and mapping caused by inertial sensory faulty measurements (due to biases, drifts and noises) injected in the kinematics (odometery based) which is commonly used as a motion model in FastSLAM approaches. In this paper, the quadcopter's dynamics with augmented bias and drift models is employed to eliminate these faults from the localization and mapping process. A modified FastSLAM is then developed in which both Kalman Filter (KF) and Extended Kalman Filter (EKF) algorithms are embedded in a PF with modified particles weights to estimate biases, drifts and landmark locations, respectively. In order to make the SLAM process robust to model mismatches due to parameter uncertainties in the dynamics, measurements are incorporated in the PF and in the particle generation process. This leads to a cascaded two-stage modified FastSLAM in which the extended FastSLAM 1.0 (to include dynamics and sensory faults) is employed in first stage and the results are used in second stage in which probabilistic inverse sensor models are incorporated in the particle generation process of the PF. The efficiency of the proposed approach is demonstrated through a co-simulation between MATLAB-2019b and Gazebo in the robotic operating system (ROS) in which the quadcopter model is simulated in Gazebo in ROS using a modified version of the Hector quadcopter ROS package. The collected pointcloud data using LiDAR is then utilised for feature extraction in the Gazebo. The simulation environment used to this aim is validated based on experimental data.

INDEX TERMS Unmanned aerial vehicle (UAV), Rao Blackwellized particle filtering (RBPF), simultaneous localization and mapping (SLAM), FastSLAM, inertial sensors, sensor calibration, robot operating system (ROS).

I. INTRODUCTION

Simultaneous localization and mapping (SLAM) is the problem of localization of an object when the environment is not modeled and needs to be mapped at the same time. SLAM is required for autonomous navigation of mobile robots in different scenarios, including indoor, outdoor, underwater, and space applications. Although Global Navigation Satellite System (GNSS) can be employed for localization on the planet earth, this system is not always accurately available [1]. SLAM is a very important part of autonomous navigation using unmanned aerial vehicles (UAVs), especially in

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

highly dangerous environments. In some applications such as nuclear decommissioning [2], a UAV quadcopter, which is subject to radiation, should be able to map a radioactive environment and localize itself simultaneously.

There exist a variety of approaches available for SLAM depending on some factors such as spatial representation, the structure and dynamics of environment and employed sensors [3]. Although the diversity of approaches [4], [5] are important, in this paper we focus on Bayesian methods and landmark based mapping to localize a quadcopter system maneuvering in an unknown 3D environment. Most sensors cannot provide full 3D information. However, stereo or RGBD cameras, such as Kinect or Asus Xtion, and 3D laser rangers, such as Velodyne laser scanners, are the most well-known sensors which can provide 3D measurements [4].

In the field of Bayesian methods, Extended Kalman Filter (EKF) based SLAM named as EKF-SLAM [6] was firstly proposed to localize and map by estimating the posterior probability distribution over robot pose and landmark positions. However, this algorithm suffers from computational complexity in larger environments with a lot of landmarks [7]. In this algorithm, the dimension of the state vector, including robot pose and landmark positions, grows with time as new landmarks appear in the environment which leads to an overall $O(L^3)$ complexity for each time step where L is the number of landmarks. As the size of the map increases, the performance of EKF-SLAM quickly degrades until it is not feasible anymore for a real-time operation [3]. To overcome this problem, FastSLAM [7] was proposed, in which the filtering based SLAM is decomposed into a robot localization problem, and a collection of landmark estimation problems conditioned on the robot pose estimate. This can be obtained using Bayes' rule combined with the statistical independency of landmark positions which leads to Rao Blackwellized particle filtering (RBPF) SLAM [8].

System model, robot's motion model in FastSLAM algorithm, is employed for particle generation as proposal distribution in the PF. This degrades the performance of the PF when the proposal distribution is poorly matched with the posterior due to model mismatches and noises. To overcome this drawback, FastSLAM 2.0 incorporates the measurement model into the proposal distribution. The distribution is derived with Gaussian assumption and linearization of a generally nonlinear measurement model around the robot pose and landmarks position vectors [9], [10].

The SLAM algorithm has also been employed for unmanned aerial vehicles (UAV) in many research works [11]–[13]. Inertial sensors (accelerometers and gyroscopes) are normally used in such applications as proprioceptive sensors [3], especially in applications that global positioning system (GPS) is unavailable [12]. In [14] an airborne EKF-SLAM is implemented to fuse data from an Inertial Measurement Unit (IMU) and a passive vision system in a real time fashion. In [12] design and implementation of a robust inertial sensor based EKF-SLAM algorithm for a UAV using bearing-only observations provided by a single color vision camera, is studied. Reference [13] provides a solution to SLAM problem for a UAV system using a camera and inertial sensors. The proposed method is an optimization based method, which results in a smoother, rather than a filter. In spite of using inertial sensors, none of the mentioned works consider the problem of sensory biases and drifts. The importance of considering inertial sensory errors is due to the fact that although MEMS inertial sensors are small, light, inexpensive and they consume less power and have short start-up time, they suffer from sensory soft faults such as biases, drifts and noises [15]. Such faulty measurements are normally used as the inputs to the kinematic model (odometery based) of the robot to solve the SLAM problem, which leads to an inaccurate localization and mapping. The problem of bias and drift compensation in SLAM scenarios has been proposed in some research works and mostly for ground mobile robots [11], [16]–[26]. Table 1 provides a general comparison between the existing methods and then to see how we are improving on the existing ones in this paper.

Referring to Table 1, it can be easily understood that the problem of FastSLAM is still not developed for our LiDAR-IMU dynamic based SLAM problem. Although the RBPF with the corresponding PF and EKFs for different subsystems is developed in [11] for the augmented kinematics, it is not possible to employ the proposed method in this paper. First off, since the sensory biases have been appearing in the kinematic model through the sensor measurements as mentioned in [11], a tightly coupled bias and pose subsystem will be achieved and this requires a special derivation of EKF. Besides, although the particle weights are computed in [11] for the developed RBPF, it is with the assumption of Gaussianity and the linearity of the camera measurement model for the observed landmark states. However, even in the presence of the Gaussian assumption for the measurement noise, the LiDAR measurement model, as explained later in the paper, is not a linear function of the observed state of landmarks.

Due to the difficulties in using the kinematic models when sensory measurements are faulty, compared to the commonly used kinematics motion model in FastSLAM, in this paper an exact dynamic model of 6DOF quadcopter system is employed. The idea of using dynamics instead of kinematics has been previously studied in [27] for autonomous racing and maneuvering using EKF-SLAM method. In the paper, it is shown that taking the tire forces into account and using the dynamic model of the system yields a significantly better accuracy for the pose estimation.

In the current investigation, however, an RBPF algorithm is proposed in which the algorithm is augmented with the dynamic model of the system as well as the LiDAR-IMU measurement model. The motivation behind this comes from the fact that the nonlinearity of the LiDAR measurement model with respect to both the robot's and landmarks' poses requires a more general weight computation method.

In this paper, both biases and drifts are estimated exactly and eliminated from measurements. To this end, an accurate

TABLE 1. Comparison of existing methods for joint inertial bias estimation and SLAM.

Reference No.	Sensors	Motion Model	General Method	Impelementation Method
[24]	IMU, monocular camera	kinematics with augmented bias model	optimization based tracking and ORB-SLAM with BA[1] mapping	simulation using the EuRoC dataset
[18]	IMU, LiDAR	kinematics with augmented bias model	graph based optimization approach is employed to estimate bias and motion parameters through linearization. 3-D planar landmarks with CP[2] representation is employed for mapping	simulation
[19]	IMU, camera	kinematics with augmented bias model	pose calibration of camera to inertial sensor is performed along with inertial bias estimation using UKF	experimental results only for sensor to sensor (MEMS-IMU and camera) calibration
[16]	gyroscope, encoder, camera	kinematics with augmented gyro z-axis deviation	EKF-SLAM	experimental results using the Samsung cleaning robot VC-RE70V
[20]	IMU, LiDAR	kinematics with augmented bias model	EKF-SLAM along with observability study for 1D and 2D environments and investigation of uncertainty bounds in estimation of biases for 2D environments	experimental results on a in-house built mobile robot
[26]	IMU, stereo camera	kinematics with augmented bias model	EKF-SALM	experimental results in the sea
[22], [21]	IMU, LiDAR	kinematics with augmented bias model	to apply Fast-SLAM the general partitioning in posterior distribution to localization and mapping with embedded bias estimation is presented, however, no derivation of RBPF for the augmented model is presented.	simulation
[11]	IMU, camera	kinematics with augmented bias model	the extended Fast-SLAM method for the augmented motion model considering the kinematics and camera measurement model with the assumption of Gaussianity and the linearity of camera measurement model of the observed landmark's state	experimental results on a UAV platform

6DOF dynamic model of quadcopter is employed through which the true motion of the system, i.e. the linear acceleration and angular velocity of the quadcopter can be modeled. Biases and drifts are also modeled and augmented to the system dynamics. Since this augmentation leads to an increased dimension compared to the normal FastSLAM, in this paper more decomposition according to the posterior distribution function and Bayes' rule is performed. This leads to an extended RBPF consisting of a PF with embedded KF and EKF due to posterior factorization into conditional landmark, conditional bias and drift distributions and posterior distribution over robot path. The developed RBPF recursively estimates the full posterior distribution over the quadcopter's states including its 6DOF pose, landmark positions as well as the sensory biases and drifts. It can be easily proved that the extra KF set, in addition to the existing EKF set in the normal RBPF SLAM, is required to estimate the inertial biases and drifts conditioned on the robot's pose particles. In the PF, the particles corresponding to the robot's path are generated using the robot's dynamics. Although the robot's dynamics provides a more precise model of robot's path than its kinematics, it still suffers from the model mismatch due to reasons such as noises and parametric uncertainties. Thus, it is proposed to incorporate the measurements for particle generation similar to FasSLAM 2.0. However, there are some drawbacks in FastSLAM 2.0 such as Gaussian assumption of noise probability models and linearizations in measurement models. These drawbacks degrade the performance of FastSLAM 2.0. Therefore, in this paper, the inverse sensor models are employed for the particle generation in addition to the robot's dynamics where possible and depending on the sensor models. Accordingly, two different proposal distributions are employed for particle generation in the proposed FastSLAM algorithm of this paper and the corresponding weights of the particles are computed. It is shown that for the particles generated using the robot's dynamics, both inertial and LiDAR measurement likelihoods get involved in the weight computation process. Also, through the posterior distribution manipulations it is proved that the particles generated using the inverse sensor model are weighted equivalently if the particles generated in the previous stage get involved in this stage. Thus, a cascaded 2-stage modified FastSLAM algorithm is proposed where, in the first stage the typical FastSLAM 1.0 is extended to include the bias and drift estimation and in the second stage, using the results of the first stage, some generated particles are regenerated using the inverse sensor models to overcome model uncertainties. It is worth noting that, both range and bearing are measured using an exteroceptive sensor such as a LiDAR while the IMU sensor is employed as a proprioceptive one and their inverse sensor model is applied for the particle generation process.

In order to implement the proposed method, Gazebo in the Robot Operating System (ROS) is employed in this paper in which the quadcopter model is simulated in Gazebo in ROS using a modified version of the Hector quadcopter ROS package and a Velodyne VLP-16 3D LiDAR is also simulated using Gazebo. The collected pointcloud data is then visualized in RViz and a ROS package is then implemented to allow features placed in the Gazebo simulation environment to be extracted. The simulation environment we are using is validated based on experimental data. The model parameters are amenable to change due LiDAR installation and

the algorithm is robust to the parameter changes in spite of designing and embedding the continuous time recursive least square (RLS) [28] algorithm.

To summarize, the main contributions of this paper are summarized as follows

1) An extended RBPF to solve the SLAM problem in a quadcopter system is developed by augmenting the dynamics of the quadcopter with the bias and drift models.

2) To make the approach robust to model uncertainties, it is proposed to generate particles using both system dynamics and inverse sensor models to involve measurements in the particle generation and to overcome the drawbacks of the FastSLAM 2.0 approach. Moreover. the exact formulation is provided for the posterior distribution approximation using the inverse sensor model.

3) A modified FastSLAM algorithm which is a cascaded 2-stage approach is proposed. In the first stage, the typical FastSLAM 1.0 is extended to include the bias estimation and in the second stage, using the results of first stage, some generated particles are regenerated using the inverse sensor models to overcome the model uncertainties.

4) The algorithm is evaluated through a co-simulation between MATLAB-2019b and Gazebo in Robot Operating System (ROS).

This paper is organized as follows. In Section II the problem is formulated by introducing the quadcopter dynamic model, the inertial sensor measurement and bias-drift model as well as range and bearing measurement models. The modified FastSLAM algorithm is extended and derived in Section III. Section IV presents simulation results and conclusions are provided in Section V.

II. SYSTEM DESCRIPTION AND PROBLEM FORMULATION

A. SYSTEM MODEL

A quadcopter UAV includes four rotors that generate propeller forces. A schematic of the quadcopter with the body coordinate system is depicted in Figure 1. Variations on forces (f_1 to f_4) and moments (τ_{M_1} to τ_{M_4}) by adjusting rotors' speeds (ω_1 to ω_4) produce attitude and translation change in the quadcopter.

The quadcopter attitude dynamics can be written in the following form [29]:

$$\begin{aligned}\dot{p} &= \frac{I_{yy}-I_{zz}}{I_{xx}}qr - I_r\Omega_r\frac{q}{I_{xx}} + \frac{u_2}{I_{xx}},\\ \dot{q} &= \frac{I_{zz}-I_{xx}}{I_{yy}}pr + I_r\Omega_r\frac{p}{I_{yy}} + \frac{u_3}{I_{yy}},\\ \dot{r} &= \frac{I_{xx}-I_{yy}}{I_{zz}}pq + \frac{u_4}{I_{zz}},\end{aligned} \tag{1}$$

where, p, q and r are angular velocities rotating around x-axis, y-axis and z-axis in the body frame, respectively. The inertia tensors are represented as I_{xx}, I_{yy} and I_{zz}. The input signals u_2, u_3 and u_4 are torques in the direction of the corresponding body frame angles. I_r is the inertia of the propellers and $\Omega_r = \omega_1 - \omega_2 + \omega_3 - \omega_4$ describes the relative speed of the propellers.

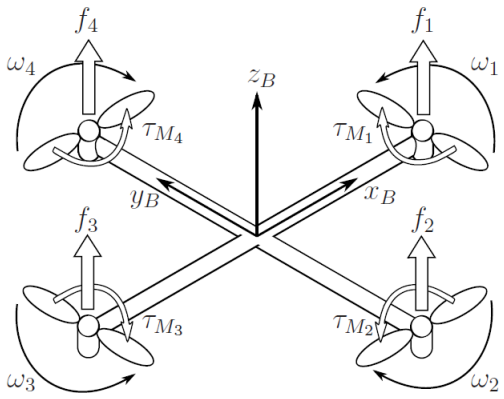

FIGURE 1. Schematic of quadcopter with coordinate axes [29].

The relationship between angular velocities in the body and inertial frames are described as follows [29]:

$$\begin{aligned}\dot{\phi} &= p + \sin(\phi)\tan(\theta)q + \cos(\phi)\tan(\theta)r,\\ \dot{\theta} &= \cos(\theta)q - \sin(\phi)r,\\ \dot{\psi} &= \frac{\sin(\phi)}{\cos(\theta)}q + \frac{cos(\phi)}{\cos(\theta)}r,\end{aligned} \tag{2}$$

where $[\phi\ \theta\ \psi]^T$ is the attitude vector of quadcopter which is defined in the inertial frame where roll angle ϕ, pitch angle θ and yaw angle ψ determine rotations around x-axis, y-axis and z-axis, respectively.

The quadcopter's translational dynamics are presented in the following [29]:

$$\begin{aligned}\ddot{x} &= \frac{u_1}{m}(\cos(\psi)\sin(\theta)\cos(\phi) + \sin(\psi)\sin(\phi)),\\ \ddot{y} &= \frac{u_1}{m}(\sin(\psi)\sin(\theta)\cos(\phi) - \cos(\psi)\sin(\phi)),\\ \ddot{z} &= -g + \frac{u_1}{m}\cos(\phi)\cos(\theta),\end{aligned} \tag{3}$$

where, $[x\,y\,z]^T$ represents the position of quadcopter in the inertial frame, u_1 defines the main thrust created by combined forces of rotors, that is $u_1 = \sum_{i=1}^{4} f_i$. Moreover, in (3) g is the gravity constant and m is mass of quadcopter.

Let $\mathbf{x}^D = [x\ y\ z\ \dot{x}\ \dot{y}\ \dot{z}\ \phi\ \theta\ \psi\ p\ q\ r]^T$ and $\mathbf{u} = [u_1\ u_2\ u_3\ u_4]^T$. It is worth mentioning that in $\mathbf{x}^D$, D stands for "Dynamic". After time discretization and considering process noises, the following general nonlinear state space model is obtained:

$$\mathbf{x}^D_{k+1} = f(\mathbf{x}^D_k, \mathbf{u}_k) + \boldsymbol{\omega}_k, \tag{4}$$

where $\mathbf{x}^D_k \in \mathbb{R}^{12}$, $k \geq 0$, is the system state vector at time instant k and $\mathbf{u}_k \in \mathbb{R}^4$, is the input vector and $\boldsymbol{\omega}_k \in \mathbb{R}^{12}$, is the zero mean and generally non-Gaussian random process noise vector with a known probability density function.

B. MEASUREMENT MODEL

In this paper, both proprioceptive and exteroceptive sensors are employed to perform localization and mapping. Inertial sensors, including gyroscopes and accelerometers, are

the proprioceptive ones which measure respectively angular velocities and linear accelerations in the body frame. A LiDAR is used in this paper as the exteroceptive sensor which measures relative range and bearing to a landmark.

1) INERTIAL SENSORS FAULTY MEASUREMENT MODELS

The inertial sensors are mostly disturbed with sensory biases and drifts. Bias is a constant offset and drift is a time varying offset from the nominal statistics of the sensor signal [15]. Firstly, in the following, the acceleration measurement model is presented:

$$\mathbf{a}_m = \mathbf{a}_b + \mathbf{b}_{0a} + \mathbf{b}_{1a} + \boldsymbol{\nu}_a \tag{5}$$

where, $\mathbf{a}_m \in \mathbb{R}^3$ and $\mathbf{a}_b \in \mathbb{R}^3$ are, respectively, the vector of measured acceleration and the true value of acceleration in the body frame. The Term $\boldsymbol{\nu}_a$ is measurement noise with the covariance matrix of $\sigma^2_{\nu_a}\mathrm{I}$ and I is a 3×3 identity matrix. The term $\mathbf{b}_{0a}$ in (5) represents the constant offset and $\mathbf{b}_{1a}$ shows the sensor drift.

Since the accelerations in (5) are measured in the body frame, the rotation matrix as a function of ϕ, θ and ψ, is used to transform the measurements from the body frame to the inertial frame. The rotation matrix (XYZ rotation) from the body frame to the inertial frame is described as:

$$\mathbf{R}^w_b = \begin{bmatrix} C\psi C\theta & C\psi S\theta S\phi - S\psi C\phi & C\psi S\theta C\phi + S\psi C\phi \\ S\psi C\theta & S\psi S\theta S\phi + C\psi C\phi & S\psi S\theta C\phi - C\psi S\phi \\ -S\theta & C\theta S\phi & C\theta C\phi \end{bmatrix}, \tag{6}$$

where C and S stand for cosine and sinusoid functions, respectively.

Therefore, the measurement model can be rewritten as:

$$\mathbf{a}_m = (\mathbf{R}^w_b)^T \mathbf{a}_w + \mathbf{b}_{0a} + \mathbf{b}_{1a} + \boldsymbol{\nu}_a, \tag{7}$$

where $\mathbf{a}_w = [\ddot{x}\ \ddot{y}\ \ddot{z}]^T$ is acceleration in the inertial frame given in equation (3). Accelerometer bias and drift in (7) can be modeled as a random walk process [30]:

$$\begin{aligned} \dot{\mathbf{b}}_{0a} &= \mathbf{0}, \\ \dot{\mathbf{b}}_{1a} &= -\frac{1}{\tau}\mathbf{b}_{1a} + \boldsymbol{\omega}_{b_{1a}}, \end{aligned} \tag{8}$$

where, τ is the time constant and $\boldsymbol{\omega}_{b_{1a}}$ is the process noise with covariance matrix of $\sigma^2_{b_{1a}}\mathrm{I}$.

Similarly, the gyroscope measurement model is as follows:

$$\boldsymbol{\omega}_m = \boldsymbol{\omega}_b + \mathbf{b}_{0\omega} + \mathbf{b}_{1\omega} + \boldsymbol{\nu}_\omega, \tag{9}$$

where, $\boldsymbol{\omega}_m \in \mathbb{R}^3$ and $\boldsymbol{\omega}_b \in \mathbb{R}^3$ are, the vector of measured and true angular velocities in the body frame i.e. $\boldsymbol{\omega}_b = [p\ q\ r]^T$ given in (1). The term $\mathbf{b}_{0\omega}$ in (9) represents the constant offset and the term $\mathbf{b}_{1\omega}$ shows sensor drift in (9). $\boldsymbol{\nu}_\omega$ is zero mean Gaussian measurement noise with the covariance matrix of $\sigma^2_{\nu_\omega}\mathrm{I}$ and I is a 3×3 identity matrix.

Gyroscope's bias and drift dynamics can be modeled as random walk process [30]:

$$\dot{\mathbf{b}}_{0\omega} = \mathbf{0},$$

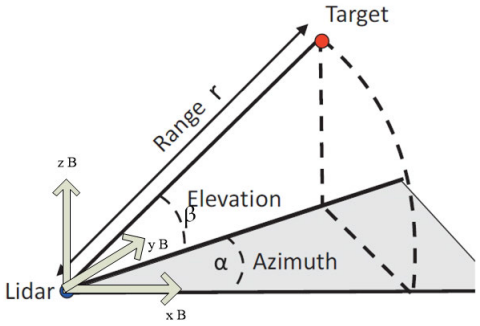

FIGURE 2. LiDAR range and bearing measurements with body frame.

$$\dot{\mathbf{b}}_{1\omega} = -\frac{1}{\tau}\mathbf{b}_{1\omega} + \boldsymbol{\omega}_{b_{1\omega}}, \tag{10}$$

where, τ is time constant and $\boldsymbol{\omega}_{b_{1\omega}}$ is the process noise with covariance matrix of $\sigma^2_{b_{1\omega}}\mathrm{I}$ and I is a 3×3 identity matrix.

Let $\mathbf{x}^B = [\mathbf{b}^T_{0a}\ \mathbf{b}^T_{0\omega}\ \mathbf{b}^T_{1a}\ \mathbf{b}^T_{1\omega}]^T \in \mathbb{R}^{12}$ be the augmented state vector, the discrete time state space model of the second subsystem (sensory bias and drift dynamics) can be written as:

$$\mathbf{x}^B_{k+1} = \mathbf{A}^b \mathbf{x}^B_k + \boldsymbol{\omega}^b_k, \tag{11}$$

where, $\mathbf{A}^b = \mathrm{blkdiag}(\mathrm{I}, \mathrm{I}, (1 - \frac{T}{\tau})\mathrm{I}, (1 - \frac{T}{\tau})\mathrm{I})$ and blkdiag refers to the block diagonal matrix, I is a 3×3 identity matrix and T is sampling time. Besides, $\boldsymbol{\omega}^b_k = [\mathbf{0}\ \ \mathbf{0}\ \ T\boldsymbol{\omega}^T_{b_{1a}}\ \ T\boldsymbol{\omega}^T_{b_{1\omega}}]^T \in \mathbb{R}^{12}$ is the vector of a zero mean process noise with covariance matrix of $\mathbf{Q}^b = \mathrm{blkdiag}(\mathbf{0}, \mathbf{0}, T^2\sigma^2_{b_{1a}}\mathrm{I}, T^2\sigma^2_{b_{1\omega}}\mathrm{I})$. In $\boldsymbol{\omega}^b_k, \mathbf{0} \in \mathbb{R}^3$ while in $\mathbf{Q}^b, \mathbf{0}$ is a 3×3 zeros matrix.

Assuming $\mathbf{z}^I = [\mathbf{a}^T_m\ \boldsymbol{\omega}^T_m]^T$, the following measurement model is obtained:

$$\mathbf{z}^I_k = \mathbf{h}^d(\mathbf{x}^D_k) + H^b \mathbf{x}^B_k + \boldsymbol{\nu}^I_k. \tag{12}$$

Here $\mathbf{h}^d(\mathbf{x}^D_k) = [\mathbf{a}^T_\omega \mathbf{R}^\omega_b\ \boldsymbol{\omega}^T_b]^T$ and $\boldsymbol{\nu}^I_k = [\boldsymbol{\nu}^T_a\ \boldsymbol{\nu}^T_\omega]^T$ are 6×1 vectors and $\boldsymbol{\nu}^I$ is zero mean Gaussian measurement noise with the covariance matrix of $\mathbf{R}^I = \mathrm{blkdiag}(\sigma^2_{\nu_a}\mathrm{I}, \sigma^2_{\nu_\omega}\mathrm{I})$ and,

$$\mathbf{H}^b = \begin{bmatrix} \mathrm{I} & \mathbf{0} & \mathrm{I} & \mathbf{0} \\ \mathbf{0} & \mathrm{I} & \mathbf{0} & \mathrm{I} \end{bmatrix},$$

where I and $\mathbf{0}$ are 3×3 identity and zero matrices, respectively.

2) RANGE AND BEARING MEASUREMENT MODELS

LiDAR sensors measure the relative range and bearing to a landmark as depicted in Figure 2. By range measurement (r), a LiDAR can measure the distance from the sensor to the landmark and by bearing measurement, azimuth (α) and elevation (β) angles are measured.

For simplicity, it is assumed in this paper that the UAV local coordinate system and the coordinate system attached to the LiDAR are collocated with the location of the origin $[x\ y\ z]^T$ relative to the inertial frame. Besides, $\mathbf{m}_j = [x_j\ y_j\ z_j]^T$ is the location of the origin of jth landmark's frame relative to

the inertial frame. Thus the range and bearing measurement models are:

$$r_j = \sqrt{(x_j - x)^2 + (y_j - y)^2 + (z_j - z)^2},$$
$$\beta_j = \arcsin \frac{z_j - z}{\sqrt{(x_j - x)^2 + (y_j - y)^2 + (z_j - z)^2}},$$
$$\alpha_j = \arctan 2 \frac{y_j - y}{x_j - x}, \tag{13}$$

Let $\mathbf{z}^S = [\beta_j \ \alpha_j \ r_j]^T$. Then by adding the measurement noise, the measurement model for the LiDAR measuring the jth landmark, is:

$$\mathbf{z}_k^S = \mathbf{h}_j(\mathbf{x}_k^D, \mathbf{m}_j) + \boldsymbol{\nu}_j^S, \tag{14}$$

where $\mathbf{h}_j$ is described in (13), $\mathbf{x}_k^D$ is presented in (4), and $\boldsymbol{\nu}_j^S$ is a zero mean Gaussian measurement noise vector with the covariance matrix of $\mathbf{R}_j^S$. The inverse measurement model which gives the position of jth landmark obtained easily using Figure 2 is given by:

$$x_j = x + r_j \cos(\beta_j) \cos(\alpha_j).$$
$$y_j = y + r_j \cos(\beta_j) \sin(\alpha_j).$$
$$z_j = z + r_j \sin(\beta_j). \tag{15}$$

In fact, (15) represents $\mathbf{m}_j = \mathbf{h}_j^{-1}(\mathbf{x}_k^D, \mathbf{z}_k^S)$.

It is worth mentioning that at different time samples, different landmarks, $j = 1, 2, \ldots, L$, are observed. Therefore, index j changes in $\mathbf{z}_k^S$ at different time samples. In other words, for simplicity, it is assumed that only one landmark is observed at every sampling step. This assumption does not restrict the method to only one landmark observation at each sampling time and it can be easily extended to the general case of observing multiple landmarks at each sampling step.

III. SLAM PROBLEM

In this section, the problem of full-SLAM [3] is extended to solve the SLAM problem using the system dynamics in the presence of model noises and uncertainties, as well as sensory biases and drifts. In the full-SLAM the whole trajectory of the vehicle and the map given all the control inputs and all the measurements are estimated [5]. It is also assumed that the problem of data association [3] has been solved during this paper. In other words, the corresponding model of observed feature in the map is considered to be perfectly identified.

A. MODIFIED FastSLAM

In this section, the problem of FastSLAM [7] is extended to include both quadcopter uncertain and noisy dynamics, as well as the inertial sensors bias and drift models, fusing information from inertial and LiDAR sensors.

In the FastSLAM, Rao Bleackwellized Particle Filtering (RBPF) is employed to estimate both vehicle trajectory and landmark positions where each landmark is estimated with Extended Kalman Filters (EKF) and PFs are employed to generate particles only used for the trajectory. In this paper, it is proposed to incorporate not only the robot's dynamics but also the inverse sensory models in the particle generation process. The idea of using dynamics is for inertial sensory bias and drift isolation and identification and incorporating the inverse sensor model in the particle generation, makes the method robust to model mismatches.

The SLAM problem in this paper, is to determine the whole history of quadcopter states (which includes its path) $\mathbf{x}_{1:k}^D = \{\mathbf{x}_1^D, \ldots, \mathbf{x}_k^D\}$, sensory biases and drifts $\mathbf{x}_k^B$ and landmark locations $\mathbf{m} = \mathbf{m}_{1:L} = \{\mathbf{m}_1, \ldots, \mathbf{m}_L\}$, where L refers to the number of landmarks in the map, given control signal sequence $\mathbf{u}_{1:k} = \{\mathbf{u}_1, \ldots, \mathbf{u}_k\}$, inertial sensor measurement history $\mathbf{z}_{1:k}^I = \{\mathbf{z}_1^I, \ldots, \mathbf{z}_k^I\}$ and LiDAR measurement history $\mathbf{z}_{1:k}^S = \{\mathbf{z}_1^S, \ldots, \mathbf{z}_k^S\}$.

The goal of Bayesian estimation theory [31] is to compute posterior probability distribution. The posterior distribution, in our SLAM problem, is as follows:

$$p(\mathbf{m}_{1:L}, \mathbf{x}_{1:k}^D, \mathbf{x}_k^B | \mathbf{z}_{1:k}^I, \mathbf{z}_{1:k}^S, \mathbf{u}_{1:k-1}). \tag{16}$$

Due to the increased dimension of the state vector, because of the bias and drift augmentation, in this paper, more decomposition is performed, compared to normal FastSLAM, according to the posterior distribution function and Bayes' rule as follows.

Using Bayes' rule and the fact that $\mathbf{m}_{1:L}$ is statistically independent of $\mathbf{z}_{1:k}^I$ and $\mathbf{u}_{1:k-1}$, and moreover $\mathbf{x}_k^B$ is statistically independent of $\mathbf{z}_{1:k}^S$ and $\mathbf{u}_{1:k-1}$, one can conclude:

$$\begin{aligned} & p(\mathbf{m}_{1:L}, \mathbf{x}_{1:k}^D, \mathbf{x}_k^B | \mathbf{z}_{1:k}^I, \mathbf{z}_{1:k}^S, \mathbf{u}_{1:k-1}) \\ & \overset{\text{Bayes'}}{=} p(\mathbf{m}_{1:L} | \mathbf{x}_{1:k}^D, \mathbf{x}_k^B, \mathbf{z}_{1:k}^I, \mathbf{z}_{1:k}^S, \mathbf{u}_{1:k-1}) \\ & \quad \times p(\mathbf{x}_{1:k}^D, \mathbf{x}_k^B | \mathbf{z}_{1:k}^I, \mathbf{z}_{1:k}^S, \mathbf{u}_{1:k-1}) \ \text{Bayes' \& indep.} \\ & = p(\mathbf{m}_{1:L} | \mathbf{x}_{1:k}^D, \mathbf{z}_{1:k}^S) \\ & \quad \times p(\mathbf{x}_k^B | \mathbf{x}_{1:k}^D, \mathbf{z}_{1:k}^I) p(\mathbf{x}_{1:k}^D | \mathbf{z}_{1:k}^I, \mathbf{z}_{1:k}^S, \mathbf{u}_{1:k-1}), \end{aligned} \tag{17}$$

where, *indep.* refers to independencies of stochastic variables and $p(\mathbf{x}_{1:k}^D | \mathbf{z}_{1:k}^I, \mathbf{z}_{1:k}^S, \mathbf{u}_{1:k-1})$ can be estimated by means of a particle filter (PF) [32] in which probability distributions are approximated by a set of particles and their corresponding weights, $\{\mathbf{x}_{1:k}^{D,i}, w_k^i\}$, $i = 1, \ldots, N$. In other words:

$$p(\mathbf{x}_{1:k}^D | \mathbf{z}_{1:k}^I, \mathbf{z}_{1:k}^S, \mathbf{u}_{1:k-1}) \approx \sum_{i=1}^{N} w_k^i \delta(\mathbf{x}_{1:k}^D - \mathbf{x}_{1:k}^{D,i}), \tag{18}$$

where $\delta(.)$ is the Kronecker Delta function.

Then for each particle sequence, $\mathbf{x}_{1:k}^{D,i}$, a Kalman Filter (KF) and Extended Kalman Filter (EKF) are respectively employed to estimate $p(\mathbf{x}_k^B | \mathbf{x}_{1:k}^{D,i}, \mathbf{z}_{1:k}^I)$ and $p(\mathbf{m}_{1:L} | \mathbf{x}_{1:k}^{D,i}, \mathbf{z}_{1:k}^S)$, where:

$$p(\mathbf{x}_k^B | \mathbf{x}_{1:k}^{D,i}, \mathbf{z}_{1:k}^I) \sim \mathcal{N}(\mathbf{x}_k^B, \hat{\mathbf{x}}_{k|k}^{B,i}, \boldsymbol{\Sigma}_{k|k}^{B,i}), \tag{19}$$

where $\hat{\mathbf{x}}_{k|k}^{B,i}$ is the updated estimate of $\mathbf{x}_k^B$ given $\mathbf{x}_{1:k}^{D,i}$ and $\boldsymbol{\Sigma}_{k|k}^{B,i}$ is the corresponding updated covariance matrix. Moreover:

$$p(\mathbf{m}_{1:L} | \mathbf{x}_{1:k}^{D,i}, \mathbf{z}_{1:k}^S) = \prod_{j=1}^{L} p(\mathbf{m}_j | \mathbf{x}_{1:k}^{D,i}, \mathbf{z}_{1:k}^S), \tag{20}$$

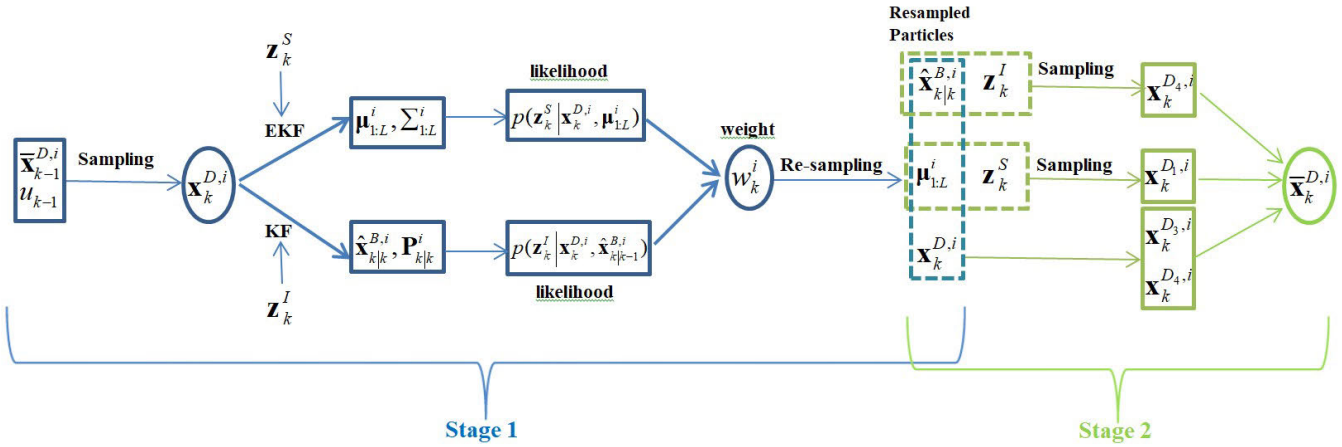

FIGURE 3. The schematic of proposed method for *i*th particle sequence.

where,

$$p(\mathbf{m}_j|\mathbf{x}^{D,i}_{1:k}, \mathbf{z}^S_{1:k}) \sim \mathcal{N}(\mathbf{m}_l, \boldsymbol{\mu}^i_j, \boldsymbol{\Sigma}^i_j), \tag{21}$$

where $\boldsymbol{\mu}^i_j$ and $\boldsymbol{\Sigma}^i_j$ are the mean vector and the covariance matrix of the estimated location of jth landmark given $\mathbf{x}^{D,i}_{1:k}$.

In the following, the sequential Monte Carlo (SMC) particle filtering is derived to approximate the posterior distribution of $p(\mathbf{x}^D_{1:k}|\mathbf{z}^I_{1:k}, \mathbf{z}^S_{1:k}, \mathbf{u}_{1:k-1})$. This is performed in two cascaded stages as explained in the following:

Stage 1: Using Bayes' rule one can conclude:

$$\begin{aligned} &p(\mathbf{x}^D_{1:k}|\mathbf{z}^I_{1:k}, \mathbf{z}^S_{1:k}, \mathbf{u}_{1:k-1}) \\ &\overset{\text{Bayes'}}{\propto} p(\mathbf{z}^I_k, \mathbf{z}^S_k|\mathbf{x}^D_{1:k}, \mathbf{z}^I_{1:k-1}, \mathbf{z}^S_{1:k-1}, \mathbf{u}_{1:k-1}) \\ &\quad \times p(\mathbf{x}^D_{1:k}|\mathbf{z}^I_{1:k-1}, \mathbf{z}^S_{1:k-1}, \mathbf{u}_{1:k-1}). \end{aligned} \tag{22}$$

Using the fact that the process model of (4) can be explained by $p(\mathbf{x}^D_k|\mathbf{x}^D_{k-1}, \mathbf{u}_{k-1})$, the following is obtained:

$$\begin{aligned} &p(\mathbf{x}^D_{1:k}|\mathbf{z}^I_{1:k-1}, \mathbf{z}^S_{1:k-1}, \mathbf{u}_{1:k-1}) \\ &\overset{\text{Bayes' \& indep.}}{=} p(\mathbf{x}^D_k|\mathbf{x}^D_{k-1}, \mathbf{u}_{k-1}) \\ &\quad \times p(\mathbf{x}^D_{1:k-1}|\mathbf{z}^I_{1:k-1}, \mathbf{z}^S_{1:k-1}, \mathbf{u}_{1:k-2}). \end{aligned} \tag{23}$$

The filtering is recursive, since the term $p(\mathbf{x}^D_{1:k-1}|\mathbf{z}^I_{1:k-1}, \mathbf{z}^S_{1:k-1}, \mathbf{u}_{1:k-2})$ is appeared. Moreover, due to statistical independency of $\mathbf{z}^I_k$ and $\mathbf{z}^S_k$, the following is obtained:

$$\begin{aligned} &p(\mathbf{z}^I_k, \mathbf{z}^S_k|\mathbf{x}^D_{1:k}, \mathbf{z}^I_{1:k-1}, \mathbf{z}^S_{1:k-1}, \mathbf{u}_{1:k-1}) \\ &= p(\mathbf{z}^I_k|\mathbf{x}^D_{1:k}, \mathbf{z}^I_{1:k-1}, \mathbf{z}^S_{1:k-1}, \mathbf{u}_{1:k-1}) \\ &\quad \times p(\mathbf{z}^S_k|\mathbf{x}^D_{1:k}, \mathbf{z}^I_{1:k-1}, \mathbf{z}^S_{1:k-1}, \mathbf{u}_{1:k-1}), \end{aligned} \tag{24}$$

where,

$$\begin{aligned} &p(\mathbf{z}^I_k|\mathbf{x}^D_{1:k}, \mathbf{z}^I_{1:k-1}, \mathbf{z}^S_{1:k-1}, \mathbf{u}_{1:k-1}) \\ &= \int p(\mathbf{z}^I_k|\mathbf{x}^D_{1:k}, \mathbf{z}^I_{1:k-1}, \mathbf{z}^S_{1:k-1}, \mathbf{u}_{1:k-1}, \mathbf{x}^B_k) \\ &\quad \times p(\mathbf{x}^B_k|\mathbf{x}^D_{1:k}, \mathbf{z}^I_{1:k-1}, \mathbf{z}^S_{1:k-1}, \mathbf{u}_{1:k-1}) d\mathbf{x}^B_k. \end{aligned} \tag{25}$$

Using the probabilistic measurement model of $\mathbf{z}^I_k$, that is $p(\mathbf{z}^I_k|\mathbf{x}^D_k, \mathbf{x}^B_k)$, and since $\mathbf{x}^B_k$ is statistically independent from $\mathbf{z}^D_{1:k-1}, \mathbf{z}^S_{1:k-1}$ and $\mathbf{u}_{1:k-1}$, (25) becomes:

$$\begin{aligned} &p(\mathbf{z}^I_k|\mathbf{x}^D_{1:k}, \mathbf{z}^I_{1:k-1}, \mathbf{z}^S_{1:k-1}, \mathbf{u}_{1:k-1}) \\ &\quad = \int p(\mathbf{z}^I_k|\mathbf{x}^D_k, \mathbf{x}^B_k) p(\mathbf{x}^B_k|\mathbf{x}^D_k, \mathbf{z}^I_{1:k-1}) d\mathbf{x}^B_k, \end{aligned} \tag{26}$$

where, $p(\mathbf{x}^B_k|\mathbf{x}^D_k, \mathbf{z}^I_{1:k-1})$ is the probabilistic prediction model for the bias and drift subsystem. This can be approximated by $\delta(\mathbf{x}^B_k - \hat{\mathbf{x}}^{B,i}_{k|k-1})$ for the ith particle sequence, in which $\hat{\mathbf{x}}^{B,i}_{k|k-1}$ is the estimation prediction of $\mathbf{x}^B_k$ in the KF algorithm for the ith particle sequence. Therefore, the observation likelihood of the ith particle sequence is, $p(\mathbf{z}^I_k|\mathbf{x}^{D,i}_k, \hat{\mathbf{x}}^{B,i}_{k|k-1})$.

Similarly,

$$\begin{aligned} &p(\mathbf{z}^S_k|\mathbf{x}^D_{1:k}, \mathbf{z}^I_{1:k-1}, \mathbf{z}^S_{1:k-1}, \mathbf{u}_{1:k-1}) \\ &= \int p(\mathbf{z}^S_k|\mathbf{x}^D_{1:k}, \mathbf{z}^I_{1:k-1}, \mathbf{z}^S_{1:k-1}, \mathbf{u}_{1:k-1}, \mathbf{m}_{1:L}) \\ &\quad \times p(\mathbf{m}_{1:L}|\mathbf{x}^D_{1:k}, \mathbf{z}^I_{1:k-1}, \mathbf{z}^S_{1:k-1}, \mathbf{u}_{1:k-1}) d\mathbf{m}_{1:L} \\ &\overset{\text{indep.}}{=} \int p(\mathbf{z}^S_k|\mathbf{x}^D_k, \mathbf{m}_{1:L}) p(\mathbf{m}_{1:L}|\mathbf{x}^D_{1:k-1}, \mathbf{z}^I_{1:k-1}) d\mathbf{m}_{1:L}. \end{aligned} \tag{27}$$

where, $p(\mathbf{m}_{1:L}|\mathbf{x}^D_{1:k-1}, \mathbf{z}^I_{1:k-1})$ is the posterior PDF of the map at previous time sample $k-1$ which can be approximated by $\delta(\mathbf{m}_{1:L} - \boldsymbol{\mu}^i_{1:L})$ for the ith particle sequence. $\boldsymbol{\mu}^i_{1:L}$ denotes the map estimation using the EKF algorithm for the ith particle sequence. Therefore, the observation likelihood of the ith particle sequence is, $p(\mathbf{z}^S_k|\mathbf{x}^{B,i}_k, \boldsymbol{\mu}^i_{1:L})$. Therefore, the unnormalized weight of the ith particle sequence before re-sampling is as follows:

$$w^i_k \propto w^i_{k-1} p(\mathbf{z}^S_k|\mathbf{x}^{D,i}_k, \boldsymbol{\mu}^i_{1:L}) p(\mathbf{z}^I_k|\mathbf{x}^{D,i}_k, \hat{\mathbf{x}}^{B,i}_{k|k-1}). \tag{28}$$

Stage 2: Using Bayes' rule and since $\mathbf{x}^D_{1:k-1}$ is independent of $\mathbf{z}^I_k$ and $\mathbf{z}^S_k$, one can conclude:

$$\begin{aligned} &p(\mathbf{x}^D_{1:k}|\mathbf{z}^I_{1:k}, \mathbf{z}^S_{1:k}, \mathbf{u}_{1:k-1}) \overset{\text{Bayes' \& indep.}}{} \\ &\quad = p(\mathbf{x}^D_k|\mathbf{x}^D_{1:k-1}, \mathbf{z}^I_{1:k}, \mathbf{z}^S_{1:k}, \mathbf{u}_{1:k-1}) \\ &\quad \times p(\mathbf{x}^D_{1:k-1}|\mathbf{z}^I_{1:k-1}, \mathbf{z}^S_{1:k-1}, \mathbf{u}_{1:k-2}). \end{aligned} \tag{29}$$

Applying again the Bayes' rule, it is concluded:

$$p(\mathbf{x}^D_k|\mathbf{x}^D_{1:k-1}, \mathbf{z}^I_{1:k}, \mathbf{z}^S_{1:k}, \mathbf{u}_{1:k-1})$$

TABLE 2. Pseudo code corresponding to the proposed self-calibration based FastSLAM.

Initialization: $\mathbf{x}_0^{D,i},\ w_0^i{}_{i=1}^N$.
At the time sample k:
Stage 1:
for $i = 1$ to N:
Step 0: Receive:

$$\{\bar{\mathbf{x}}_{1:k-1}^{D,i}, (\boldsymbol{\mu}_1^i, \boldsymbol{\Sigma}_1^i), ..., (\boldsymbol{\mu}_L^i, \boldsymbol{\Sigma}_L^i), (\hat{\mathbf{x}}_{k-1|k-1}^{B,i}, \mathbf{P}_{k-1|k-1}^i)\}$$

· **Step 1**: Generate particles for UAV's state vector using the first subsystem model:

$$\mathbf{x}_k^{D,i} \sim p(\mathbf{x}_k^D | \mathbf{x}_{k-1}^{D,i}, \mathbf{u}_{k-1}).$$

Step 2: Bias and drift estimation prediction (Table 4):

$$(\hat{\mathbf{x}}_{k|k-1}^{B,i}, \mathbf{P}_{k|k-1}^i) = KF - prediction(\hat{\mathbf{x}}_{k-1|k-1}^{B,i}, \mathbf{P}_{k-1|k-1}^i).$$

Step 3: Normalized weight computation:

$$w_k^i = \frac{p(\mathbf{z}_k^S | \mathbf{x}_k^{D,i}, \boldsymbol{\mu}_{1:L}^i) p(\mathbf{z}_k^I | \mathbf{x}_k^{D,i}, \hat{\mathbf{x}}_{k|k-1}^{B,i})}{\sum_{j=1}^N p(\mathbf{z}_k^S | \mathbf{x}_k^{D,j}, \boldsymbol{\mu}_{1:L}^j) p(\mathbf{z}_k^I | \mathbf{x}_k^{D,j}, \hat{\mathbf{x}}_{k|k-1}^{B,j})}.$$

Step 4: Bias and drift estimation update (Table 4):

$$(\hat{\mathbf{x}}_{k|k}^{B,i}, \mathbf{P}_{k|k}^i) = KF - update(\hat{\mathbf{x}}_{k|k-1}^{B,i}, \mathbf{P}_{k|k-1}^i, \mathbf{z}_k^I).$$

Step 5: Map generation:

For the observed landmark j:
if the landmark is observed for the first time **then** (use Table 3):

$$(\boldsymbol{\mu}_j^i, \boldsymbol{\Sigma}_j^i) = EKF - prediction(\mathbf{z}_k^S),$$

else

$$(\boldsymbol{\mu}_j^i, \boldsymbol{\Sigma}_j^i) = EKF - update(\boldsymbol{\mu}_j^i, \boldsymbol{\Sigma}_j^i, \mathbf{z}_k^S),$$

endif.
Leave all unobserved landmarks unchanged.
endfor.

Step 6: Re-sample the particles:

$$\{w_k^i, \mathbf{x}_{1:k}^{D,i}, (\boldsymbol{\mu}_1^i, \boldsymbol{\Sigma}_1^i), ..., (\boldsymbol{\mu}_L^i, \boldsymbol{\Sigma}_L^i), (\hat{\mathbf{x}}_{k|k}^{B,i}, \mathbf{P}_{k|k}^i)\}_{i=1}^N,$$

and obtain N equally weighted particles.
Step 7: Return:

$$\{\frac{1}{N}, \mathbf{x}_{1:k}^{D,i}, (\boldsymbol{\mu}_1^i, \boldsymbol{\Sigma}_1^i), ..., (\boldsymbol{\mu}_L^i, \boldsymbol{\Sigma}_L^i), (\hat{\mathbf{x}}_{k|k}^{B,i}, \mathbf{P}_{k|k}^i)\}_{i=1}^N.$$

Stage 2:
for $i = 1$ to N:
Step 8: Generate particles for UAV's position and angular velocity using the inverse sensor models:

$$\mathbf{x}_k^{D_4,i} \sim p(\mathbf{x}_k^{D_4} | \boldsymbol{\omega}_m, \hat{\mathbf{x}}_{k|k}^{B_1,i}),$$

$$\mathbf{x}_k^{D_1,i} \sim p(\mathbf{x}_k^{D_1} | \mathbf{z}_k^S, \boldsymbol{\mu}_{1:L}^i),$$

and, use the corresponding parts of $\mathbf{x}_k^{D,i}$ generated in Step 7 for $\mathbf{x}_k^{D_2,i}$ and $\mathbf{x}_k^{D_3,i}$.
endfor.

Step 9: Let $\bar{\mathbf{x}}_k^{D,i} = [(\mathbf{x}_k^{D_1,i})^T\ (\mathbf{x}_k^{D_2,i})^T\ (\mathbf{x}_k^{D_3,i})^T\ (\mathbf{x}_k^{D_4,i})^T]^T$ and **return**:

$$\{\frac{1}{N}, \bar{\mathbf{x}}_{1:k}^{D,i}, (\boldsymbol{\mu}_1^i, \boldsymbol{\Sigma}_1^i), ..., (\boldsymbol{\mu}_L^i, \boldsymbol{\Sigma}_L^i), (\hat{\mathbf{x}}_{k|k}^{B,i}, \mathbf{P}_{k|k}^i)\}_{i=1}^N.$$

TABLE 3. EKF algorithm for *j*th landmark and *i*th particle.

EKF-predict: If landmark j is observed for the first time:
- Initialize the landmark position estimate:

$$\boldsymbol{\mu}_j^i = \mathbf{h}_j^{-1}(\mathbf{x}_k^{D,i}, \mathbf{z}_k^S).$$

- Compute Jacobian of $\mathbf{h}_j(.)$ with respect to $\mathbf{m}_j$:

$$\mathbf{H}_j^i = \frac{d\mathbf{h}_j(\mathbf{m}_j, \mathbf{x}_k^D)}{d\mathbf{m}_j}|_{\mathbf{m}_j=\boldsymbol{\mu}_j^i, \mathbf{x}_k^D=\mathbf{x}_k^{D,i}}.$$

- Initialize the landmark estimate covariance:

$$\boldsymbol{\Sigma}_j^i = (\mathbf{H}_j^i)^{-1}\mathbf{R}_j^S((\mathbf{H}_j^i)^{-1})^T.$$

EKF-update:
- Compute measurement prediction:

$$\hat{\mathbf{z}}_k^S = \mathbf{h}_j(\mathbf{x}_k^{D,i}, \boldsymbol{\mu}_j^i).$$

- Compute Jacobian of $\mathbf{h}_j(.)$ with respect to $\mathbf{m}_j$:

$$\mathbf{H}_j^i = \frac{d\mathbf{h}_j(\mathbf{m}_j, \mathbf{x}_k^D)}{d\mathbf{m}_j}|_{\mathbf{m}_j=\boldsymbol{\mu}_j^i, \mathbf{x}_k^D=\mathbf{x}_k^{D,i}}$$

- Compute measurement covariance: $\mathbf{S}_j^i = \mathbf{H}_j^i\boldsymbol{\Sigma}_j^i(\mathbf{H}_j^i)^T + \mathbf{R}_j^S$.
- Compute Kalman gain: $\mathbf{K}_j^i = \boldsymbol{\Sigma}_j^i(\mathbf{H}_j^i)^T(\mathbf{S}_j^i)^{-1}$.
- Update estimation: $\boldsymbol{\mu}_j^i = \boldsymbol{\mu}_j^i + \mathbf{K}_j^i(\mathbf{z}_k^S - \hat{\mathbf{z}}_k^S)$.
- Update covariance: $\boldsymbol{\Sigma}_j^i = (\mathbf{I} - \mathbf{K}_j^i\mathbf{H}_j^i)\boldsymbol{\Sigma}_j^i$.

TABLE 4. KF algorithm for bias and drift estimation for *i*th particle.

KF-prediction:
- Initialize: $\hat{\mathbf{x}}_{0|0}^{B,i}$ and $\mathbf{P}_{0|0}^i$.
- Predict bias and drift estimate: $\hat{\mathbf{x}}_{k|k-1}^{B,i} = \mathbf{A}^b\hat{\mathbf{x}}_{k-1|k-1}^{B,i}$.
- Predict covariance: $\mathbf{P}_{k|k-1}^i = \mathbf{A}^b\mathbf{P}_{k-1|k-1}^i\mathbf{A}^b + \mathbf{Q}^b$.

KF-update:
- Compute measurement: $\bar{\mathbf{z}}_k^I = \mathbf{z}_k^I - \mathbf{h}^d(\mathbf{x}_k^{D,i})$.

If k=1
- Reinitialize: $\hat{\mathbf{x}}_{k|k}^{B,i} = (\mathbf{H}_b^T\mathbf{H}_b)^{-1}\mathbf{H}_b^T\bar{\mathbf{z}}_k^I$.

else:
- Compute measurement prediction: $\hat{\bar{\mathbf{z}}}_k^I = \mathbf{H}_b\hat{\mathbf{x}}_{k|k-1}^{B,i}$.
- Compute measurement covariance:

$$\mathbf{S}_b^i = \mathbf{H}^b\mathbf{P}_{k|k-1}^i(\mathbf{H}^b)^T + \mathbf{R}^I.$$

- Compute Kalman gain: $\mathbf{K}_2^i = \mathbf{P}_{k|k-1}^i(\mathbf{H}^b)^T(\mathbf{S}_b^i)^{-1}$.
- Update estimation: $\hat{\mathbf{x}}_{k|k}^{B,i} = \hat{\mathbf{x}}_{k|k-1}^{B,i} + \mathbf{K}_2^i(\bar{\mathbf{z}}_k^I - \hat{\bar{\mathbf{z}}}_k^I)$.
- Update covariance: $\mathbf{P}_{k|k}^i = (\mathbf{I} - \mathbf{K}_2^i\mathbf{H}^b)\mathbf{P}_{k|k-1}^i$.

$$\overset{\text{Bayes'}}{\propto} p(\mathbf{x}_{1:k-1}^D | \mathbf{x}_k^D, \mathbf{z}_{1:k}^I, \mathbf{z}_{1:k}^S, \mathbf{u}_{1:k-1}) \times p(\mathbf{x}_k^D | \mathbf{z}_{1:k}^I, \mathbf{z}_{1:k}^S, \mathbf{u}_{1:k-1}) \quad (30)$$

Now, let $\mathbf{x}_k^D$ decompose to $\mathbf{x}_k^{D_1} = [x\ y\ z]^T$, $\mathbf{x}_k^{D_2} = [\dot{x}\ \dot{y}\ \dot{z}]^T$, $\mathbf{x}_k^{D_3} = [\phi\ \ \theta\ \ \psi]^T$ and $\mathbf{x}_k^{D_4} = [p\ \ q\ \ r]^T$. Due to the discretized version of the system dynamics, see (4), $\mathbf{x}_k^D$ is a function of $\mathbf{x}_{k-1}^D$ and $\mathbf{u}_{k-1}$ and thus $\mathbf{x}_k^{D_1}$ to $\mathbf{x}_k^{D_4}$ are statistically independent. Therefore, one can conclude:

$$\begin{aligned} p(\mathbf{x}_k^D | \mathbf{z}_{1:k}^I, \mathbf{z}_{1:k}^S, \mathbf{u}_{1:k-1}) &= p(\mathbf{x}_k^{D_1} | \mathbf{z}_{1:k}^I, \mathbf{z}_{1:k}^S, \mathbf{u}_{1:k-1}) \\ &\times p(\mathbf{x}_k^{D_2}, \mathbf{x}_k^{D_3} | \mathbf{z}_{1:k}^I, \mathbf{z}_{1:k}^S, \mathbf{u}_{1:k-1}) \\ &\times p(\mathbf{x}_k^{D_4} | \mathbf{z}_{1:k}^I, \mathbf{z}_{1:k}^S, \mathbf{u}_{1:k-1}). \end{aligned} \quad (31)$$

Moreover:

$$p(\mathbf{x}_k^{D_4}|\mathbf{z}_{1:k}^I, \mathbf{z}_{1:k}^S, \mathbf{u}_{1:k-1}) = \int p(\mathbf{x}_k^{D_4}|\mathbf{z}_{1:k}^I, \mathbf{z}_{1:k}^S, \mathbf{u}_{1:k-1}, \mathbf{x}_k^{B_1}) \times p(\mathbf{x}_k^{B_1}|\mathbf{z}_{1:k}^I, \mathbf{z}_{1:k}^S, \mathbf{u}_{1:k-1}) d\mathbf{x}_k^{B_1}, \quad (32)$$

where $\mathbf{x}_k^{B_1} = [\mathbf{b}_{0\omega}^T \ \mathbf{b}_{1\omega}^T]^T$.

Considering the measurement model of the gyroscope in (9), the gyroscope's probabilistic inverse sensor model is $p(\mathbf{x}_k^{D_4}|\boldsymbol{\omega}_m, \mathbf{x}_k^{B_1})$. Due to the statistical independence:

$$p(\mathbf{x}_k^{D_4}|\mathbf{z}_{1:k}^I, \mathbf{z}_{1:k}^S, \mathbf{u}_{1:k-1}, \mathbf{x}_k^{B_1}) \overset{\text{indep.}}{=} p(\mathbf{x}_k^{D_4}|\boldsymbol{\omega}_m, \mathbf{x}_k^{B_1}). \quad (33)$$

Moreover:

$$p(\mathbf{x}_k^{B_1}|\mathbf{z}_{1:k}^I, \mathbf{z}_{1:k}^S, \mathbf{u}_{1:k-1}) \overset{\text{indep.}}{=} p(\mathbf{x}_k^{B_1}|\mathbf{z}_{1:k}^I) = \int p(\mathbf{x}_k^{B_1}|\mathbf{x}_{1:k}^D, \mathbf{z}_{1:k}^I) p(\mathbf{x}_{1:k}^D|\mathbf{z}_{1:k}^I, \mathbf{z}_{1:k}^S, \mathbf{u}_{1:k-1}) d\mathbf{x}_{1:k}^D, \quad (34)$$

where $p(\mathbf{x}_k^{B_1}|\mathbf{x}_{1:k}^D, \mathbf{z}_{1:k}^I)$ can be approximated by $\delta(\mathbf{x}_k^{B_1} - \hat{\mathbf{x}}_{k|k}^{B_1,i})$ for the ith particle sequence, in which $\hat{\mathbf{x}}_{k|k}^{B_1,i}$ is the estimation of $\mathbf{x}_k^{B_1}$ using the KF algorithm and is then resampled in first stage: Substituting (33) and (34) in (32), the ith particle, $\mathbf{x}_k^{D_4,i}$, is generated using the proposal distribution of $p(\mathbf{x}_k^{D_4}|\boldsymbol{\omega}_m, \hat{\mathbf{x}}_{k|k}^{B_1,i})$.

Similarly, considering the LiDAR measurement model explained in (13) and (14), the proposal distribution to generate particles for $\mathbf{x}_k^{D_1}$ using the LiDAR inverse sensor model is $p(\mathbf{x}_k^{D_1}|\mathbf{z}_k^S, \boldsymbol{\mu}_{1:L}^i)$ where $\boldsymbol{\mu}_{1:L}^i$ denotes the map estimation using the EKF algorithm corresponding to the ith particle sequence in first stage.

Since $\mathbf{x}_k^{D_2}$ and $\mathbf{x}_k^{D_3}$ do not appear directly in the measurement models, the generated particles in the previous stage, $\mathbf{x}_k^{D_2,i}$ and $\mathbf{x}_k^{D_3,i}$, are employed, as there exist no inverse sensor model to generate them in second stage.

$$p(\mathbf{x}_k^{D_2}, \mathbf{x}_k^{D_3}|\mathbf{z}_{1:k}^I, \mathbf{z}_{1:k}^S, \mathbf{u}_{1:k-1}) = \int p(\mathbf{x}_k^{D_2}, \mathbf{x}_k^{D_3}|\mathbf{x}_{1:k-1}^D, \mathbf{z}_{1:k}^S, \mathbf{u}_{1:k-1}) \times p(\mathbf{x}_{1:k-1}^D|\mathbf{z}_{1:k-1}^I, \mathbf{z}_{1:k-1}^S, \mathbf{u}_{1:k-2}) d\mathbf{x}_{1:k-1}^D. \quad (35)$$

It is worth mentioning that, since the resampled ith paths $\mathbf{x}_{1:k}^{D,i}$ and $\mathbf{x}_{1:k-1}^{D,i}$ are respectively employed in (34) and (35) to approximate the posterior distributions, $p(\mathbf{x}_{1:k}^D|\mathbf{z}_{1:k}^I, \mathbf{z}_{1:k}^S, \mathbf{u}_{1:k-1})$ and $p(\mathbf{x}_{1:k-1}^D|\mathbf{z}_{1:k-1}^I, \mathbf{z}_{1:k-1}^S, \mathbf{u}_{1:k-2})$, the corresponding weight of each particle is $\frac{1}{N}$ after normalization.

Finally, let $\bar{\mathbf{x}}_k^{D,i} = [(\mathbf{x}_k^{D_1,i})^T \ (\mathbf{x}_k^{D_2,i})^T \ (\mathbf{x}_k^{D_3,i})^T \ (\mathbf{x}_k^{D_4,i})^T]^T$. The general framework of the method is depicted in Figure 3 and the corresponding pseudo code of the method is presented in Table 2.

It is obvious from Figure 3 that the proposed modified FastSLAM is a cascaded 2-stage approach where in first stage the typical FastSLAM 1.0 is extended to include bias estimation and in second stage, using the results of first stage, some generated particles are regenerated using the inverse sensor models to overcome model uncertainties. Table 3 and Table 4 elaborate the KF and EKF algorithms embedded in the main part (Table 2). It is worth mentioning that a reinitialization step using the inverse measurement model of $\bar{\mathbf{z}}_k^I$ is added to the KF algorithm in Table 4 to correct the initialization process of the bias subsystem's estimate.

IV. SIMULATION RESULTS

In this section, simulation results, obtained through a co-simulation between MATLAB-2019b and Gazebo in the Robot Operating System (ROS), are presented to evaluate the performance of the proposed cascaded SLAM approach in which the states of the quadcopter UAV system, as well as inertial sensors biases and drifts and landmarks' positions are estimated. The simulation environment we are using here is validated based on the experimental data as reported in [33]. Therefore, here we use this platform as a realistic environment to evaluate and test the performance of the proposed algorithm. The algorithm developed in this way can be easily deployed into the hardware for a real experimental test and it brings the advantage of iterative development and tuning of the algorithm without the risk of damaging expensive hardware. Also, ROS and Gazebo allow for the parallel simulation of other sensors such as inertial and visual sensors in complex 3D environments similar to what happens in the laboratory settings. This makes ROS an ideal environment for practical evaluation of the algorithm developed in this paper.

A. SETTING UP THE ROS ENVIRONMENT

The quadcopter model is simulated in Gazebo in ROS using a modified version of the Hector quadcopter ROS package [33]. A Velodyne VLP-16 3D LiDAR is also simulated using Gazebo and the collected pointcloud data is visualized in RViz. A ROS package is implemented to allow features placed in the Gazebo simulation environment to be extracted [34]. The package takes the collected pointcloud data and uses it to identify features, in this case poles, placed around the environment. These features are then identified as landmarks and assigned an ID. Figure 4 shows a graph of the ROS nodes and topics used in the co-simulation. An ellipse identifies a ROS node while a rectangle identifies a ROS topic through which information travels. The node *ROSLINK_ALLINONE_FINAL_87413* is the Simulink model which subscribes to the topics */ground_truth/state* and */feature_extraction_node/keypoints*, and publishes topics */PosForce* and */AngleForce* which carry control signals to ROS. Figure 5 shows a screenshot from the Gazebo displaying the quadcopter and poles simulated in Gazebo on the left, and the 3D pointcloud data and estimated landmark positions in RViz on the right. The feature extraction package is able to be implemented on an on-board computer using real pointcloud data collected from a 3D LiDAR, making it possible for this approach to be implemented on a real system.

The Gazebo simulation environment allows a more comprehensive simulation to run with non-linearities and disturbances, providing a more accurate-to-life scenario. The

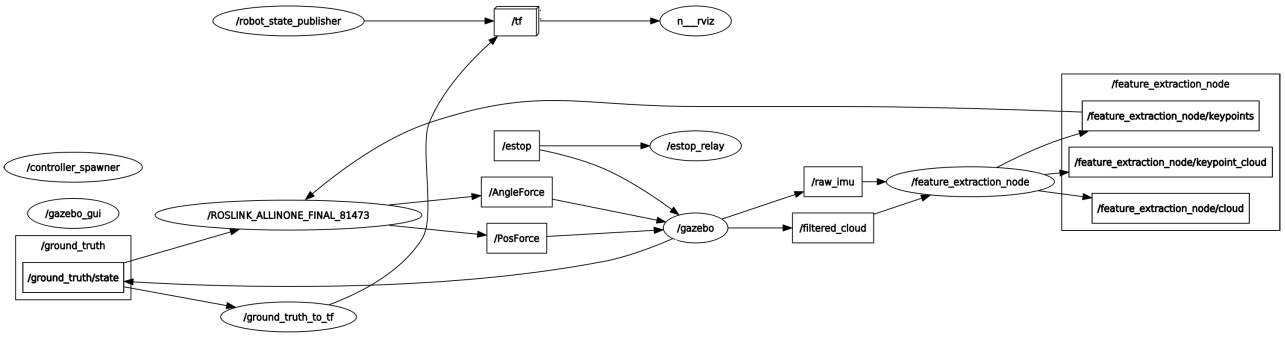

FIGURE 4. Graph showing ROS topics connections between nodes.

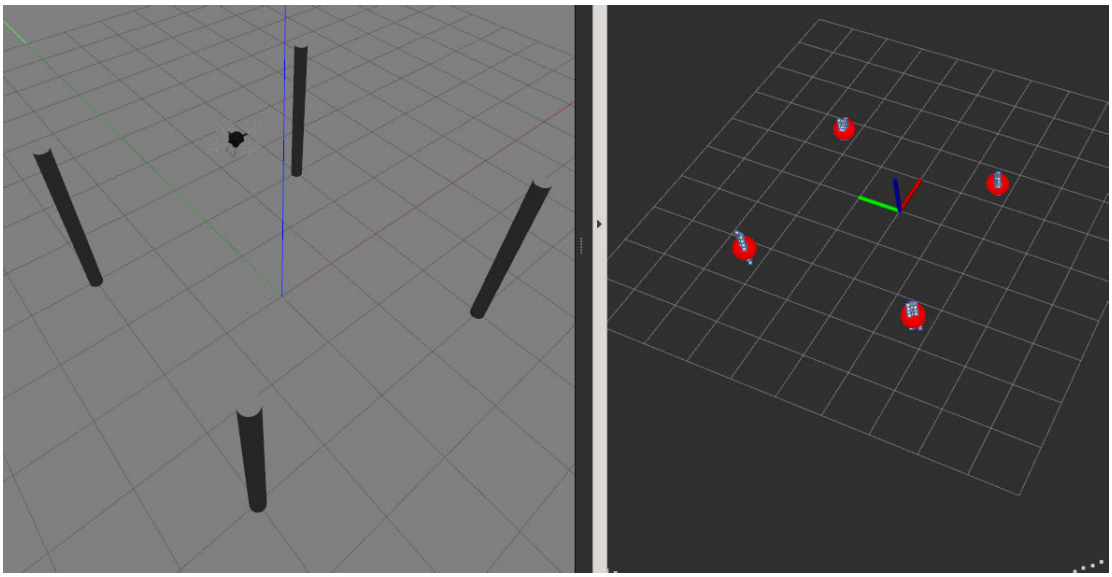

FIGURE 5. Screenshot of Gazebo and RViz simulation environments showing quadcopter model and estimated landmark positions.

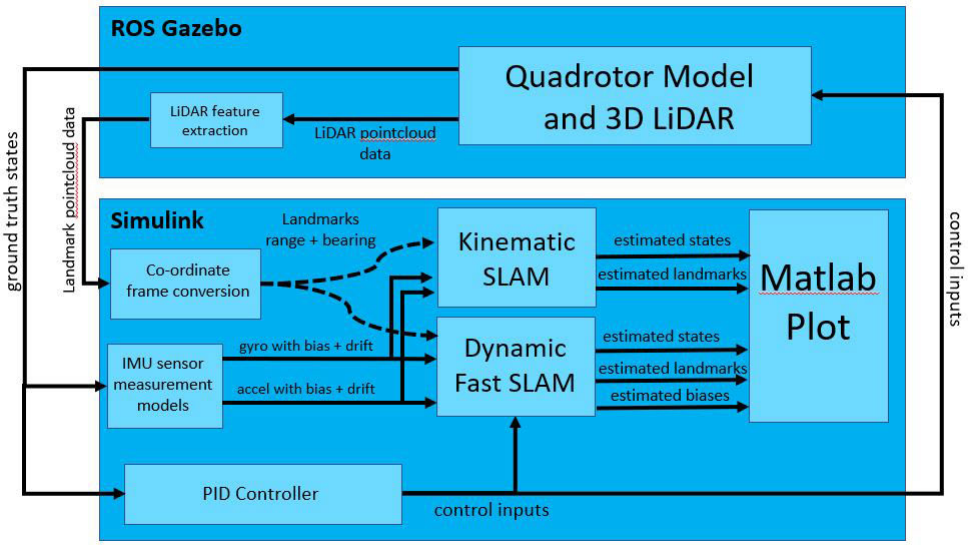

FIGURE 6. Flow diagram showing co-simulation between ROS and Simulink.

quadcopter is stabilized using a nested PID controller in Simulink, with control signals u_1, u_2, u_3 and u_4 transferred to the Gazebo simulation of the quadcopter model. The ground truth states from Gazebo can then be read in the

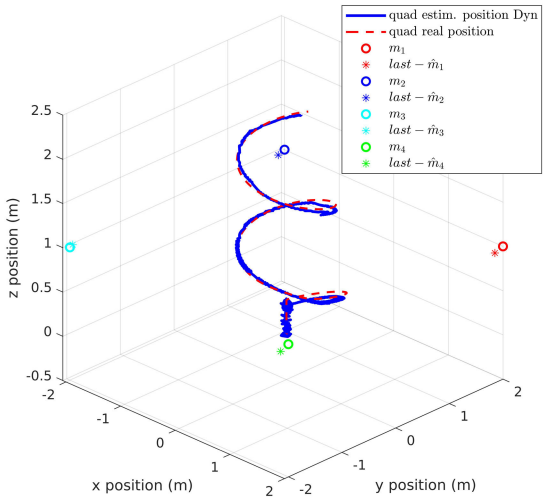

FIGURE 7. Real and estimated position of quadcopter and landmarks.

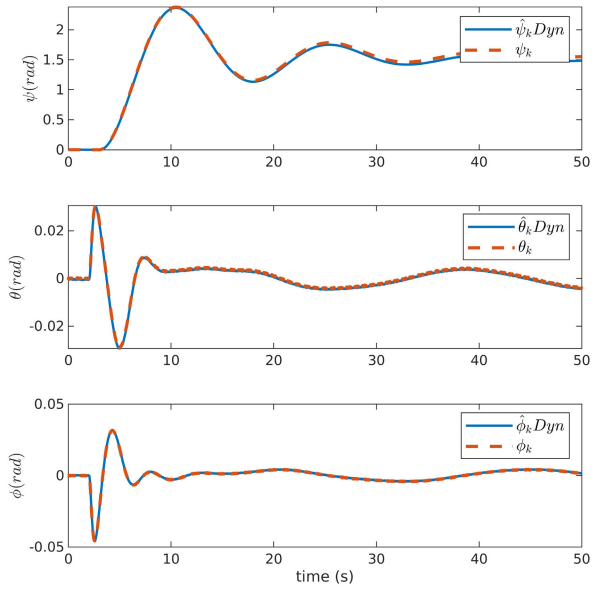

FIGURE 8. Real and estimated Euler angles.

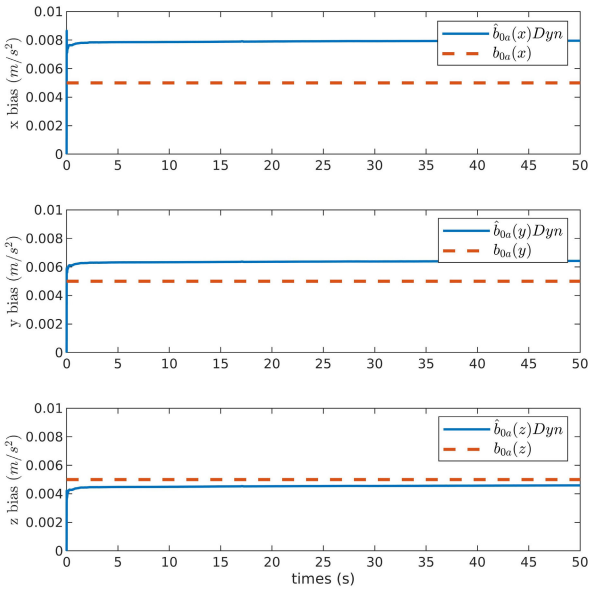

FIGURE 9. Real and estimated accelerometer biases.

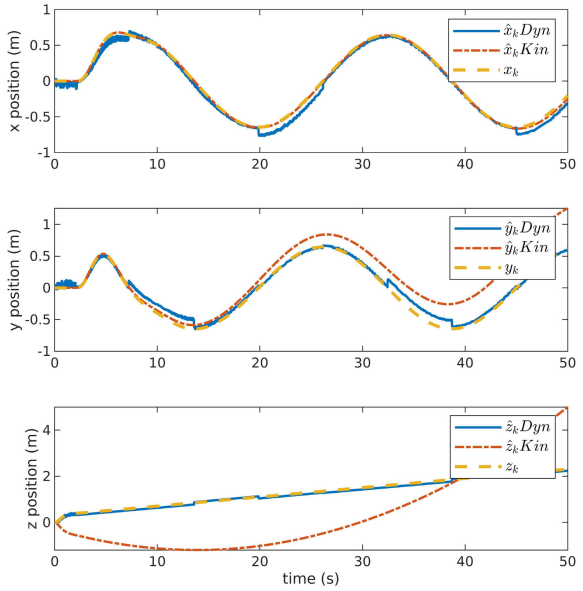

FIGURE 10. Real, dynamics, and kinematics-based estimation of quadcopter's position (biases: 0.0005 m/s^2 and 0.0005 *rad*/*s*).

Simulink environment. The simulations are synchronised using a Gazebo Pacer block in Simulink with a time step of $T = 0.01s$. The co-simulation environment is depicted in the flow diagram in Figure 6.

The physical parameters of the quadcopter system are assumed as $I_{xx} = 0.01152\ kgm^2$, $I_{yy} = 0.01152\ kgm^2$, $I_{zz} = 0.0218\ kgm^2$, and $m = 1.477\ kg$. However, these parameters are amenable to change due to installation of the LiDAR sensor. One way to overcome the problem of parameter uncertainty in the model, is to use RLS algorithm to estimate the parameters of moment of inertia online. For example, a continuous time RLS [28], can be used by converting the three equations in (1) to a regression vector. However, as it will be shown in the next section, the proposed dynamic-based estimation technique is robust enough with respect to the changes in the inertial values and hence no RLS algorithm is used here in the sequel.

B. EVALUATION OF THE ALGORITHM

The performance of the proposed algorithm is evaluated and compared against the kinematic SLAM in this subsection. To capture all four landmarks, the quadcopter is controlled to move about z-axis in a helical trajectory. The number of particles is selected as $N = 500$. Figure 7 depicts the estimated and real position of the quadcopter as well as the landmarks

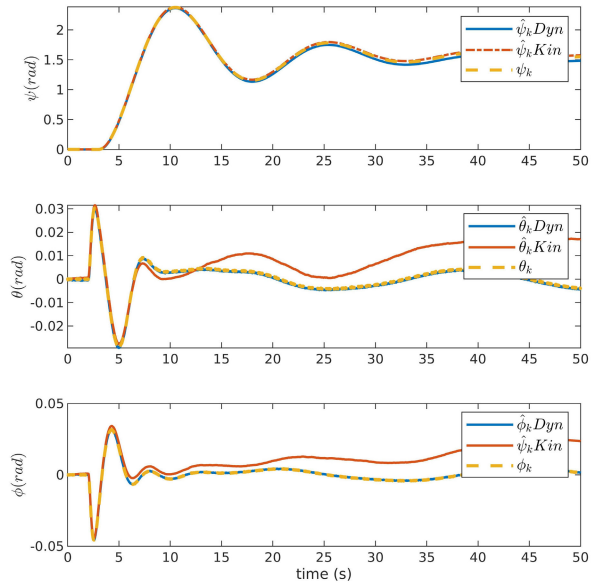

FIGURE 11. Real, dynamics, and kinematics-based estimation of quadcopter's Euler angles (biases: 0.0005 m/s^2 and 0.0005 rad/s).

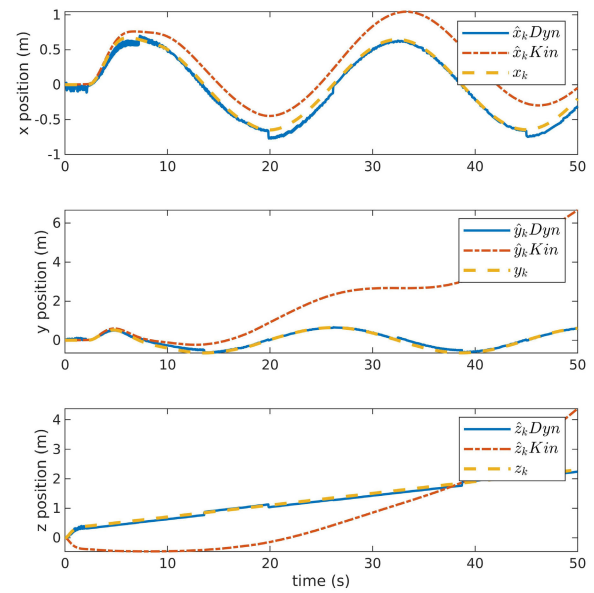

FIGURE 12. Real, dynamics, and kinematics-based estimation of quadcopter's position (biases: 0.005 m/s^2 and 0.005 rad/s).

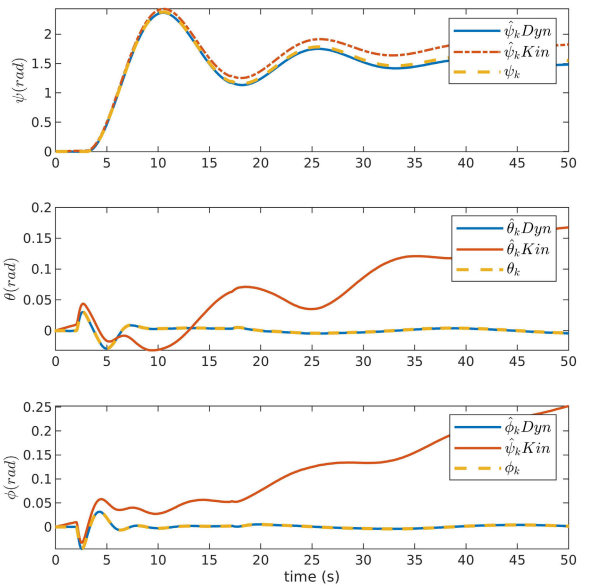

FIGURE 13. Real, dynamics, and kinematics-based estimation of quadcopter's Euler angles (biases: 0.005 m/s^2 and 0.005 rad/s).

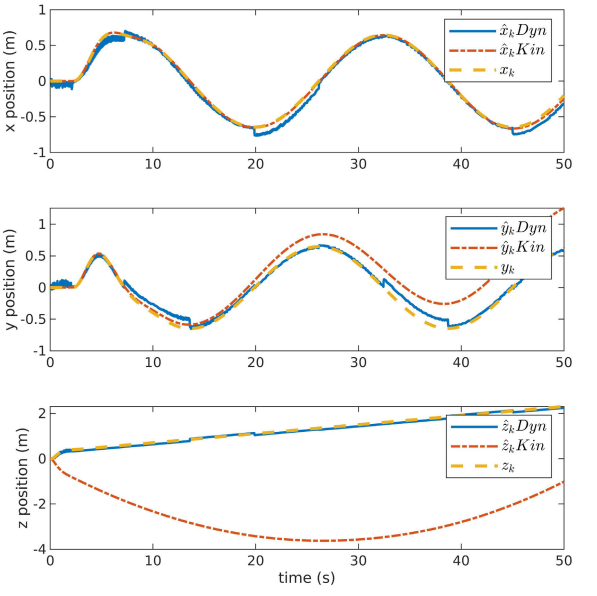

FIGURE 14. Real, dynamics, and kinematics-based estimation of quadcopter's position (biases: 0.0005 m/s^2 and 0.0005 rad/s) without RLS and increased inertia estimate.

in 3D space. Four landmarks are placed in the environment in the form of poles with the following coordinates, where the center of each pole is tracked as a landmark:

$$m_1 = [2\ 2\ -1]^T,\ \ m_2 = [-2\ 2\ -1]^T,$$
$$m_3 = [-2\ -2\ -1]^T,\ \ m_4 = [2\ -2\ -1]^T.$$

Real and estimated Euler angles are depicted in Figure 8. The bias of 0.005 m/s^2 is considered for the accelerometer in three axes of x, y and z. Figure 9 depicts the estimation of the biases compared with the real ones. In Figures 10 and 11, the proposed method is compared with kinematics based FastSLAM without calibration of inertial sensors when the biases of 0.0005 m/s^2 and 0.0005 rad/s are considered for the accelerometer and the gyroscopes in three axes of x, y and z, respectively. The results are also shown for the biases of 0.005 m/s^2 and 0.005 rad/s in Figures 12 and 13. It is clear from the figures that when the sensory faults are increased, using dynamics instead of kinematics and compensating sensory biases and drifts, improve the estimation significantly. Finally, to assess the robustness of the algorithm, it was tested with biases of 0.0005 m/s^2 and 0.0005 rad/s and with parametric uncertainty for inertia. The

new parameters were $I_{xx} = 0.0288\ kgm^2, I_{yy} = 0.0288\ kgm^2$, $I_{zz} = 0.0545\ kgm^2$, which are a 150% increase on their actual values and no RLS algorithm is used to be estimated. Figure 14 shows that dynamics based FastSLAM remained robust, while kinematic SLAM had increased deviation in the estimation of the z position of the quadcopter. That means on top of the uncertainty added by the LiDAR the proposed estimation algorithm has shown a good robustness property with respect to parametric uncertainties in the UAV model.

V. CONCLUSION

In this paper, a modified two-stage FastSLAM algorithm was proposed for a quadcopter UAV system in which inertial sensory biases and drifts are estimated during the SLAM process. In this method, the quadcopter's dynamics with augmented bias and drift models instead of a commonly used kinematics makes the estimation and elimination of such sensory faults possible. For this purpose, the FastSLAM 1.0 approach was developed by factorization of the full posterior distribution into conditional landmark, conditional bias and drift distributions, and the posterior distribution over the robot path. This led to an extended FastSLAM 1.0 in the first stage in which both KF and EKF were embedded in the PF to estimate biases, drifts and landmarks' positions, respectively, and then the particles' the corresponding weights were updated. In the second stage, the PF was extended to include measurements in the particle generation to overcome the inaccuracies caused by the model mismatches.

Simulation results, achieved through a co-simulation between MATLAB-2019b and Gazebo in the Robot Operating System (ROS), were presented to evaluate the performance of the proposed cascaded SLAM approach in a realistic setting. The quadcopter model was simulated in Gazebo in ROS using the modified version of the Hector quadcopter ROS package. The robustness of the proposed estimation algorithm is evaluated against uncertainties when the quadcopter's moments of inertia parameters are changed significantly.

REFERENCES

[1] C. Debeunne and D. Vivet, "A review of visual-LiDAR fusion based simultaneous localization and mapping," *Sensors*, vol. 20, no. 7, p. 2068, 2020.

[2] N. S. Nokhodberiz, H. Nemati, and A. Montazeri, "Event-triggered based state estimation for autonomous operation of an aerial robotic vehicle," *IFAC-PapersOnLine*, vol. 52, no. 13, pp. 2348–2353, 2019.

[3] J.-A. Fernández-Madrigal, *Simultaneous Localization and Mapping for Mobile Robots: Introduction and Methods: Introduction and Methods*. Hershey, PA, USA: IGI Global, 2012.

[4] S. Saeedi, M. Trentini, M. Seto, and H. Li, "Multiple-robot simultaneous localization and mapping: A review," *J. Field Robot.*, vol. 33, no. 1, pp. 3–46, 2016.

[5] G. Bresson, Z. Alsayed, L. Yu, and S. Glaser, "Simultaneous localization and mapping: A survey of current trends in autonomous driving," *IEEE Trans. Intell. Veh.*, vol. 2, no. 3, pp. 194–220, Sep. 2017.

[6] M. G. Dissanayake, P. Newman, H. F. Durrant-Whyte, S. Clark, and M. Csorba, "An experimental and theoretical investigation into simultaneous localisation and map building," in *Experimental Robotics VI* (Lecture Notes in Control and Information Sciences), vol. 250. London, U.K.: Springer, 2000, pp. 265–274.

[7] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "Fastslam: A factored solution to the simultaneous localization and mapping problem," in *Proc. AAAI/IAAI*, 2002, Art. no. 593598.

[8] A. Doucet, N. De Freitas, and N. Gordon, "An introduction to sequential Monte Carlo methods," in *Sequential Monte Carlo Methods in Practice* (Statistics for Engineering and Information Science). New York, NY, USA: Springer, 2001, pp. 3–14.

[9] *FastSLAM 2.0*, Springer, Berlin, Germany, 2007, pp. 63–90.

[10] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "Fastslam 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges," in *Proc. IJCAI*, vol. 3, 2003, pp. 1151–1156.

[11] D. Törnqvist, T. B. Schön, R. Karlsson, and F. Gustafsson, "Particle filter SLAM with high dimensional vehicle model," *J. Intell. Robotic Syst.*, vol. 55, nos. 4–5, pp. 249–266, Aug. 2009.

[12] M. Bryson and S. Sukkarieh, "Building a robust implementation of bearing-only inertial SLAM for a UAV," *J. Field Robot.*, vol. 24, nos. 1–2, pp. 113–143, 2007.

[13] Z. Sjanic, M. Skoglund, T. Schön, and F. Gustafsson, *Solving SLAM Problem for Unmanned Aerial Vehicles Using Smoothed Estimates*. Linköping, Sweden: Linköping Univ. Electronic Press, 2010.

[14] J. Kim and S. Sukkarieh, "Real-time implementation of airborne inertial-SLAM," *Robot. Auto. Syst.*, vol. 55, no. 1, pp. 62–71, Jan. 2007.

[15] N. Sadeghzadeh-Nokhodberiz, J. Poshtan, A. Wagner, E. Nordheimer, and E. Badreddin, "Distributed observers for pose estimation in the presence of inertial sensory soft faults," *ISA Trans.*, vol. 53, no. 4, pp. 1307–1319, Jul. 2014.

[16] H.-K. Lee, K. Choi, J. Park, and H. Myung, "Self-calibration of gyro using monocular SLAM for an indoor mobile robot," *Int. J. Control, Autom. Syst.*, vol. 10, no. 3, pp. 558–566, Jun. 2012.

[17] W. Huang and H. Liu, "Online initialization and automatic camera-IMU extrinsic calibration for monocular visual-inertial SLAM," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 5182–5189.

[18] P. Geneva, K. Eckenhoff, Y. Yang, and G. Huang, "LIPS: LiDAR-inertial 3D plane SLAM," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 123–130.

[19] J. Kelly and G. S. Sukhatme, "Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration," *Int. J. Robot. Res.*, vol. 30, no. 1, pp. 56–79, 2011.

[20] L. D. L. Perera, W. S. Wijesoma, and M. D. Adams, "SLAM with joint sensor bias estimation: Closed form solutions on observability, error bounds and convergence rates," *IEEE Trans. Control Syst. Technol.*, vol. 18, no. 3, pp. 732–740, May 2010.

[21] M. Euston and J. Kim, "Rao-blackwellised inertial-slam with partitioned vehicle subspace," in *Proc. Australas. Conf. Robot. Autom. (ACRA)*, Brisbane, QLD, Australia: Citeseer, Dec. 2007, pp. 1–6.

[22] L. Huang, B. He, and T. Zhang, "An autonomous navigation algorithm for underwater vehicles based on inertial measurement units and sonar," in *Proc. 2nd Int. Asia Conf. Informat. Control, Autom. Robot. (CAR)*, Mar. 2010, pp. 311–314.

[23] J. Kim, "Aerial inertial-SLAM: Progresses and future challenges," in *Proc. 12th Int. Conf. Ubiquitous Robots Ambient Intell. (URAI)*, Oct. 2015, pp. 212–213.

[24] R. Mur-Artal and J. D. Tardós, "Visual-inertial monocular SLAM with map reuse," *IEEE Robot. Automat. Lett.*, vol. 2, no. 2, pp. 796–803, Apr. 2017.

[25] X. Mu, J. Chen, Z. Zhou, Z. Leng, and L. Fan, "Accurate initial state estimation in a monocular visual–inertial SLAM system," *Sensors*, vol. 18, no. 2, p. 506, Feb. 2018.

[26] F. Bonin-Font, M. Massot-Campos, P. Negre-Carrasco, G. Oliver-Codina, and J. Beltran, "Inertial sensor self-calibration in a visually-aided navigation approach for a micro-AUV," *Sensors*, vol. 15, no. 1, pp. 1825–1860, Jan. 2015.

[27] H. Skeppström Lehto and R. Hedlund, "Impact of vehicle dynamics modelling on feature based slam for autonomous racing," M.S. thesis, Ind. Eng. Manage., TRITA ITM-EX, KTH Stockholm, Sweden, 2019, p. 133.

[28] K. J. Åström and B. Wittenmark, *Adaptive Control*. Chelmsford, MA, USA: Courier Corporation, 2013.

[29] T. Luukkonen, *Modelling and Control of Quadcopter*, vol. 22. Espoo, Finland: Independent Research Project in Applied Mathematics, 2011.

[30] J. Poshtan, A. Wagner, E. Nordheimer, and E. Badreddin, "Cascaded Kalman and particle filters for photogrammetry based gyroscope drift and robot attitude estimation," *ISA Trans.*, vol. 53, pp. 524–532, Mar. 2014.

[31] S. M. Kay, *Fundamentals of Statistical Signal Processing*. Upper Saddle River, NJ, USA: Prentice-Hall, 1993.
[32] A. Doucet and A. M. Johansen, "A tutorial on particle filtering and smoothing: Fifteen years later," *Handbook Nonlinear Filtering*, vol. 12, nos. 656–704, p. 3, 2009.
[33] J. Meyer, A. Sendobry, S. Kohlbrecher, U. Klingauf, and O. von Stryk, "Comprehensive simulation of quadrotor UAVS using ROS and Gazebo," in *Proc. 3rd Int. Conf. Simulation, Modeling Program. Auto. Robots (SIMPAR)*, 2012, pp. 400–411.
[34] *Github—Gavlab/Feature_extraction: A ROS Package for Using PCL for LIDAR Feature Extraction*. Accessed: Aug. 28, 2021. [Online]. Available: https://github.com/GAVLab/feature_extraction

NARGESS SADEGHZADEH-NOKHODBERIZ received the B.Sc. and M.Sc. degrees in control engineering from Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran, in 2006 and 2008, respectively, and the Ph.D. degree in control engineering from Iran University of Science and Technology, Tehran, in collaboration with the Automation Laboratory, Heidelberg University, Heidelberg, Germany, in September 2014.

She was with the Department of Engineering, Islamic Azad University Central Tehran Branch, Tehran, as an Assistant Professor, from 2015 to 2019. She was as a Visiting Scholar at the Department of Electrical Engineering, Qatar University, Doha, Qatar, in 2016, and the Department of Engineering, Lancaster University, Lancaster, U.K., in 2018. Since 2019, she has been an Assistant Professor with the Department of Electrical and Computer Engineering, Qom University of Technology, Qom, Iran. Her research interests include control systems theory, Monte Carlo state estimation, cyber-physical systems and networked control systems, robot control, mobile robot localization and mapping (SLAM), fault diagnosis, and distributed computing for control system applications.

AYDIN CAN was born in Merseyside, U.K., in 1996. He received the M.E. degree in mechanical engineering from Lancaster University, in 2019, where he is currently pursuing the Ph.D. degree with the Department of Engineering. His doctoral research is investigating the use of a multi-agent network of UAVs as a mobile sensor networks for remote characterization of nuclear legacy sites. He is also currently pursuing the Ph.D. degree in industry with The Centre for Innovative Nuclear Decommissioning (CINDe) at the National Nuclear Laboratory Workington facility in the U.K.

His research interests include multi-agent systems, non-linear control, cyber-physical systems, and robotics.

RUSTAM STOLKIN received the bachelor's and master's degrees in engineering from Oxford University, and the Ph.D. degree in robot vision undertaken between University College London and U.K. imaging industry. He is the Chair of Robotics with the University of Birmingham, U.K. He is also a Royal Society Industry Fellow in robotics and founded the U.K. National Centre for Nuclear Robotics, in 2017. He is an Interdisciplinary Engineer with diverse research interests, although mainly focuses on robotics. His main research interests include vision and sensing, robotic grasping and manipulation, robotic vehicles, human–robot interaction, AI, and machine learning.

ALLAHYAR MONTAZERI received the B.S. degree in electrical engineering from Tehran University, Tehran, Iran, in 2000, and the M.Sc. and Ph.D. degrees in electrical engineering from Iran University of Science and Technology, Tehran, in 2002 and 2009, respectively.

Between 2010 and 2011, he joined Fraunhofer Institute, Germany, as a Research Fellow, and then carried on his research with Fraunhofer IDMT and the Control Engineering Group, Ilmenau University, Germany, from 2011 to 2013. He has been a Visiting Research Scholar with the Control Engineering Group, ETH Zürich, Switzerland, and the Chemical Engineering Group, Norwegian University of Science and Technology (NTNU), Trondheim, Norway. Since 2013, he has been an Assistant Professor with the Engineering Department, Lancaster University, U.K. His research interests include a wide range of areas on control theory and digital signal processing. Particularly, he is interested in adaptive signal processing and control, robust control, linear and nonlinear system identification, estimation theory, and evolutionary computing and optimization with applications in active noise and vibration control systems and robotics.

Dr. Montazeri was a recipient of the European Research Consortium on Informatic and Mathematic (ERCIM) and Humboldt Research Awards, in 2010 and 2011, respectively. He is also a fellow of Higher Education Academy. His research is funded by different councils and industries in U.K., such as the Engineering and Physical Research Council, Sellafield Ltd., the National Nuclear Laboratory, and the Nuclear Decommissioning Authority. He is currently serving with the IFAC Technical Committees "Adaptive and Learning Systems" and "Modeling, Identification, and Signal Processing."