Tightly-Coupled Factor Graph Formulation For Radar-Inertial Odometry

Jan Michalczyk[1], Julius Quell[2], Florian Steidle[2], Marcus G. Müller[2] and Stephan Weiss[1]

***Abstract*— In this paper, we present a Radar-Inertial Odometry (RIO) method based on the nonlinear optimization of factor graphs in a sliding window fashion. Our method makes use of a light-weight, low-power, inexpensive and commonly available hardware enabling easy deployment on small Unmanned Aerial Vehicles (UAV)s. We keep the state estimation problem bounded by employing partial marginalization of the oldest states, rendering the method real-time capable. We compare the implemented approach to the state-of-the-art multi-state Extended Kalman Filter (EKF)-based method in a *one-to-one* fashion. That is, we implemented in a single custom C++ RIO framework both estimation back-ends with all other parts shared and thus identical for a fair direct comparison. In the real-world flight experiments, we compare the two methods and show that both perform similarly in terms of accuracy when the linearization point is not far from the true state. Upon wrong initialization, the factor graph approach heavily outperforms the EKF approach. We also acknowledge that the influence of undetected outliers can overwhelm the inherent benefits of the nonlinear optimization approach leading to the insight that the estimator front-end has an important (and often underestimated) role in the overall performance. The open source code and datasets can be found here: `https://github.com/aau-cns/aaucns_rio`.**

I. Introduction

Achieving accurate spatial awareness in Global Navigation Satellite System (GNSS)-denied environments is a key component of reliably operating autonomous UAVs in many scenarios. This problem is commonly approached by the fusion of measurements from a combination of sensors within a state estimation framework. Recently, fusing Frequency Modulated Continuous Wave (FMCW) radar and Inertial Measurement Unit (IMU) measurements has gained popularity in the UAVs autonomy research. A setup composed of these two sensors offers robustness against low-visibility environments thanks to the properties of electromagnetic waves used in radars [1], [2], and allows the IMU drift reduction, enabling accurate ego-motion estimation even in conditions challenging to camera or LiDAR-based setups.

Several kinds of FMCW radars have been used in the context of autonomous navigation. The most common ones are scanning radars [3], [4], [5], [6] and System-on-Chip (SoC) radars [7], [8], [9], [10], [1], [11], [2], [12], [13]. Scanning radars are bulky and expensive mechanically rotating sensors. After performing a 360 ° scan they provide polar images of the environment with a high angular resolution. They typically do not provide the relative velocity (Doppler) information. SoC radars are usually much smaller in size and require less power. SoC radars output distance, relative velocity, azimuth (and sometimes elevation) angles of reflecting points in the environment in the form of a 4D pointcloud (3D position and Doppler velocity). Their accuracy and resolution vary broadly depending on the antenna array characteristics and on-chip processing algorithms. Using the millimeter-wave technology in automotive industry [14], [15], [16] brought about the miniaturization of the radar sensors and boosted their accuracy. This in turn, paved the way for using them onboard small UAVs for fusion with the IMU sensors.

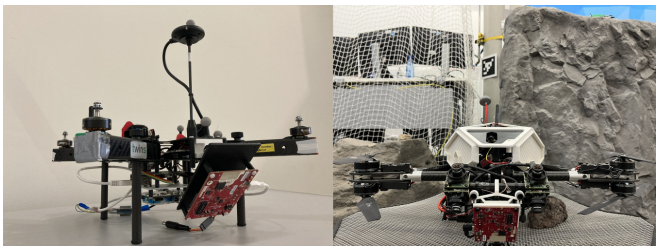

Fig. 1. Experimental platforms used in this work. CNS-UAV on the left and DLR's ARDEA-X on the right. The red chip mounted at 45 ° inclination on each platform is the TI AWR1843BOOST FMCW radar, which outputs highly noisy and sparse 4D pointclouds (3D points and Doppler velocities).

In this work, we present a novel RIO method based on the nonlinear optimization of factor graphs [17] allowing the estimation of the full 6DoF state of a small UAV using only IMU sensor and a light-weight, inexpensive, low-power Texas Instruments AWR1843BOOST FMCW SoC radar (see Fig. 1) in unknown and unprepared environments. The optimization problem underlying the state estimation task is maintained computationally tractable in real-time by employing a sliding window of states with the partial marginalization of oldest states using the Schur Complement technique. In our formulation we construct tightly-coupled radar factors from the distances to the 3D points matched between subsequent radar scans, instantaneous relative (Doppler) velocities and the distances to persistent landmarks. Tight coupling allows the construction of factors from single measurements, thus bypassing the necessity of computing pose increments from noisy and sparse radar pointclouds using a method such as ICP, which in such case is prone to fail [18]. In an attempt to make an exact *one-*

[1]Authors are with the Control of Networked Systems Group, University of Klagenfurt, Austria {firstname.lastname}@ieee.org

[2]Authors are with the Institute of Robotics and Mechatronics, German Aerospace Center (DLR), Münchener Straße 20, 82234 Wessling, Germany {firstname.lastname}@dlr.de

This research received funding from the Austrian Ministry of Climate Action and Energy (BMK) under the grant agreement 880057 (CARNIVAL).

Pre-print version, ©IEEE. DOI: 10.1109/IROS58592.2024.10801945.

to-one comparison with the state-of-the-art multi-state EKF-based RIO approach in [7], we implement both methods in a single custom C++ framework where all front-end features such as point matching, measurement trails construction, velocity-based point pruning based on RANSAC etc. are shared with only estimation back-ends changing. To our knowledge it is the first such comparison of these two common estimation back-ends in the RIO context. Our main contributions are:

- Tightly-coupled formulation of a sliding window factor graph-based RIO making maximal use of all the noisy measurements from a light-weight, inexpensive SoC FMCW radar sensor to correct the IMU drift.
- Efficient real-time capable implementation thanks to the employment of the sliding window paradigm with partial marginalization of the oldest states.
- *One-to-one* comparison with a state-of-the-art multi-state EKF-based RIO [7] in a single custom C++ framework made open-source to the community with the present paper.
- Validation in real-world flight experiments on two different UAV platforms to show the robustness and portability of the presented method.

This paper is organized as follows. Section II reviews the recent related work in the domain of RIO. In Section III, we broadly describe our RIO algorithm. In Subsection III-A we introduce the notation and definitions used across the paper. Subsection III-B outlines the overview of the estimator. In Subsection III-C we describe all factors used in our method except for the prior factor which is defined in the Subsection III-D. In Section IV, we explain the experiments and evaluation conducted in order to demonstrate and validate the proposed method and compare it with the multi-state EKF-based RIO. Finally, we present conclusions in Section V.

II. Related Work

Among the approaches to RIO we can mainly distinguish methods based either on filtering or nonlinear optimization. We briefly review the most important work in both branches. In [12] and [19] EKF-based RIO is presented in which no scan matching is performed, only current Doppler velocity measurements are used in the update step of the filter. The exhibited drift is mitigated by the use of a pressure sensor and in [20] still the same method is augmented with the Manhattan world assumptions to limit the yaw drift. Another filter-based RIO method is presented in [21] where an iterated EKF is used in a full SLAM framework. The use of an expensive high-end industry-grade SoC radar allows the authors to perform loop closures thus attaining high accuracy. The approach in [13] uses loosely-coupled Unscented Kalman Filter (UKF) to fuse the pose computed using Normal Distributions Transform (NDT) with the IMU measurements. The presented system attains high accuracy. Nevertheless, it is demonstrated in 2D environments with low-dynamics trajectories.

Within RIO methods employing the nonlinear optimization, the authors in [2] arrange two SoC radar sensors in an orthogonal manner in order to acquire strong reflections from more than one direction. Special casing destined for harsh environments is designed to house the sensor suite composed of the radars and an IMU. As the estimation back-end the authors use a pose graph composed of custom instantaneous linear velocity factors which they solve with the GTSAM package [22]. Interestingly, they do not fuse IMU measurements as separate factors but use them to initialize the orientation in the velocity factor. The authors do not mention any marginalization strategy which is crucial for obtaining a good prior in the factor graph. In [1] a RIO system is described in which nonlinear optimization is solved over a moving window of states to estimate the 3D ego-velocity of an UAV. The approach is demonstrated as effective in low-visibility conditions. Also here the authors do not mention how the oldest states are marginalized out and the method is focused on estimating the ego-velocity rather than the full 6DoF state of the system.

RIO approaches exist which make use of deep learning. In [4] a method is developed which combines unsupervised deep learning for features extraction with the factor graph optimization for state estimation. In [5] deep learning is used to learn keypoints for scan matching and delta-pose computation. Both above mentioned methods use scanning radars unsuitable for small UAVs.

III. Tightly-Coupled Factor Graph Formulation For Radar-Inertial State Estimation

A. Notation And Definitions

Names of reference frames are capitalized and calligraphic, e.g., $\{\mathcal{I}\}$ for IMU. A pose between the reference frames $\mathcal{A}$ and $\mathcal{B}$ is defined as ${}^{\mathcal{A}}\mathbf{T}_{\mathcal{B}} = \begin{bmatrix} {}^{\mathcal{A}}\mathbf{R}_{\mathcal{B}} & {}^{\mathcal{A}}\mathbf{p}_{\mathcal{B}} \\ \mathbf{0}^{\mathsf{T}} & 1 \end{bmatrix} \in \mathrm{SE}(3)$, with $\mathbf{R} \in \mathrm{SO}(3)$ and $\mathbf{p} \in \mathbb{R}^3$. The transformation of a coordinate vector ${}^{\mathcal{C}}_{\mathcal{C}}\mathbf{p}_{P_1}$ pointing from the origin of the reference frame $\mathcal{C}$ to a point P_1, expressed in $\mathcal{C}$, can be transformed into the frame $\mathcal{A}$ by $\begin{bmatrix} {}^{\mathcal{A}}_{\mathcal{A}}\mathbf{p}_{P_1} \\ 1 \end{bmatrix} = {}^{\mathcal{A}}\mathbf{T}_{\mathcal{C}} \begin{bmatrix} {}^{\mathcal{C}}_{\mathcal{C}}\mathbf{p}_{P_1} \\ 1 \end{bmatrix}$ (read as ${}^{from}_{in}\mathbf{x}_{to}$). Rotations are stored as unit quaternions $\bar{\mathbf{q}} \in \mathrm{SO}(3)$ with $\|\bar{\mathbf{q}}\| = 1$ allowing a direct mapping between rotation matrices and unit Hamiltonian quaternions by ${}^{\mathcal{A}}\mathbf{R}_{\mathcal{B}} = \mathbf{R}\{{}^{\mathcal{A}}\bar{\mathbf{q}}_{\mathcal{B}}\} \in \mathrm{SO}^3$ and ${}^{\mathcal{A}}\bar{\mathbf{q}}_{\mathcal{B}} = \bar{\mathbf{q}}\{{}^{\mathcal{A}}\mathbf{R}_{\mathcal{B}}\}$ [23]. $\mathbf{I}$ is the identity matrix. For vectors and block matrices, semicolons and colons improve the readability such that $[\mathbf{A};\mathbf{B}] \equiv \begin{bmatrix} \mathbf{A} \\ \mathbf{B} \end{bmatrix}$ and $[\mathbf{A},\mathbf{B}] \equiv \begin{bmatrix} \mathbf{A} & \mathbf{B} \end{bmatrix}$.

We define the state variables of our system as follows:

$$\begin{aligned} \mathbf{x}_I &= \left[{}^{\mathcal{G}}\mathbf{p}_{\mathcal{I}}; {}^{\mathcal{G}}\bar{\mathbf{q}}_{\mathcal{I}}; {}^{\mathcal{G}}\mathbf{v}_{\mathcal{I}}; \mathbf{b}_{\mathbf{a}}; \mathbf{b}_{\boldsymbol{\omega}}\right] \\ \mathbf{x}_L &= \left[{}^{\mathcal{G}}\mathbf{p}_{\mathcal{L}}\right] \\ \mathbf{X} &= [\mathbf{x}_{I_1}; \ldots; \mathbf{x}_{I_N}; \mathbf{x}_{L_1}; \ldots; \mathbf{x}_{L_M}] \end{aligned} \tag{1}$$

with the IMU state $\mathbf{x}_I$ and a state of a persistent landmark $\mathbf{x}_L$. ${}^{\mathcal{G}}\mathbf{p}_{\mathcal{I}}$, ${}^{\mathcal{G}}\mathbf{v}_{\mathcal{I}}$, and ${}^{\mathcal{G}}\bar{\mathbf{q}}_{\mathcal{I}}$ are the position, velocity, and orientation of the IMU/body frame $\{\mathcal{I}\}$ with respect to the navigation frame $\{\mathcal{G}\}$, respectively. $\mathbf{b}_{\boldsymbol{\omega}}$ and $\mathbf{b}_{\mathbf{a}}$ are the measurement biases of the gyroscope and accelerometer, respectively. ${}^{\mathcal{G}}\mathbf{p}_{\mathcal{L}}$ define the position of a persistent landmark

$\{\mathcal{L}\}$ with respect to the navigation frame $\{\mathcal{G}\}$. $\mathbf{X}$ is the set of all states contained within a single sliding window. In order for the optimizer to solve the problem we must compute the jacobians of observation models with respect to the error-state which we define as follows:

$$\begin{aligned}\tilde{\mathbf{x}}_I &= \left[{}^{\mathcal{G}}\tilde{\mathbf{p}}_{\mathcal{I}}; {}^{\mathcal{G}}\tilde{\boldsymbol{\theta}}_{\mathcal{I}}; {}^{\mathcal{G}}\tilde{\mathbf{v}}_{\mathcal{I}}; \tilde{\mathbf{b}}_{\mathbf{a}}; \tilde{\mathbf{b}}_{\boldsymbol{\omega}}\right] \\ \tilde{\mathbf{x}}_L &= \left[{}^{\mathcal{G}}\tilde{\mathbf{p}}_{\mathcal{L}}\right]\end{aligned} \tag{2}$$

For translational components, e.g., the position, the error is defined as ${}^{\mathcal{G}}\tilde{\mathbf{p}}_{\mathcal{I}} = {}^{\mathcal{G}}\hat{\mathbf{p}}_{\mathcal{I}} - {}^{\mathcal{G}}\mathbf{p}_{\mathcal{I}}$, while for rotations/quaternions it is defined as $\tilde{\bar{\mathbf{q}}} = \hat{\bar{\mathbf{q}}}^{-1} \otimes \bar{\mathbf{q}} = \left[1; \frac{1}{2}\tilde{\boldsymbol{\theta}}\right]$, with $\otimes$ and $\tilde{\boldsymbol{\theta}}$ being quaternion product and small angle approximation, respectively.

B. Estimator Overview

In our RIO approach at every iteration of the estimator an optimization problem is formulated as a nonlinear factor graph over a sliding window of N successive IMU states corresponding to time instants at which radar measurements were taken and a set of L persistent landmarks. We depict

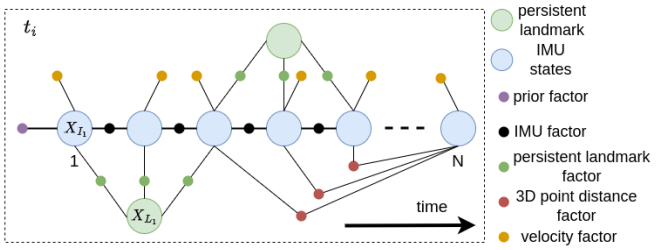

Fig. 2. Snapshot of the sliding window of states and measurements and the corresponding factor graph. Note the tightly-coupled nature of the graph, in which single measurements corresponding to Doppler velocities of 3D points, matched 3D points between radar scans and matched persisted landmarks are used to construct each factor. The tight coupling allows for maximal exploitation of sensor information. Note that the number of factors in the graph is only illustrative - in a real graph in our system there are many more factors involved.

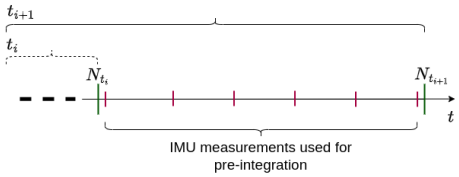

Fig. 3. Initialization of the newest IMU state in the sliding window. Dashed and solid overbraces mark the sliding window before and after the shift forwards which occurs upon acquiring a new radar measurement (green vertical lines). The new IMU state $\mathbf{x}_{I_N}(t_{i+1})$ in the window will be initialized with the prediction from the previous solution $\mathbf{x}_{I_N}(t_i)$ using the pre-integrated new IMU measurements (marked with red vertical lines) between the time instants of the previous and the current radar measurements.

the representation of a single factor graph corresponding to one sliding window of states in the Fig. 2. The edges in the graph represent factors and nodes the estimated states. Each time a new radar measurement is received we form a new graph and solve it to obtain the estimate of $\mathbf{X}$. To accommodate the new IMU state in the graph, we first marginalize out the oldest one and use it to form the new prior (see Section III-D). We populate the new graph with factors corresponding to the IMU states which remained after the marginalization and the persistent landmarks seen from them. Then, we initialize all states in the window $\mathbf{X}$ with their previous solutions. The new IMU state in $\mathbf{X}$ is initialized from the last IMU state in the previous solution by applying the delta-motion from the pre-integrated IMU measurements from the timespan between the previous measurement (corresponding to the last estimated state) and the current one (see Fig. 3). We solve the graph using Levenberg-Marquardt algorithm with the GTSAM package [22].

The overall cost function for the proposed factor graph is as follows:

$$\mathbf{X}^* = \underset{\mathbf{X}}{\operatorname{argmin}}\, e_I + e_P + e_D + e_V + e_L \tag{3}$$

where, e_I, e_P, e_D, e_V, e_L are the cost contributions from all of the IMU pre-integration, prior, radar 3D distance, velocity, persistent landmarks factors present in the graph, respectively.

C. Factors

In factor graph formulations, factors represent probabilistic constraints on the variables involved in the estimation and are obtained from measurements or prior knowledge. To define a factor we typically define a probabilistic measurement model constraining a subset of the state variables and upon creation, supply the corresponding measurement. Moreover, for solvers using the "*lift-solve-retract*" paradigm [24] we must provide a jacobian of the model with respect to the error-state defined in the tangent space of the state manifold. In Fig. 2, we show all types of factors included in our factor graph approach. IMU pre-integration factor constrains two estimated IMU states and we construct it from all the IMU measurements obtained between the two constrained states. Using IMU pre-integration factor is necessary since the IMU measurements come at high frequency and not pre-integrating them would lead to a huge number of variables in the optimization. We use the factor formulated in [24].

Our tightly-coupled relative velocity factor is a unary factor, which means that it introduces constraints on the subset of the variables in only one IMU state. Hence, in the Fig. 2 these factors have connections only to one node representing the state at which the velocity measurements were taken. As seen in the Eq. 4, the constrained state variables are ${}^{\mathcal{G}}\bar{\mathbf{q}}_{\mathcal{I}}$ and ${}^{\mathcal{G}}\mathbf{v}_{\mathcal{I}}$. The factor expresses the projection of the current robot ego-velocity transformed into the radar frame onto the direction vector pointing towards the corresponding 3D point:

$$\begin{aligned}{}^{\mathcal{R}}\mathbf{v}_{\mathcal{P}_i} = -\frac{\mathbf{r}^{\mathsf{T}}}{\|\mathbf{r}\|}\big({}^{\mathcal{I}}\mathbf{R}_{\mathcal{R}}^{\mathsf{T}}\,{}^{\mathcal{G}}\mathbf{R}_{\mathcal{I}}^{\mathsf{T}}\,{}^{\mathcal{G}}\mathbf{v}_{\mathcal{I}} + \\ {}^{\mathcal{I}}\mathbf{R}_{\mathcal{R}}^{\mathsf{T}}\left({}_{\mathcal{I}}\boldsymbol{\omega} \times {}_{\mathcal{I}}^{\mathcal{I}}\mathbf{p}_{\mathcal{R}}\right)\big)\end{aligned} \tag{4}$$

where $\mathbf{r} = {}^{\mathcal{R}}\mathbf{p}_{\mathcal{P}_i}$ is the 3D point detected in the current scan, ${}_{\mathcal{I}}\boldsymbol{\omega}$ is the current angular velocity of the IMU in the IMU

frame, and ${}^{\mathcal{G}}\mathbf{v}_{\mathcal{I}}$ is the current linear velocity of the IMU in the navigation frame. In order to reject outliers, we use Dynamic Covariance Scaling (DCS) robust kernel function [25] in this factor.

In order to further constrain the graph variables and make maximal use of the rich information provided by the radar, we build tightly-coupled factors for persistent landmarks and 3D point distance measurements organized in measurements trails. As we aim at an exact *one-to-one* comparison with the [7], we use the exact same code for 3D point matching, obtaining measurement trails and persistent landmarks as in [7], [8].

We construct 3D point distance factors from a set of point trails which contain a history of continuous detections of the same 3D points (see [7] for details). For all points in a trail we use the IMU states to transform all points ${}^{\mathcal{R}}\mathbf{p}_{\mathcal{P}_j}^{t_p}$ from the trail history at time instance t_p, where $p = 1, \ldots, V$ and V is the length of the matched trail, to the current radar reference frame:

$$\begin{aligned} {}^{\mathcal{R}}\mathbf{p}'^{t_p}_{\mathcal{P}_j} = {}^{\mathcal{I}}\mathbf{R}_{\mathcal{R}}^{\mathsf{T}} \Big(-{}^{\mathcal{I}}_{\mathcal{I}}\mathbf{p}_{\mathcal{R}} + ({}^{\mathcal{G}}\mathbf{R}_{\mathcal{I}}^{t_c})^{\mathsf{T}} \big(-{}^{\mathcal{G}}\mathbf{p}_{\mathcal{I}}^{t_c} + \\ {}^{\mathcal{G}}\mathbf{p}_{\mathcal{I}}^{t_p} + {}^{\mathcal{G}}\mathbf{R}_{\mathcal{I}}^{t_p} \left({}^{\mathcal{I}}_{\mathcal{I}}\mathbf{p}_{\mathcal{R}} + {}^{\mathcal{I}}\mathbf{R}_{\mathcal{R}}\, {}^{\mathcal{R}}\mathbf{p}_{\mathcal{P}_j}^{t_p} \right) \big) \Big) \end{aligned} \tag{5}$$

where ${}^{\mathcal{I}}\mathbf{R}_{\mathcal{R}}$ and ${}^{\mathcal{I}}\mathbf{p}_{\mathcal{R}}$ is the constant pose (orientation and position) of the radar frame with respect to the IMU frame. ${}^{\mathcal{G}}\mathbf{R}_{\mathcal{I}}^{\{t_c,t_p\}}$ and ${}^{\mathcal{G}}\mathbf{p}_{\mathcal{I}}^{\{t_c,t_p\}}$ are the IMU orientation and position corresponding to the trail history element at time t_p and current radar scan at t_c, with respect to the navigation frame $\{\mathcal{G}\}$. For factor construction we use the distance to the transformed matched point:

$$d_{\mathcal{P}_j} = \left\| {}^{\mathcal{R}}\mathbf{p}'^{t_p}_{\mathcal{P}_j} \right\| \tag{6}$$

where $d_{\mathcal{P}_j}$ is the distance to a single point j in the matched trail history ${}^{\mathcal{R}}\mathbf{p}'^{t_p}_{\mathcal{P}_j}$ at t_p aligned to the current radar pose at t_c.

We also use persistent landmarks to build factors which introduce constraints between the landmarks and the IMU states from which these landmarks have been seen (Fig. 2). Promotion of trails with sufficiently long history detection to persistent landmarks is described in details in [7]. Measurement model used in the factor is:

$$\mathbf{l}'_m = {}^{\mathcal{R}}_{\mathcal{R}}\mathbf{p}_{\mathcal{L}_m} = {}^{\mathcal{I}}\mathbf{R}_{\mathcal{R}}^{\mathsf{T}} \big({}^{\mathcal{G}}\mathbf{R}_{\mathcal{I}}^{\mathsf{T}} (\mathbf{l}_m - {}^{\mathcal{G}}_{\mathcal{G}}\mathbf{p}_{\mathcal{I}}) - {}^{\mathcal{I}}_{\mathcal{I}}\mathbf{p}_{\mathcal{R}} \big), \tag{7}$$

$$d_{\mathbf{l}_m} = \left\| \mathbf{l}'_m \right\| \tag{8}$$

As for the velocity factor, for both 3D points and persistent landmark factors, we use DCS for outlier rejection. Compared to the chi-squared-based outlier rejection in [7], using the DCS does not remove the factors judged as out-of-distribution based on their residuals, only down-weigh them.

All 3D points delivered by radar and used for constructing factors are pruned for outliers using RANSAC similarly to [12].

D. Partial Marginalization

As the robot evolves in its environment, new IMU states and persistent landmarks are being added to the state vector. Nonetheless, to keep the state estimation task computationally feasible, we must bound the number of variables in the underlying optimization problem. Hence, upon the addition of new states we must remove the oldest ones by marginalizing them out. In our system we achieve marginalization with the Schur Complement technique (see Fig. 4). Namely, when forming a new graph upon obtaining a new radar measurement, due to conditional independence [26], we consider a sub-graph containing only the states to be marginalized out and the states connected to these states (sometimes called *Markov blanket*). We linearize the resulting sub-graph around the previous solution (current estimate) to obtain its hessian matrix and gradient vector:

$$\begin{bmatrix} \mathbf{H}_{\mu\mu} & \mathbf{H}_{\mu\lambda} \\ \mathbf{H}_{\lambda\mu} & \mathbf{H}_{\lambda\lambda} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{x}}_{\mu} \\ \tilde{\mathbf{x}}_{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_{\mu} \\ \mathbf{b}_{\lambda} \end{bmatrix} \tag{9}$$

Where μ and λ denote the sets of states to marginalize out and states connected to those states, respectively. We calculate the Schur Complement of the states to marginalize out in the hessian and the corresponding gradient:

$$\begin{aligned} \mathbf{H}^*_{\lambda\lambda} &= \mathbf{H}_{\lambda\lambda} - \mathbf{H}_{\lambda\mu}\mathbf{H}_{\mu\mu}^{-1}\mathbf{H}_{\mu\lambda} \\ \mathbf{b}^*_{\lambda} &= \mathbf{b}_{\lambda} - \mathbf{H}_{\lambda\mu}\mathbf{H}_{\mu\mu}^{-1}\mathbf{b}_{\mu} \end{aligned} \tag{10}$$

and use the resulting matrix and vector to form the new prior factor.

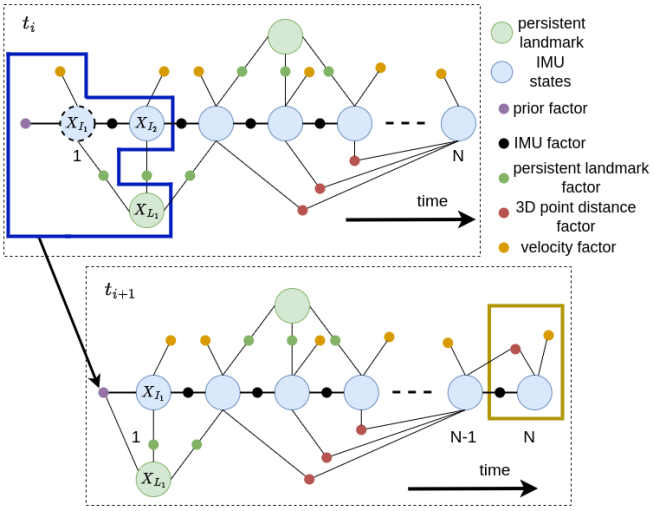

Fig. 4. Depiction of the marginalization of the oldest IMU state (circle with dashed black line). Sub-graph (marked with the blue rectangle) is formed from the state to marginalize out and the states it is connected to (*Markov blanket*). We linearize the sub-graph and in the obtained Gauss-Newton system of equations we apply the Schur Complement to the marginalized out variables to form the new prior factor. Newly added states and factors are marked with the yellow rectangle. Note that the indices adjust in the window from time t_i to t_{i+1} (e.g., $\mathbf{x}_{I_k}(t_i)$ becomes $\mathbf{x}_{I_{k-1}}(t_{i+1})$) as we only keep a maximum of N IMU states.

IV. Results

A. Experimental Setups

We perform real-world experiments with two different platforms (Fig. 1) to demonstrate our RIO method on different systems. One of the platforms (CNS-UAV) is described thoroughly in [7]. The other one is the ARDEA-X UAV designed and built from ground up at the German Aerospace Center's (DLR) Institute of Robotics and Mechatronics. The system was designed for autonomous exploration of unknown regions in the context of planetary space robotics. An Intel NUC for higher-level and a Pixhawk for lower-level tasks make up ARDEA's two primary navigational components. The radar module is connected to the Intel NUC. The IMU data is obtained from the Pixhawk. On both platforms the same SoC FMCW radar chip is mounted (Texas Instruments AWR1843) and configured in the same way (as in [7]). We gather datasets with each platform in which we record the ground truth trajectories using motion capture system and sensor readings from the IMU and radar. In the case of ARDEA-X we record the EKF RIO estimates computed onboard. For CNS-UAV both factor graph and EKF are executed offline on the recorded sensor data on an Intel Core i7-10850H vPRO laptop with 16 GB RAM in a custom C++ framework compiled with gcc 9.4.0 at -O3 optimization level. In the case of CNS-UAV, for both factor graph and EKF RIO we manually calibrate the extrinsic parameters between the IMU and the radar. In the case of the ARDEA-X, for EKF RIO these parameters are estimated online. Both estimators use exactly same front-end parameters and in both cases the sliding window size is set to $N = 10$. Execution time for our factor graph RIO on the above mentioned desktop PC is $16.15\,\mathrm{ms}$ on average which proves its real-time capability. Compared with $2.15\,\mathrm{ms}$ (propagation and update) for the EKF RIO in [7] the latter is (as expected) performing better.

B. Evaluation

We use the data recorded with the two platforms described in Subsec. IV-A for evaluation of our factor graph RIO approach and for comparison with the EKF RIO method from [7]. With each of the platforms we create a dataset of several flown trajectories (five in case of ARDEA-X and three in case of CNS-UAV). Within the CNS-UAV dataset the trajectories are not pre-planned, manually flown, are between 150 m - 175 m long and include pronounced motions in all three dimensions. ARDEA-X dataset contains two pre-planned waypoint-based and three not pre-planned manually flown shorter trajectories. Sample trajectories from each dataset can be seen in the Fig. 5. For every flight in each dataset we compute the norm of the RMSE of position together with the mean and standard deviation of the RMSE values (see Tab. I). Our comparison shows that both methods perform similarly on average as seen in the Tab. I. This is perhaps a counter-intuitive conclusion since the optimization of factor graphs is often considered to provide superior accuracy thanks to successive linearizations performed during the optimization. Nevertheless, in the case where the linearization point is well-determined, this advantage turns out not to be the crucial factor in determining the accuracy. Indeed, in the case of our experiments, the system is either initialized with the knowledge of the ground truth, or with its initial pose being the frame of reference for the estimator. Such settings leave very little room for any transient behaviours of the estimator. Thus, diminishing the value added from possible multiple linearizations.

TABLE I

Norm of RMSE values of position across flights performed with ARDEA-X and CNS-UAV for both methods

ARDEA-X dataset		
Nr	RMSE Norm EKF	RMSE Norm FG
1	0.136	0.213
2	0.083	0.153
3	0.377	0.446
4	0.698	0.711
5	0.265	0.279
Average	**0.312**	**0.360**
Std. dev.	**0.218**	**0.200**
CNS-UAV dataset		
1	1.417	0.625
2	0.877	1.660
3	1.077	1.008
Average	**1.124**	**1.098**
Std. dev.	**0.223**	**0.426**

To demonstrate the benefits of iterative linearizations in the factor graphs during transient phases, we initialize both the EKF and the factor graph-based estimators with wrong initial velocity set to $[2.0, 2.0, 0.0]\frac{m}{s}$ (whereas the true one is equal to $[0.0, 0.0, 0.0]\frac{m}{s}$). In the Fig. 9 we show that in both cases when we do, and do not account for the wrong velocity initialization in the initial covariance, the factor graph-based approach always outperforms the EKF.

In our comparison we note the importance of the front-end in any state estimation system. While both RIO methods perform similarly on average, it turns out that there are qualitative differences between the estimation results of particular UAV trajectories (See Fig. 6, 7, 8). These discrepancies are attributed to different outlier rejection strategies between the EKF and the factor graph-based RIO.

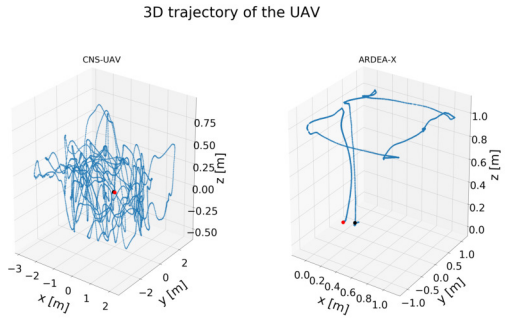

Fig. 5. Sample trajectories from each dataset with the take-off and landing points marked in red and black respectively. Trajectories collected with the CNS-UAV are not pre-planned, longer and in general more challenging in terms of dynamics.

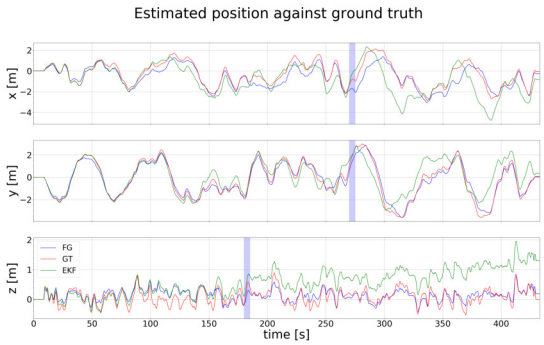

Fig. 6. Estimated position of the UAV using both methods for the first trajectory from the CNS-UAV dataset. The factor graph-based method performs visibly better with the norm of RMSE less than the half of the corresponding RMSE value for the EKF-based method. The thin shaded regions mark points in time where the EKF-based estimator started acquiring very heavy drift due to outliers.

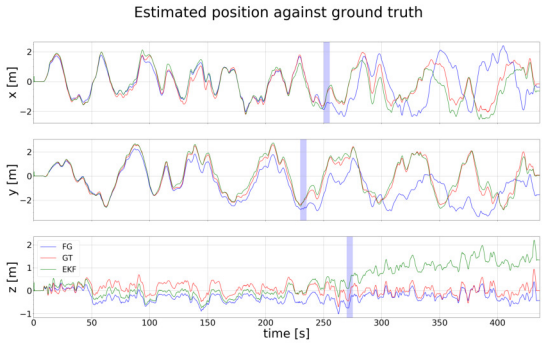

Fig. 7. Estimation results for the position of the UAV using both methods for the second trajectory from the CNS-UAV dataset. Thin shaded regions are marked as areas where the estimators failed to reject outliers. This resulted in major degradation of the accuracy. We see that in the first two sub-plots factor-graph based method acquires very strong yaw drift (x and y axes seem swapped) from which it does not recover. For the EKF the concentration of the outliers in the marked region in the third sub-plot results in a strong vertical drift.

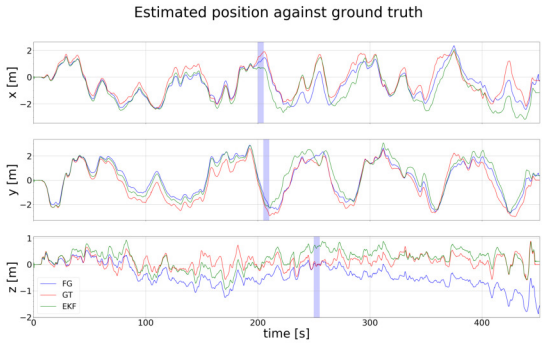

Fig. 8. Plot of the estimates of position of the UAV using both methods for the third trajectory from the CNS-UAV dataset. In this case both estimators perform on-par in tems of RMSE with the outliers affecting the vertical drift for the factor graph-based estimator and x, y coordinates for the EKF-based one.

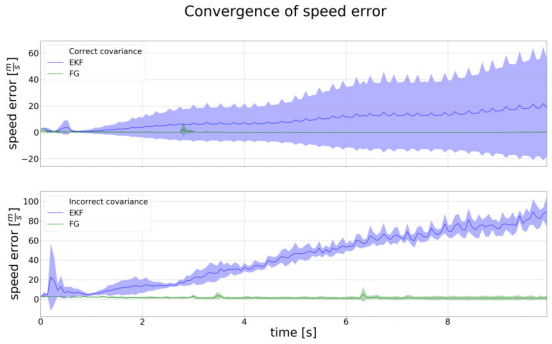

Fig. 9. Plot of convergence of the mean of errors in speed (norm of the velocity) in the case of incorrect velocity initialization for both methods. The mean is taken across the whole ARDEA-X dataset during the first 10 s of the data recording when the UAV stands still on the ground before the take-off. In the upper sub-plot, we initialize incorrectly the velocity, yet we account for this uncertainty by setting accordingly large corresponding covariance entries. The factor graph approach is unaffected and converges quickly to the correct value ($0\,\frac{m}{s}$), whereas the EKF never reaches convergence. In the lower sub-plot, for the same wrong initialization, we do not adjust the covariance settings, effectively misleading the estimator into believing it is initialized correctly. In this case the EKF diverges catastrophically, while factor graph approach still converges although to a slightly offset value. Shaded regions denote the 1σ bounds of the plotted means.

V. Conclusions

In this paper we presented a novel tightly-coupled RIO method based on the nonlinear optimization of factor graphs in a sliding window of states and measurements. The used sensor suite is light-weight, inexpensive, low-power and consumer-grade, hence widely accessible. The presented approach is rendered real-time capable thanks to the application of the partial marginalization of oldest states in order to bound the state vector size and form an informative prior for the estimator. We performed a *one-to-one* comparison with a state-of-the-art multi-state EKF RIO method on the in-house datasets collected with two different UAV platforms in order to demonstrate the soundness of our framework. Comparing the two methods reveals that they perform on-par in terms of accuracy when the linearization point is not far from the true state. In terms of CPU load the comparison shows that the EKF RIO is less resource-demanding. We demonstrated the advantages that successive linearizations in the factor graph-based method bring to the convergence of the estimator in transient phases (when the linearization point is far from the true state). We provided this demonstration in both the case when the wrong initialization is, and when it is not reflected in the uncertainty of the initial state (which is often the case in practice). The whole comparison was performed in such way that both EKF and factor graph backends were implemented in the same software framework, effectively sharing exactly the same front-end features and parameters. We make the software framework and datasets used in the present paper open-source in order to facilitate further research and comparisons.

REFERENCES

[1] A. Kramer, C. Stahoviak, A. Santamaria-Navarro, A.-A. Agha-Mohammadi, and C. Heckman, "Radar-inertial ego-velocity estimation for visually degraded environments," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 5739–5746.

[2] Y. S. Park, Y.-S. Shin, J. Kim, and A. Kim, "3d ego-motion estimation using low-cost mmwave radars via radar velocity factor for pose-graph slam," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7691–7698, 2021.

[3] Z. Hong, Y. Petillot, and S. Wang, "Radarslam: Radar based large-scale slam in all weathers," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 5164–5170.

[4] K. Burnett, D. J. Yoon, A. P. Schoellig, and T. D. Barfoot, "Radar odometry combining probabilistic estimation and unsupervised feature learning," *arXiv preprint arXiv:2105.14152*, 2021.

[5] D. Barnes and I. Posner, "Under the radar: Learning to predict robust keypoints for odometry estimation and metric localisation in radar," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 9484–9490.

[6] Y. S. Park, Y.-S. Shin, and A. Kim, "Pharao: Direct radar odometry using phase correlation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 2617–2623.

[7] J. Michalczyk, R. Jung, C. Brommer, and S. Weiss, "Multi-state tightly-coupled ekf-based radar-inertial odometry with persistent landmarks," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 4011–4017.

[8] J. Michalczyk, R. Jung, and S. Weiss, "Tightly-coupled ekf-based radar-inertial odometry," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 12 336–12 343.

[9] P. Gao, S. Zhang, W. Wang, and C. X. Lu, "Dc-loc: Accurate automotive radar based metric localization with explicit doppler compensation," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 4128–4134.

[10] Y. Z. Ng, B. Choi, R. Tan, and L. Heng, "Continuous-time radar-inertial odometry for automotive radars," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 323–330.

[11] X. Li, H. Zhang, and W. Chen, "4d radar-based pose graph slam with ego-velocity pre-integration factor," *IEEE Robotics and Automation Letters*, vol. 8, no. 8, pp. 5124–5131, 2023.

[12] C. Doer and G. F. Trommer, "An ekf based approach to radar inertial odometry," in *2020 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*. IEEE, 2020, pp. 152–159.

[13] Y. Almalioglu, M. Turan, C. X. Lu, N. Trigoni, and A. Markham, "Milli-rio: Ego-motion estimation with low-cost millimetre-wave radar," *IEEE Sensors Journal*, vol. 21, no. 3, pp. 3314–3323, 2020.

[14] J. Hasch, E. Topak, R. Schnabel, T. Zwick, R. Weigel, and C. Waldschmidt, "Millimeter-wave technology for automotive radar sensors in the 77 ghz frequency band," *IEEE Transactions on Microwave Theory and Techniques*, vol. 60, no. 3, pp. 845–860, 2012.

[15] H. Rohling and M.-M. Meinecke, "Waveform design principles for automotive radar systems," in *2001 CIE International Conference on Radar Proceedings (Cat No. 01TH8559)*. IEEE, 2001, pp. 1–4.

[16] M. Schneider, "Automotive radar-status and trends," in *German microwave conference*, 2005, pp. 144–147.

[17] F. Dellaert, M. Kaess *et al.*, "Factor graphs for robot perception," *Foundations and Trends® in Robotics*, vol. 6, no. 1-2, pp. 1–139, 2017.

[18] P. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[19] C. Doer and G. F. Trommer, "Radar inertial odometry with online calibration," in *2020 European Navigation Conference (ENC)*. IEEE, 2020, pp. 1–10.

[20] ——, "Yaw aided radar inertial odometry using manhattan world assumptions," in *2021 28th Saint Petersburg International Conference on Integrated Navigation Systems (ICINS)*, 2021, pp. 1–9.

[21] Y. Zhuang, B. Wang, J. Huai, and M. Li, "4d iriom: 4d imaging radar inertial odometry and mapping," *IEEE Robotics and Automation Letters*, vol. 8, no. 6, pp. 3246–3253, 2023.

[22] "Gtsam." [Online]. Available: https://github.com/borglab/gtsam

[23] H. Sommer, I. Gilitschenski, M. Bloesch, S. Weiss, R. Siegwart, and J. Nieto, "Why and how to avoid the flipped quaternion multiplication," *Aerospace*, vol. 5, no. 3, p. 72, 2018.

[24] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration for real-time visual–inertial odometry," *IEEE Transactions on Robotics*, vol. 33, no. 1, pp. 1–21, 2016.

[25] P. Agarwal, G. D. Tipaldi, L. Spinello, C. Stachniss, and W. Burgard, "Robust map optimization using dynamic covariance scaling," in *2013 IEEE International Conference on Robotics and Automation*, 2013, pp. 62–69.

[26] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual–inertial odometry using nonlinear optimization," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2015.