Embedding Manifold Structures into Kalman Filters

Dongjiao He[1], Wei Xu[1], Fu Zhang[1]

***Abstract*—Error-state Kalman filter is an elegant and effective filtering technique for robotic systems operating on manifolds. However, to implement an error-state Kalman filter for a certain robotic system, one usually needs to derive each step of the filter by switching between the original and the error state, a tedious and prone-to-error process. To avoid such repetitive derivations, this paper proposes a generic symbolic representation for error-state Kalman filters on manifolds. Utilizing the $\boxplus\backslash\boxminus$ operations and further defining a $\oplus$ operation on the respective manifold, we propose a canonical representation of the robotic system. Such a canonical form enables us to separate the manifold structures from the system descriptions in each step of the Kalman filter, ultimately leading to a generic, symbolic, and manifold-embedding Kalman filter framework. A major advantage of the proposed manifold-embedding Kalman filter framework is that users need only to cast the system model into the canonical form without going through the cumbersome hand-derivation of the on-manifold Kalman filter. This is particularly useful when the robotic system is of high dimension (e.g., augmented internal state, multiple extrinsic parameters, swarms). Furthermore, the manifold-embedding Kalman filter is implemented as a toolkit in C++ packages with which an user needs only to define the system, and call the respective filter steps (e.g., propagation, update) according to the events (e.g., reception of input, reception of measurement). The existing implementation supports full iterated Kalman filtering for systems on manifold $\mathcal{S} = \mathbb{R}^m \times SO(3) \times \cdots \times SO(3) \times \mathbb{S}^2 \times \cdots \times \mathbb{S}^2$ or any of its sub-manifolds, and is extendable to other types of manifold when necessary. The proposed symbolic Kalman filter and the developed toolkit are verified by implementing a tightly-coupled lidar-inertial navigation system. Results show that the developed toolkit leads to superior filtering performances and computation efficiency comparable to hand-engineered counterparts. Finally, the toolkit is opened sourced at https://github.com/hku-mars/IKFoM to assist practitioners to quickly deploy an on-manifold Kalman filter.**

I. Introduction

A popular robotic filtering technique is the error-state extended Kalman filter (ESEKF), such as in attitude estimation [1]–[3], online extrinsic calibration [4, 5], GPS/IMU navigation [6], visual inertial navigation [7]–[12] and lidar-inertial navigation [13]–[15]. The basic idea of ESEKF is to repeatedly parameterize the state trajectory $\mathbf{x}_\tau \in \mathcal{S}$ by an error state trajectory $\delta\mathbf{x}_{\tau|k} \in \mathbb{R}^n$ from the current state predict $\mathbf{x}_{\tau|k}$: $\mathbf{x}_\tau = \mathbf{x}_{\tau|k} \boxplus \delta\mathbf{x}_{\tau|k}$. Then a normal extended Kalman filter is performed on the error state trajectory $\delta\mathbf{x}_{\tau|k}$ to update the error state, and adds the updated error state back to the original state on manifolds. Since this error is small, minimal parameterization (e.g., rotation axis and Euler angle) can be employed without concerning the singularity issue (see Fig. 1). In addition, compared to other techniques such as unscented Kalman filter (UKF), the efficiency of the extended Kalman filter is higher. With the superiority of accuracy, stability and efficiency, the ESEKF provides an elegant Kalman filter framework for nonlinear robotic systems.

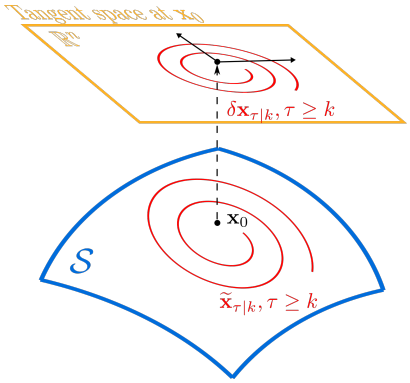

Fig. 1. Illustration of the error state trajectory when $\mathcal{S}$ is a group with operation $\cdot$. The $\mathbb{R}^n$ space is tangent of $\mathcal{S}$ space at the identity $\mathbf{x}_0$. $\delta\mathbf{x}_{\tau|k}$ is a minimal parameterization of the error state $\widetilde{\mathbf{x}}_{\tau|k} = \mathbf{x}_{\tau|k}^{-1} \cdot \mathbf{x}_\tau \in \mathcal{S}$.

Despite these advantages, deploying an ESEKF for a certain robotic system is usually more difficult than normal EKFs. Due to the lack of canonical representation of systems on manifolds, existing ESEKFs are designed case by case, and usually require a user to fully understand its underlying principle (e.g., switching between the original state and the error state) and to manually derive each step (e.g., propagation, update, reset) from scratch for a customized system. Although this may seem like a mere book-keeping issue but in practice it tends to be particularly cumbersome and error-prone, especially for systems of high dimension, such as robotic swarms and systems with augmented internal states [16] or multiple extrinsic parameters [17]. Besides the system dimension, the difficulty in hand-derivation also rapidly escalates when the error-state is coupled with iteration (e.g., iterated error-state Kalman filter), which has recently found more applications in visual-inertial [11] and lidar-inertial navigation [14, 15] to mitigate the linearization error in extended Kalman filters [18, 19].

In this paper, we address the above issues by embedding the manifold structures into the Kalman filter framework. Specifically, our contributions are as follows: 1) We propose a canonical and generic representation of robotic systems in discrete time, i.e., $\mathbf{x}_{k+1} = \mathbf{x}_k \oplus (\Delta t \mathbf{f}(\mathbf{x}_k, \mathbf{w}_k))$; 2) Based on the canonical system representation, we show that the manifold-specific structures are well separated from the the system-specific descriptions in each step of a Kalman filter, enabling us to embed the manifold structures into the Kalman filter. We further derive a fully iterated, symbolic, and error-state Kalman filter termed as *IKFoM* on the canonical system representation; 3) We embed the manifold structures into the

[1]All authors are with Department of Mechanical Engineering, University of Hong Kong. { hdj65822, xuweii, fuzhang}@hku.hk

derived iterated Kalman filter and develop an open source C++ package. Its main advantage is hiding all the Kalman filter derivations and manifold-specific operations, and leaving the user to supply system-specific descriptions only and call the respective filter steps (e.g., propagation, update) in the running time; 4) We verify our formulation and implementations with a tightly-coupled lidar-inetial navigation system and on various real-world datasets.

II. Related Work

Kalman filter and its variants are very effective techniques for robots state estimation. However, Kalman filter operates in state space of Euclidean space $\mathbb{R}^n$ while robotic systems usually have their states on manifolds (e.g., rotation group $SO(3)$). To overcome this discrepancy, one could use a parameterization of the state with minimal dimensions [20]. This minimal parameterization unfortunately has singularities. For example, Euler angle representation of $SO(3)$ has singularities at $\pm 90^\circ$ rotations along the second rotation axis, and the axis-angle representation has singularities at 180° of rotations [21]. Workarounds for this singularity exist and they either avoid these parts of the state space, as done in the Apollo Lunar Module [22], or switch between alternative orderings of the parameterization each of which exhibits singularities in different areas of the state space.

Another approach to overcome the singularity is representing the system states using redundant parameters (i.e., over-parameterization). For example, unit quaternion is often used to represent rotations on $SO(3)$. Yet, the over-parameterization shifts the problem from system representation to the filtering algorithm: viewing the over-parameterized state as a normal vector in Euclidean space and applying the Kalman filter (or its variants) will make the propagated state no longer lie on the manifold (i.e., unit quaternion $\mathbf{q}^T\mathbf{q} = 1$ is violated). One ad-hoc way to ensure the propagated state stay on the manifold is normalization. Since the normalization imposes constraints on the state, the propagated covariance should be adjusted in parallel. For example, a unit quaternion $\mathbf{q}^T\mathbf{q} = 1$ leads to an error satisfying $\mathbf{q}^T\delta\mathbf{q} = 0$, which means that the error is zero along the direction $\mathbf{q}$ and the corresponding covariance should be adjusted to zero [23] too. The adjusted covariance propagation is therefore singular. Although the Kalman filter still works with this singular covariance as long as the innovation covariance remains positive definite, it is unknown if this phenomenon causes further problems, e.g., the zero-uncertainty direction could create overconfidence in other directions after a nonlinear update [24]. An alternative way to interpret the normalization is viewing 1 as the measurement of $\mathbf{q}^T\mathbf{q}$, thus one more nonlinear measurement $\mathbf{h}(\mathbf{q}) = \mathbf{q}^T\mathbf{q}$ should be added to the system. The augmented measurements will then update the covariance in the Kalman filter framework. This approach is somewhat equivalent to the first (viewing 1 as the measurement of $\mathbf{q}^T\mathbf{q}$ is equivalent to viewing 0 as the measurement of $\mathbf{q}^T\delta\mathbf{q}$ to the first order) and hence suffers from the same problem.

A more elegant approach is transforming the original system that operates on a manifold to its equivalent error space (i.e., tangent space) which is defined as the difference between the groundtruth state and its most recent prediction. Since this error is small when the Kalman filter converges, it can be safely parameterized by a minimal set of parameters (e.g., axis-angle) without occurring singularity. Then a normal EKF is used to update the minimally-parameterized error state, which is finally added back to the original state on the manifold. Such an indirect way to update the state estimate has different names, such as "error state" EKF (ESEKF) [6], indirect EKF [2], or multiplicative EKF [1]. ESEKF provides an elegant way to incorporate filtering techniques into systems on manifolds, and has been widely used in a variety of robotic applications [1]–[10, 12, 13]. To better describe the relation between the original state on manifold and the error state, the $\boxplus\backslash\boxminus$ operations are introduced in [25] and widely adopted by unscented Kalman filters [24, 26] and more recently iterated Kalman filters [11, 14, 15]. The $\boxplus\backslash\boxminus$ operations have also been widely used in manifold-based optimizations [27, 28] such as calibration [29], graph-SLAM [30] and parameter identification [31].

This paper focuses on deriving a generic and symbolic Kalman filter framework for robotic systems naturally operating on manifolds. We propose a canonical representation of robotic systems, based on which a fully iterated and symbolic Kalman fitler framework is derived. For well-studied Special Orthogonal group $SO(3)$, our work eventually leads to nearly the same Kalman filter as in [1]–[10, 12, 13] for a specific system (up to the discretization accuracy), but unifies all of them into one canonical form. Moreover, our work provides a general way to incorporate new manifolds structures that are less studied, such as the 2-sphere $\mathbb{S}^2$ for modeling the bearing vector of a visual landmark [11].

The rest of the paper is organized as follows: Section III introduces the $\boxplus\backslash\boxminus$ and $\oplus$ operations. Section IV presents the canonical representation of robotic systems, based on which Section V derives a fully iterated and symbolic Kalman filter. Section VI implements the symbolic error-state iterated Kalman filter as a C++ package. Experiment results are presented in Section VII. Finally, Section VIII concludes this paper.

III. Operations on manifolds

A. $\boxplus\backslash\boxminus$ and $\oplus$ operations

Let $\mathcal{S}$ be a n-manifold the robot is operating on, then $\mathcal{S}$ is locally homeomorphic to $\mathbb{R}^n$, the tangent space. Let the local map at $\mathbf{x} \in \mathcal{S}$ be denoted as ${}^{\mathcal{S}}\boldsymbol{\varphi}_{\mathbf{x}} : \mathcal{S} \mapsto \mathbb{R}^n$ with inverse map ${}^{\mathcal{S}}\boldsymbol{\varphi}_{\mathbf{x}}^{-1}$. Further assume that the map is centered at $\mathbf{x}$ (i.e., ${}^{\mathcal{S}}\boldsymbol{\varphi}_{\mathbf{x}}(\mathbf{x}) = \mathbf{0}$). Referring to [25], we establish a bijective map from a local neighborhood in $\mathcal{S}$ to $\mathbb{R}^n$ via two operators $\boxplus_{\mathcal{S}}$ ("boxplus") and $\boxminus_{\mathcal{S}}$ ("boxminus"):

$$\begin{aligned} \boxplus &: \mathcal{S} \times \mathbb{R}^n \to \mathcal{S} \\ &\quad \mathbf{x} \boxplus_{\mathcal{S}} \mathbf{u} = {}^{\mathcal{S}}\boldsymbol{\varphi}_{\mathbf{x}}^{-1}(\mathbf{u}) \\ \boxminus &: \mathcal{S} \times \mathcal{S} \to \mathbb{R}^n \\ &\quad \mathbf{y} \boxminus_{\mathcal{S}} \mathbf{x} = {}^{\mathcal{S}}\boldsymbol{\varphi}_{\mathbf{x}}(\mathbf{y}) \end{aligned} \tag{1}$$

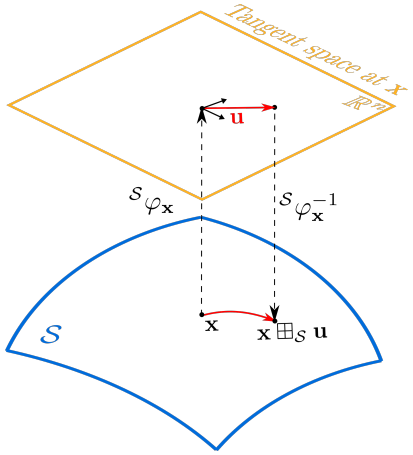

Fig. 2. Illustration of the $\boxplus_{\mathcal{S}}$ operation on manifold $\mathcal{S}$.

It can be shown that $\mathbf{x}\boxplus_{\mathcal{S}}(\mathbf{y}\boxminus_{\mathcal{S}}\mathbf{x})=\mathbf{y}$ and $(\mathbf{x}\boxplus_{\mathcal{S}}\mathbf{u})\boxminus_{\mathcal{S}}\mathbf{x}=\mathbf{u},\ \forall\mathbf{x},\ \mathbf{y}\in\mathcal{S},\mathbf{u}\in\mathbb{R}^n$. The physical interpretation of $\mathbf{y}=\mathbf{x}\boxplus_{\mathcal{S}}\mathbf{u}$ is adding a small perturbation $\mathbf{u}\in\mathbb{R}^n$ to $\mathbf{x}\in\mathcal{S}$, as illustrated in Fig. 2. And the inverse operation $\mathbf{u}=\mathbf{y}\boxminus_{\mathcal{S}}\mathbf{x}$ determines the perturbation $\mathbf{u}$ which yields $\mathbf{y}\in\mathcal{S}$ when $\boxplus_{\mathcal{S}}$-added to $\mathbf{x}$. These two operators create a local, vectorized view of the globally more complex structure of the manifold.

In particular, when $\mathcal{S}$ is a Lie group (e.g., $\mathbb{R}^n$, $SO(3)$, $SE(3)$), the local map $^{\mathcal{S}}\boldsymbol{\varphi}_{\mathbf{x}}(\cdot)$ reduces to:

$$\begin{aligned}\mathbf{x}\boxplus_{\mathcal{S}}\mathbf{u}&=\mathbf{x}\cdot\mathrm{Exp}(\mathbf{u})\\ \mathbf{y}\boxminus_{\mathcal{S}}\mathbf{x}&=\mathrm{Log}(\mathbf{x}^{-1}\cdot\mathbf{y})\end{aligned}\tag{2}$$

where $\cdot$ is the binary operation on $\mathcal{S}$ such that $(\mathcal{S},\cdot)$ forms a Lie group, $\mathrm{Exp}(\cdot)$ is the exponential function [32], and $\mathbf{x}^{-1}$ is the inverse of $\mathbf{x}$ that always exist for an element on Lie groups by definition.

In addition to $\boxplus/\boxminus$, we define a binary operation $\oplus_{\mathcal{S}}:\mathcal{S}\times\mathbb{R}^l\mapsto\mathcal{S}$ that drives the state in $\mathcal{S}$ according to an input in $\mathbb{R}^l$. In particular, when $\mathcal{S}$ is a Lie group (e.g., $\mathbb{R}^n$, $SO(3)$, $SE(3)$) which is naturally driven by its Lie algebra by the exponential map, the binary operation $\oplus$ reduces to $\boxplus$.

$$\mathbf{x}\oplus_{\mathcal{S}}\mathbf{v}=\mathbf{x}\boxplus_{\mathcal{S}}\mathbf{u}=\mathbf{x}\cdot\mathrm{Exp}(\mathbf{v})\quad(\text{i.e.},l=n)\tag{3}$$

For the sake of notation simplicity, in the following discussion, we drop the subscript $\mathcal{S}$ in operations $\boxplus,\boxminus$ and $\oplus$ when no ambiguity exists.

B. Differentiations

In the Kalman filter that will be derived later in Section. V, the partial differentiation of $(((\mathbf{x}\boxplus\mathbf{u})\oplus\mathbf{v})\boxminus\mathbf{y})$ with respect to $\mathbf{u}$ and $\mathbf{v}$ will be used, where $\mathbf{x},\mathbf{y}\in\mathcal{S},\mathbf{u}\in\mathbb{R}^n$ and $\mathbf{v}\in\mathbb{R}^l$. This can be obtained easily from the chain rule as follows:

$$\begin{aligned}\frac{\partial(((\mathbf{x}\boxplus\mathbf{u})\oplus\mathbf{v})\boxminus\mathbf{y})}{\partial\mathbf{u}}&=\frac{\partial^{\mathcal{S}}\boldsymbol{\varphi}_{\mathbf{y}}(\mathbf{z})}{\partial\mathbf{z}}\Big|_{\mathbf{z}=(\mathbf{x}\boxplus\mathbf{u})\oplus\mathbf{v}}\\&\quad\cdot\frac{\partial(\mathbf{z}\oplus\mathbf{v})}{\partial\mathbf{z}}\Big|_{\mathbf{z}=\mathbf{x}\boxplus\mathbf{u}}\cdot\frac{\partial^{\mathcal{S}}\boldsymbol{\varphi}_{\mathbf{x}}^{-1}(\mathbf{z})}{\partial\mathbf{z}}\Big|_{\mathbf{z}=\mathbf{u}}\\ \frac{\partial(((\mathbf{x}\boxplus\mathbf{u})\oplus\mathbf{v})\boxminus\mathbf{y})}{\partial\mathbf{v}}&=\frac{\partial^{\mathcal{S}}\boldsymbol{\varphi}_{\mathbf{y}}(\mathbf{z})}{\partial\mathbf{z}}\Big|_{\mathbf{z}=(\mathbf{x}\boxplus\mathbf{u})\oplus\mathbf{v}}\cdot\frac{\partial((\mathbf{x}\boxplus\mathbf{u})\oplus\mathbf{z})}{\partial\mathbf{z}}\Big|_{\mathbf{z}=\mathbf{v}}\end{aligned}\tag{4}$$

For certain manifolds (e.g., $SO(3)$), it is usually more convenient to compute the differentiations $\frac{\partial(((\mathbf{x}\boxplus\mathbf{u})\oplus\mathbf{v})\boxminus\mathbf{y})}{\partial\mathbf{u}}$ and $\frac{\partial(((\mathbf{x}\boxplus\mathbf{u})\oplus\mathbf{v})\boxminus\mathbf{y})}{\partial\mathbf{v}}$ directly instead of using the above chain rule.

C. Important manifolds in practice

Example 1: Euclidean space $\mathcal{S}=\mathbb{R}^n$:

$$\begin{aligned}\mathbf{x}\boxplus\mathbf{u}&=\mathbf{x}+\mathbf{u}\\ \mathbf{y}\boxminus\mathbf{x}&=\mathbf{y}-\mathbf{x}\\ \mathbf{x}\oplus\mathbf{v}&=\mathbf{x}+\mathbf{v}\\ \frac{\partial(((\mathbf{x}\boxplus\mathbf{u})\oplus\mathbf{v})\boxminus\mathbf{y})}{\partial\mathbf{u}}&=\mathbf{I}_{n\times n}\\ \frac{\partial(((\mathbf{x}\boxplus\mathbf{u})\oplus\mathbf{v})\boxminus\mathbf{y})}{\partial\mathbf{v}}&=\mathbf{I}_{n\times n}\end{aligned}\tag{5}$$

Example 2: Special orthogonal group $\mathcal{S}=SO(3)$:

$$\begin{aligned}\mathbf{x}\boxplus\mathbf{u}&=\mathbf{x}\cdot\mathrm{Exp}(\mathbf{u})\\ \mathbf{y}\boxminus\mathbf{x}&=\mathrm{Log}(\mathbf{x}^{-1}\cdot\mathbf{y})\\ \mathbf{x}\oplus\mathbf{v}&=\mathbf{x}\cdot\mathrm{Exp}(\mathbf{v})\\ \frac{\partial(((\mathbf{x}\boxplus\mathbf{u})\oplus\mathbf{v})\boxminus\mathbf{y})}{\partial\mathbf{u}}&=\mathbf{A}(((\mathbf{x}\boxplus\mathbf{u})\oplus\mathbf{v})\boxminus\mathbf{y})^{-T}\mathrm{Exp}(-\mathbf{v})\mathbf{A}(\mathbf{u})^T\\ \frac{\partial(((\mathbf{x}\boxplus\mathbf{u})\oplus\mathbf{v})\boxminus\mathbf{y})}{\partial\mathbf{v}}&=\mathbf{A}(((\mathbf{x}\boxplus\mathbf{u})\oplus\mathbf{v})\boxminus\mathbf{y})^{-T}\mathbf{A}(\mathbf{v})^T\end{aligned}\tag{6}$$

where

$$\begin{aligned}\mathrm{Exp}(\mathbf{u})&=\exp(\lfloor\mathbf{u}\rfloor)\\ \mathbf{A}(\mathbf{u})&=\mathbf{I}+\left(\tfrac{1-\cos(\|\mathbf{u}\|)}{\|\mathbf{u}\|}\right)\tfrac{\lfloor\mathbf{u}\rfloor}{\|\mathbf{u}\|}+\left(1-\tfrac{\sin(\|\mathbf{u}\|)}{\|\mathbf{u}\|}\right)\tfrac{\lfloor\mathbf{u}\rfloor^2}{\|\mathbf{u}\|^2}\\ \mathbf{A}(\mathbf{u})^{-1}&=\mathbf{I}-\tfrac{1}{2}\lfloor\mathbf{u}\rfloor+(1-\alpha(\|\mathbf{u}\|))\tfrac{\lfloor\mathbf{u}\rfloor^2}{\|\mathbf{u}\|^2}\\ \alpha(\|\mathbf{u}\|)&=\tfrac{\|\mathbf{u}\|}{2}\cot\left(\tfrac{\|\mathbf{u}\|}{2}\right)=\tfrac{\|\mathbf{u}\|}{2}\tfrac{\cos(\|\mathbf{u}\|/2)}{\sin(\|\mathbf{u}\|/2)}\end{aligned}\tag{7}$$

The derivation of the above differentiation is shown in Lemma 1 in Appendix A. And the notation $\lfloor\mathbf{u}\rfloor$ denotes the skew-symmetric matrix that maps the cross product of $\mathbf{u}\in\mathbb{R}^3$.

Example 3: Special Euclidean $\mathcal{S}=SE(3)$:

$$\begin{aligned}\mathbf{x}\boxplus\mathbf{u}&=\mathbf{x}\cdot\mathrm{Exp}(\mathbf{u})\\ \mathbf{y}\boxminus\mathbf{x}&=\mathrm{Log}(\mathbf{x}^{-1}\cdot\mathbf{y})\end{aligned}\tag{8}$$

where $\mathbf{u}=\begin{bmatrix}\boldsymbol{\rho}^T & \boldsymbol{\theta}^T\end{bmatrix}^T\in\mathbb{R}^6$, $\mathrm{Exp}(\mathbf{u})=\begin{bmatrix}\exp(\lfloor\boldsymbol{\theta}\rfloor) & \boldsymbol{\rho}\\ \mathbf{0} & 1\end{bmatrix}$. One difficulty with $SE(3)$ is that its Jacobian has no closed form as shown in [33], hence $SE(3)$ should be avoided by viewing it as a *compound manifold* $\mathcal{S}=SO(3)\times\mathbb{R}^3$.

Example 4: 2-sphere, $\mathcal{S}=\mathbb{S}^2(r)\triangleq\{\mathbf{x}\in\mathbb{R}^3|\|\mathbf{x}\|=r,r>0\}$. The 2-sphere manifold is usually used to describe vectors of fixed length r, such as the gravity vector with known magnitude and the bearing vector of a visual feature [11]. Referring to Fig. 3, one way to define $\mathbf{x}\boxplus\mathbf{u}$ is rotating $\mathbf{x}$ along an vector $\mathbf{u}\in\mathbb{R}^2$ in the tangent plane, the result would still remain on $\mathbb{S}^2(r)$ as required. Assume $\mathbf{b}_1,\mathbf{b}_2$ are two orthonormal basis in the tangent plane and recall the definition of $\mathrm{Exp}(\cdot)$ in (7), we have

$$\mathbf{x}\boxplus\mathbf{u}\triangleq\mathrm{Exp}(\begin{bmatrix}\mathbf{b}_1 & \mathbf{b}_2\end{bmatrix}\mathbf{u})\cdot\mathbf{x}\tag{9}$$

In many practical robotic systems (see Section. IV), the state on $\mathbb{S}^2(r)$ usually represents a direction that may undergo certain angular motion. Hence, a suitable choice for the binary operation $\oplus$ is a rotation of an angle-axis vector $\mathbf{v}\in\mathbb{R}^3$:

$$\begin{aligned}\oplus:\mathbb{S}^2(r)\times\mathbb{R}^3&\to\mathbb{S}^2(r)\\ \mathbf{x}\oplus\mathbf{v}&=\mathrm{Exp}(\mathbf{v})\mathbf{x}\end{aligned}\tag{10}$$

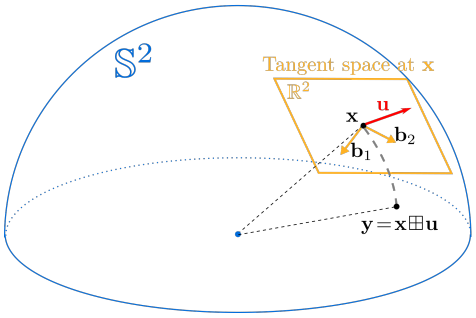

Fig. 3. Illustration of the $\boxplus$ operation on the $\mathcal{S}^2$ space.

Notice that $\mathbf{b}_1$ and $\mathbf{b}_2$ depend on $\mathbf{x}$, and denote $\mathbf{B}(\mathbf{x}) = \begin{bmatrix}\mathbf{b}_1 & \mathbf{b}_2\end{bmatrix} \in \mathbb{R}^{3\times 2}$. Then

$$
\begin{aligned}
&\mathbf{x}\boxplus\mathbf{u} = \mathrm{Exp}\left(\mathbf{B}\left(\mathbf{x}\right)\mathbf{u}\right)\cdot\mathbf{x}\\
&\mathbf{y}\boxminus\mathbf{x} = \mathbf{B}\left(\mathbf{x}\right)^T\left(\theta\frac{\lfloor\mathbf{x}\rfloor\mathbf{y}}{\|\lfloor\mathbf{x}\rfloor\mathbf{y}\|}\right), \theta = \mathrm{atan2}\left(\|\lfloor\mathbf{x}\rfloor\mathbf{y}\|, \mathbf{x}^{\mathrm{T}}\mathbf{y}\right)\\
&\mathbf{x}\oplus\mathbf{v} = \mathrm{Exp}(\mathbf{v})\mathbf{x}\\
&\frac{\partial((\mathbf{x}\boxplus\mathbf{u})\oplus\mathbf{v})\boxminus\mathbf{y})}{\partial\mathbf{u}} = \mathbf{N}((\mathbf{x}\boxplus\mathbf{u})\oplus\mathbf{v},\mathbf{y})\mathrm{Exp}(\mathbf{v})\mathbf{M}\left(\mathbf{x},\mathbf{u}\right)\\
&\frac{\partial((\mathbf{x}\boxplus\mathbf{u})\oplus\mathbf{v})\boxminus\mathbf{y})}{\partial\mathbf{v}} = -\mathbf{N}((\mathbf{x}\boxplus\mathbf{u})\oplus\mathbf{v},\mathbf{y})\mathrm{Exp}(\mathbf{v})\lfloor\mathbf{x}\boxplus\mathbf{u}\rfloor\mathbf{A}(\mathbf{v})^{\mathrm{T}}
\end{aligned}
\tag{11}
$$

where the $\mathbf{N}\left(\mathbf{x},\mathbf{y}\right)$ and $\mathbf{M}\left(\mathbf{x},\mathbf{u}\right)$ are defined as:

$$
\begin{aligned}
&\mathbf{N}(\mathbf{x},\mathbf{y}) = \tfrac{\partial(\mathbf{x}\boxminus\mathbf{y})}{\partial\mathbf{x}} = \mathbf{B}(\mathbf{y})^T\Big(\tfrac{\theta}{\|\lfloor\mathbf{y}\rfloor\mathbf{x}\|}\lfloor\mathbf{y}\rfloor + \lfloor\mathbf{y}\rfloor\mathbf{x}\cdot\mathbf{P}(\mathbf{x},\mathbf{y})\Big)\\
&\mathbf{M}(\mathbf{x},\mathbf{u}) = \tfrac{\partial(\mathbf{x}\boxplus\mathbf{u})}{\partial\mathbf{u}} = -\mathrm{Exp}(\mathbf{B}(\mathbf{x})\mathbf{u})\lfloor\mathbf{x}\rfloor\mathbf{A}(\mathbf{B}(\mathbf{x})\mathbf{u})^{\mathrm{T}}\mathbf{B}(\mathbf{x})\\
&\mathbf{P}(\mathbf{x},\mathbf{y}) = \tfrac{1}{r^4}\left(\tfrac{-\mathbf{y}^T\mathbf{x}\|\lfloor\mathbf{y}\rfloor\mathbf{x}\| + r^4\theta}{\|\lfloor\mathbf{y}\rfloor\mathbf{x}\|^3}\mathbf{x}^T\lfloor\mathbf{y}\rfloor^2 - \mathbf{y}^T\right)
\end{aligned}
\tag{12}
$$

where $\mathbf{A}(\cdot)$ is defined in (7). Note that we have $\mathbf{N}(\mathbf{y},\mathbf{y}) = \frac{1}{r^2}\mathbf{B}(\mathbf{y})^T\lfloor\mathbf{y}\rfloor, \forall\mathbf{y}\in\mathbb{S}^2$.

The above results do not specify the basis $\mathbf{B}(\mathbf{x})$, which can be made arbitrary as long as it forms an orthonormal basis in the tangent plane of $\mathbf{x}$. For example, we could adopt the method in [34] (see Fig. 4): rotate one of the three canonical basis $\mathbf{e}_i, i = 1,2,3$ to $\mathbf{x}$ (along the *geodesics*) and the rest two $\mathbf{e}_i$ after the rotation would be $\mathbf{B}(\mathbf{x})$. To avoid the singularity in the rotation when $\mathbf{x} = -r\mathbf{e}_i$, $\mathbf{e}_i$ is instantaneously chosen such that it has the largest distance to $-\mathbf{x}$, i.e.,

$$
\begin{aligned}
&i = \underset{j}{\mathrm{argmax}}\ \mathbf{x}^T\mathbf{e}_j,\\
&\mathbf{R}_i(\mathbf{x}) = \mathrm{Exp}\left(\tfrac{\lfloor\mathbf{e}_i\rfloor\mathbf{x}}{\|\lfloor\mathbf{e}_i\rfloor\mathbf{x}\|}\mathrm{atan2}\left(\|\lfloor\mathbf{e}_i\rfloor\mathbf{x}\|, \mathbf{e}_i^T\mathbf{x}\right)\right),\\
&\mathbf{B}(\mathbf{x}) = \mathbf{R}_i(\mathbf{x})\begin{bmatrix}\mathbf{e}_j & \mathbf{e}_k\end{bmatrix}.
\end{aligned}
\tag{13}
$$

where $j = i+1, k = i+2$ but wrapped below 3.

D. $\boxplus\backslash\boxminus$ and $\oplus$ operations for compound manifolds

Based on the principles of the Cartesian product of manifolds, the $\boxplus\backslash\boxminus$ and $\oplus$ on a *compound manifold* of two (and by induction arbitrary numbers of) sub-manifolds are defined as:

$$
\underbrace{\begin{bmatrix}\mathbf{x}_1\\ \mathbf{x}_2\end{bmatrix}}_{\mathbf{x}}\boxplus\underbrace{\begin{bmatrix}\mathbf{u}_1\\ \mathbf{u}_2\end{bmatrix}}_{\mathbf{u}} = \begin{bmatrix}\mathbf{x}_1\boxplus\mathbf{u}_1\\ \mathbf{x}_2\boxplus\mathbf{u}_2\end{bmatrix}, \underbrace{\begin{bmatrix}\mathbf{x}_1\\ \mathbf{x}_2\end{bmatrix}}_{\mathbf{x}}\oplus\underbrace{\begin{bmatrix}\mathbf{v}_1\\ \mathbf{v}_2\end{bmatrix}}_{\mathbf{v}} = \begin{bmatrix}\mathbf{x}_1\oplus\mathbf{v}_1\\ \mathbf{x}_2\oplus\mathbf{v}_2\end{bmatrix}.
\tag{14}
$$

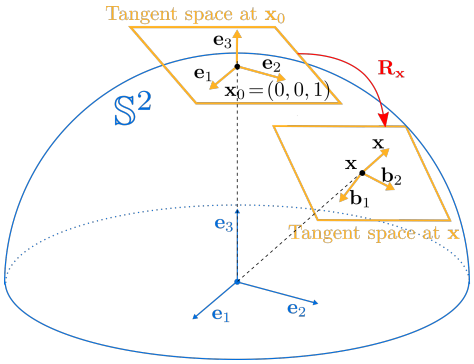

Fig. 4. Method adopted in [34] to obtain the orthonormal basis in the tangent plane on the $\mathcal{S}^2$ space.

As proved in Lemma 2 in Appendix B, the partial differentiation on the *compound manifold* is:

$$
\begin{aligned}
\frac{\partial((\mathbf{x}\boxplus\mathbf{u})\oplus\mathbf{v})\boxminus\mathbf{y})}{\partial\mathbf{u}} &= \begin{bmatrix}\frac{\partial((\mathbf{x}_1\boxplus\mathbf{u}_1)\oplus\mathbf{v}_1)\boxminus\mathbf{y}_1)}{\partial\mathbf{u}_1} & \mathbf{0}\\ \mathbf{0} & \frac{\partial((\mathbf{x}_2\boxplus\mathbf{u}_2)\oplus\mathbf{v}_2)\boxminus\mathbf{y}_2)}{\partial\mathbf{u}_2}\end{bmatrix}\\
\frac{\partial((\mathbf{x}\boxplus\mathbf{u})\oplus\mathbf{v})\boxminus\mathbf{y})}{\partial\mathbf{v}} &= \begin{bmatrix}\frac{\partial((\mathbf{x}_1\boxplus\mathbf{u}_1)\oplus\mathbf{v}_1)\boxminus\mathbf{y}_1)}{\partial\mathbf{v}_1} & \mathbf{0}\\ \mathbf{0} & \frac{\partial((\mathbf{x}_2\boxplus\mathbf{u}_2)\oplus\mathbf{v}_2)\boxminus\mathbf{y}_2)}{\partial\mathbf{v}_2}\end{bmatrix}
\end{aligned}
\tag{15}
$$

The $\boxplus\backslash\boxminus$ and $\oplus$ operations and their derivatives on a *compound manifold* are extremely useful, enabling us to define the $\boxplus\backslash\boxminus$ and $\oplus$ operations and their derivatives for *primitive manifolds* (e.g., $SO(3), \mathbb{R}^n, \mathbb{S}^2(r)$) only and then extend these definitions to more complicated *compound manifolds*.

IV. Canonical Representation

Consider a robotic system in discrete time with sampling period Δt, we can cast it into the following canonical form by zero-order hold discretization:

$$
\begin{aligned}
\mathbf{x}_{k+1} &= \mathbf{x}_k \oplus_{\mathcal{S}} \left(\Delta t\mathbf{f}\left(\mathbf{x}_k,\mathbf{u}_k,\mathbf{w}_k\right)\right), \mathbf{x}_k\in\mathcal{S},\\
\mathbf{z}_k &= \mathbf{h}\left(\mathbf{x}_k,\mathbf{v}_k\right), \mathbf{z}_k\in\mathcal{M},\\
\mathbf{w}_k &\sim\mathcal{N}\left(\mathbf{0},\mathcal{Q}_k\right), \mathbf{v}_k\sim\mathcal{N}\left(\mathbf{0},\mathcal{R}_k\right).
\end{aligned}
\tag{16}
$$

where the measurement $\mathbf{z}_k$ is assumed to be on the manifold $\mathcal{M}$ of dimension m. This is the case such as loosely-coupled visual-inertial odometry or lidar-inertial odometry where the measurements is a pose, an element in $SE(3)$. When compared to higher-order discretization methods (e.g., Runge-Kutta integration) used in prior work [8, 12], the zero-order hold discretization is usually less accurate. However, such difference is negligible when the sampling period is small.

In the following, we show how to cast different state components into the canonical form in (16). Then with the composition property (14), the complete state equation can be obtained by concatenating all components.

Example 1: Vectors in Euclidean space (e.g., position and velocity). Assume $\mathbf{x}\in\mathbb{R}^n$ subject to $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x},\mathbf{u},\mathbf{w})$. Using zero-order hold discretization, $\mathbf{f}(\mathbf{x},\mathbf{u},\mathbf{w})$ is assumed constant during the sampling period Δt, hence

$$
\begin{aligned}
\mathbf{x}_{k+1} &= \mathbf{x}_k + \left(\Delta t\mathbf{f}\left(\mathbf{x}_k,\mathbf{u}_k,\mathbf{w}_k\right)\right)\\
&= \mathbf{x}_k \oplus_{\mathbb{R}^n} \left(\Delta t\mathbf{f}\left(\mathbf{x}_k,\mathbf{u}_k,\mathbf{w}_k\right)\right).
\end{aligned}
\tag{17}
$$

Example 2: Attitude kinematics in a global reference frame (e.g., the earth-frame). Let $\mathbf{x} \in SO(3)$ be the body attitude relative to the global frame and ${}^{G}\boldsymbol{\omega}$ be the global angular velocity which holds constant for one sampling period Δt, then

$$\begin{aligned}\dot{\mathbf{x}} = \lfloor {}^{G}\boldsymbol{\omega} \rfloor \cdot \mathbf{x} \Longrightarrow \mathbf{x}_{k+1} = \mathrm{Exp}(\Delta t\ {}^{G}\boldsymbol{\omega}_k) \cdot \mathbf{x}_k = \mathbf{x}_k \\ \cdot \mathrm{Exp}\left(\Delta t(\mathbf{x}_k^T \cdot {}^{G}\boldsymbol{\omega}_k)\right) = \mathbf{x}_k \oplus_{SO(3)} \left(\Delta t \mathbf{f}\left(\mathbf{x}_k, {}^{G}\boldsymbol{\omega}_k\right)\right), \\ \mathbf{f}\left(\mathbf{x}_k, {}^{G}\boldsymbol{\omega}_k\right) = \mathbf{x}_k^T \cdot {}^{G}\boldsymbol{\omega}_k.\end{aligned} \tag{18}$$

Example 3: Attitude kinematics in body frame. Let $\mathbf{x} \in SO(3)$ be the body attitude relative to the global frame and ${}^{B}\boldsymbol{\omega}$ be the body angular velocity which holds constant for one sampling period Δt, then

$$\begin{aligned}\dot{\mathbf{x}} = \mathbf{x} \cdot \lfloor {}^{B}\boldsymbol{\omega} \rfloor \Longrightarrow \mathbf{x}_{k+1} = \mathbf{x}_k \cdot \mathrm{Exp}(\Delta t\ {}^{B}\boldsymbol{\omega}_k) \\ = \mathbf{x}_k \oplus_{SO(3)} \left(\Delta t \mathbf{f}\left({}^{B}\boldsymbol{\omega}_k\right)\right), \ \mathbf{f}\left({}^{B}\boldsymbol{\omega}_k\right) = {}^{B}\boldsymbol{\omega}_k.\end{aligned} \tag{19}$$

Example 4: Vectors of known magnitude (e.g., gravity) in the global frame. Let $\mathbf{x} \in \mathbb{S}^2(g)$ be the gravity vector in the global frame with known magnitude g. Then,

$$\dot{\mathbf{x}} = \mathbf{0} \Longrightarrow \mathbf{x}_{k+1} = \mathbf{x}_k = \mathbf{x}_k \oplus_{\mathbb{S}^2(g)} (\Delta t \mathbf{f}(\mathbf{x}_k)), \mathbf{f}(\mathbf{x}_k) = \mathbf{0}. \tag{20}$$

Example 5: Vectors of known magnitude (e.g., gravity) in body frame. Let $\mathbf{x} \in \mathbb{S}^2(g)$ be the gravity vector in the body frame and ${}^{B}\boldsymbol{\omega}$ be the body angular velocity which holds constant for one sampling period Δt. Then,

$$\begin{aligned}\dot{\mathbf{x}} = -\lfloor {}^{B}\boldsymbol{\omega} \rfloor \mathbf{x} \Longrightarrow \mathbf{x}_{k+1} = \mathrm{Exp}(-\Delta t\ {}^{B}\boldsymbol{\omega}_k)\mathbf{x}_k \\ = \mathbf{x}_k \oplus_{\mathbb{S}^2(g)} \left(\Delta t \mathbf{f}\left({}^{B}\boldsymbol{\omega}_k\right)\right), \mathbf{f}\left({}^{B}\boldsymbol{\omega}_k\right) = -{}^{B}\boldsymbol{\omega}_k.\end{aligned} \tag{21}$$

Example 6: Bearing-distance parameterization of visual landmarks [11]. Let $\mathbf{x} \in \mathbb{S}^2(1)$ and $d(\rho) \in \mathbb{R}$ be the bearing vector and depth (with parameter ρ), respectively, of a visual landmark, and ${}^{G}\mathbf{R}_C, {}^{G}\mathbf{p}_C$ be the attitude and position of the camera. Then the visual landmark in the global frame is ${}^{G}\mathbf{R}_C(\mathbf{x}d(\rho)) + {}^{G}\mathbf{p}_C$, which is constant over time:

$$\begin{aligned}\frac{d({}^{G}\mathbf{R}_C(\mathbf{x}d(\rho)) + {}^{G}\mathbf{p}_C)}{dt} = \mathbf{0} \Longrightarrow \\ \lfloor {}^{C}\boldsymbol{\omega} \rfloor (\mathbf{x}d(\rho)) + \dot{\mathbf{x}}d(\rho) + \mathbf{x}d'(\rho)\dot{\rho} + {}^{C}\mathbf{v} = \mathbf{0}.\end{aligned} \tag{22}$$

Left multiplying (22) by $\mathbf{x}^T$ and using $\mathbf{x}^T\dot{\mathbf{x}} = 0$ yield $\dot{\rho} = -\mathbf{x}^T \cdot {}^{C}\mathbf{v}/d'(\rho)$. Substituting this to (22) leads to

$$\begin{aligned}\dot{\mathbf{x}} = -\lfloor {}^{C}\boldsymbol{\omega} + \tfrac{1}{d(\rho)} \lfloor \mathbf{x} \rfloor \cdot {}^{C}\mathbf{v} \rfloor \cdot \mathbf{x} \Longrightarrow \\ \mathbf{x}_{k+1} = \mathrm{Exp}\left(-\Delta t \left({}^{C}\boldsymbol{\omega}_k + \tfrac{1}{d(\rho)} \lfloor \mathbf{x}_k \rfloor \cdot {}^{C}\mathbf{v}_k\right)\right) \mathbf{x}_k \\ = \mathbf{x}_k \oplus_{\mathbb{S}^2(1)} \left(\Delta t \mathbf{f}\left(\mathbf{x}_k, {}^{C}\boldsymbol{\omega}_k, {}^{C}\mathbf{v}_k\right)\right), \\ \mathbf{f}\left(\mathbf{x}_k, {}^{C}\boldsymbol{\omega}_k, {}^{C}\mathbf{v}_k\right) = -{}^{C}\boldsymbol{\omega}_k - \tfrac{1}{d(\rho)} \lfloor \mathbf{x}_k \rfloor \cdot {}^{C}\mathbf{v}_k.\end{aligned} \tag{23}$$

where ${}^{C}\boldsymbol{\omega} + \frac{1}{d(\rho)} \lfloor \mathbf{x} \rfloor \cdot {}^{C}\mathbf{v}$ is assumed constant for one sampling period Δt due to the zero-order hold assumption.

V. Error-State Kalman Filters on Manifolds

In this chapter, we derive a symbolic Kalman filter based on the canonical system representation (16). To avoid singularity of the minimal parameterization of the system original state which lies on manifolds, we employ the error-state idea that has been previously studied in prior work such as [6] and [16]. The presented derivation is very abstract, although being more concise, compact and generic. Moreover, for a complete treatment, we derive the full multi-rate iterated Kalman filter. Readers may refer to [6] for more detailed derivations/explanations or [16] for a brief derivation on a concrete example.

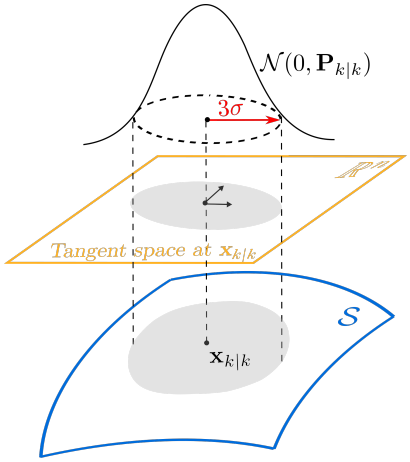

Fig. 5. The covariance matrix ($\mathbf{P}_{k|k}$) of the error state ($\delta\mathbf{x}_{k|k}$).

In the following presentations, we use the below notations:

(i) $\mathcal{S}$ denotes the manifold that the state $\mathbf{x}$ lies on. And $\mathcal{M}$ denotes the manifold that the measurement $\mathbf{z}$ lies on. For sake of notation simplification, we drop the subscripts $\mathcal{S}, \mathcal{M}$ for $\boxplus$ and $\boxminus$ when the context is made clear.

(ii) The subscript k denotes the time index, e.g., $\mathbf{x}_k$ is the ground truth of the state $\mathbf{x}$ at step k.

(iii) The subscript $\tau|k$ denotes the estimation of a quantity at step τ based on all the measurements up to step k, e.g., $\mathbf{x}_{\tau|k}$ means the estimation of state $\mathbf{x}_\tau$ based on measurements up to step k. For filtering problem, it requires $\tau \geq k$. More specifically, we have $\tau > k$ for state predict (i.e., prior estimate) and $\tau = k$ for state update (i.e., posteriori estimate).

(iv) $\delta\mathbf{x}_{\tau|k} = \mathbf{x}_\tau \boxminus \mathbf{x}_{\tau|k}$ denotes the estimation error in the tangent space of $\mathbf{x}_{\tau|k}$. It is a random vector in $\mathbb{R}^n$ since the ground true state $\mathbf{x}$ is random.

(v) $\mathbf{P}_{\tau|k}$ denotes the covariance of the estimation error $\delta\mathbf{x}_{\tau|k}$.

(vi) superscript j denotes the j-th iteration of the iterated Kalman filter, e.g. $\mathbf{x}_{k|k}^j$ denotes the estimate of state $\mathbf{x}_k$ at the j-th iteration based on measurements up to step k.

A. Initialization

Assume we have received measurements up to step k and updated the state at that time step as $\mathbf{x}_{k|k}$ along with the updated covariance matrix $\mathbf{P}_{k|k}$. According to the notation conventions above, $\mathbf{P}_{k|k}$ denotes the covariance of $\delta\mathbf{x}_{k|k}$, an error in the tangent space of the state update $\mathbf{x}_{k|k}$. The relation between $\delta\mathbf{x}_{k|k}$ and $\mathbf{P}_{k|k}$ is shown in Fig. 5.

B. State propagation

The state propagation from step k follows directly from the system model in equation (16) by setting $\mathbf{w} = \mathbf{0}$:

$$\mathbf{x}_{\tau+1|k} = \mathbf{x}_{\tau|k} \oplus \left(\Delta t \mathbf{f}\left(\mathbf{x}_{\tau|k}, \mathbf{u}_{\tau|k}, \mathbf{0}\right)\right); \tau \geq k \tag{24}$$

If only one step needs to be propagated, which is usually the case for measurements being the same sampling rate as that of the input, then $\tau = k$. Otherwise, the propagation proceeds at each input and stops when a measurement comes.

C. The error-state system

The error-state Kalman filter propagates the covariance matrix in the error state in order to avoid the the over-parameterization in $\mathbf{x}$. The error state is defined for $\tau \geq k$ as follows

$$\delta\mathbf{x}_{\tau|k} = \mathbf{x}_\tau \boxminus \mathbf{x}_{\tau|k}, \tau \geq k. \tag{25}$$

Substituting (16) and (24) into (25) leads to

$$\begin{aligned}\delta\mathbf{x}_{\tau+1|k} = \mathbf{x}_{\tau+1} \boxminus \mathbf{x}_{\tau+1|k} =& \left(\mathbf{x}_\tau \oplus \left(\Delta t \mathbf{f}\left(\mathbf{x}_\tau, \mathbf{u}_\tau, \mathbf{w}_\tau\right)\right)\right) \\ & \boxminus \left(\mathbf{x}_{\tau|k} \oplus \left(\Delta t \mathbf{f}\left(\mathbf{x}_{\tau|k}, \mathbf{u}_\tau, \mathbf{0}\right)\right)\right).\end{aligned} \tag{26}$$

Then substituting (25) into the above equation leads to

$$\begin{aligned}\delta\mathbf{x}_{\tau+1|k} = & \left(\left(\mathbf{x}_{\tau|k} \boxplus \delta\mathbf{x}_{\tau|k}\right) \oplus \left(\Delta t \mathbf{f}\left(\mathbf{x}_{\tau|k} \boxplus \delta\mathbf{x}_{\tau|k}, \mathbf{u}_\tau, \mathbf{w}_\tau\right)\right)\right) \\ & \boxminus \left(\mathbf{x}_{\tau|k} \oplus \left(\Delta t \mathbf{f}\left(\mathbf{x}_{\tau|k}, \mathbf{u}_\tau, \mathbf{0}\right)\right)\right),\end{aligned} \tag{27}$$

which defines a *new* system starting from $\tau = k$. This system describes the time evolvement of error state $\delta\mathbf{x}_{\tau|k}$ and hence is referred to as the error-state system. Since the new error-state system originates from the current measurement time k, it is re-defined once a new measurement is received to update the state estimate. Such a repeating process effectively restricts the error trajectory within a neighbor of the identity, validating the minimal parameterization in $\delta\mathbf{x}_{\tau|k}$. In case $\mathcal{S}$ is a Lie group, the error state in tangent space of $\mathbf{x}_{\tau|k}$ is $\delta\mathbf{x}_{\tau|k} = \text{Log}\left(\mathbf{x}_{\tau|k}^{-1} \cdot \mathbf{x}_\tau\right)$. Define $\widetilde{\mathbf{x}}_{\tau|k} = \mathbf{x}_\tau^{-1} \cdot \mathbf{x}_\tau$ the error state on the original manifold $\mathcal{S}$, the relation between the two trajectories $\delta\mathbf{x}_{\tau|k}$ and $\widetilde{\mathbf{x}}_{\tau|k}$ is shown in Fig. 1.

Since the error system (27) has minimal parameterization, the standard Kalman filter variants could be employed. Accordingly, the two Kalman filter steps, propagation and update, are referred to as "error-state propagation" and "error-state update", respectively, in order to distinguish from the original state space (16). In the following, we show in detail the error-state propagation and error-state update.

1) Initial condition: The error system (27) starts from $\tau = k$. The initial estimation is

$$\delta\mathbf{x}_{(k|k)|k} = \left(\mathbf{x}_k \boxminus \mathbf{x}_{k|k}\right)_{|k} = \mathbf{x}_{k|k} \boxminus \mathbf{x}_{k|k} = \mathbf{0} \tag{28}$$

Here, the notation $\delta\mathbf{x}_{(k|k)|k}$ denotes the estimation of the random vector $\delta\mathbf{x}_{k|k}$ (recall that this is indeed random due to its definition in (25) and that the ground truth state $\mathbf{x}_k$ is random) based on measurements up to k. The result in (28) is not surprising as $\delta\mathbf{x}_{k|k}$ is the error after conditioning on the measurements (up to k) already, so conditioning on the same measurements again does not give more information.

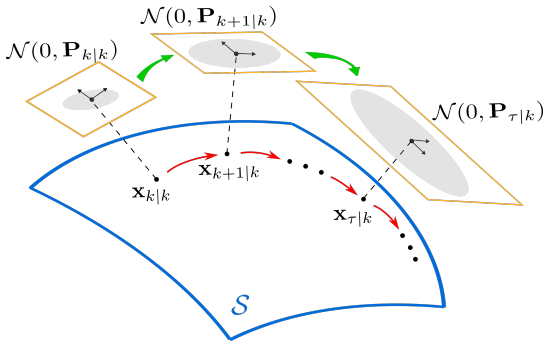

Fig. 6. Propagation of the state (red arrows on the manifold) and its covariance (green arrows on the tangent planes).

2) Error state propagation: The error state propagation follows directly from the error-state system model in (27) by setting $\mathbf{w} = \mathbf{0}$:

$$\begin{aligned}\delta\mathbf{x}_{(\tau+1|k)|k} = & \left(\left(\mathbf{x}_{\tau|k} \boxplus \delta\mathbf{x}_{(\tau|k)|k}\right)\right. \\ & \left.\oplus \left(\Delta t \mathbf{f}\left(\mathbf{x}_{\tau|k} \boxplus \delta\mathbf{x}_{(\tau|k)|k}, \mathbf{u}_\tau, \mathbf{0}\right)\right)\right) \\ & \boxminus \left(\mathbf{x}_{\tau|k} \oplus \left(\Delta t \mathbf{f}\left(\mathbf{x}_{\tau|k}, \mathbf{u}_\tau, \mathbf{0}\right)\right)\right); \tau \geq k\end{aligned} \tag{29}$$

Starting from the initial condition in (28), we obtain

$$\delta\mathbf{x}_{(\tau|k)|k} = \mathbf{0}; \forall \tau \geq k. \tag{30}$$

Next, to propagate the error covariance, we need to linearize the system (27) as follows

$$\delta\mathbf{x}_{\tau+1|k} \approx \mathbf{F}_{\mathbf{x}_\tau} \delta\mathbf{x}_{\tau|k} + \mathbf{F}_{\mathbf{w}_\tau} \mathbf{w}_\tau \tag{31}$$

where $\mathbf{F}_{\mathbf{x}_\tau}$ is the partial differention of (27) w.r.t $\delta\mathbf{x}_{\tau|k}$ at point $\delta\mathbf{x}_{(\tau|k)|k} = \mathbf{0}$, as follows

$$\begin{aligned}\mathbf{F}_{\mathbf{x}_\tau} = & \frac{\partial\left(\left(\mathbf{x}_{\tau|k} \boxplus \delta\mathbf{x}_{\tau|k}\right) \oplus \left(\Delta t \mathbf{f}\left(\mathbf{x}_{\tau|k}, \mathbf{u}_\tau, \mathbf{0}\right)\right)\right) \boxminus \left(\mathbf{x}_{\tau|k} \oplus \left(\Delta t \mathbf{f}\left(\mathbf{x}_{\tau|k}, \mathbf{u}_\tau, \mathbf{0}\right)\right)\right)}{\partial \delta\mathbf{x}_{\tau|k}} \\ & + \frac{\partial\left(\mathbf{x}_{\tau|k} \oplus \left(\Delta t \mathbf{f}\left(\mathbf{x}_{\tau|k} \boxplus \delta\mathbf{x}_{\tau|k}, \mathbf{u}_\tau, \mathbf{0}\right)\right)\right) \boxminus \left(\mathbf{x}_{\tau|k} \oplus \left(\Delta t \mathbf{f}\left(\mathbf{x}_{\tau|k}, \mathbf{u}_\tau, \mathbf{0}\right)\right)\right)}{\partial \delta\mathbf{x}_{\tau|k}} \\ = & \mathbf{G}_{\mathbf{x}_\tau} + \Delta t \mathbf{G}_{\mathbf{f}_\tau} \frac{\partial \mathbf{f}\left(\mathbf{x}_{\tau|k} \boxplus \delta\mathbf{x}, \mathbf{u}_\tau, \mathbf{0}\right)}{\partial \delta\mathbf{x}}\Big|_{\delta\mathbf{x}=\mathbf{0}}\end{aligned} \tag{32}$$

and $\mathbf{F}_{\mathbf{w}_\tau}$ is the partial differentiation of (27) w.r.t $\mathbf{w}_\tau$ at the point $\mathbf{w}_\tau = \mathbf{0}$, as follows

$$\begin{aligned}\mathbf{F}_{\mathbf{w}_\tau} = & \frac{\partial\left(\mathbf{x}_{\tau|k} \oplus \left(\Delta t \mathbf{f}\left(\mathbf{x}_{\tau|k}, \mathbf{u}_\tau, \mathbf{w}_\tau\right)\right)\right) \boxminus \left(\mathbf{x}_{\tau|k} \oplus \left(\Delta t \mathbf{f}\left(\mathbf{x}_{\tau|k}, \mathbf{u}_\tau, \mathbf{0}\right)\right)\right)}{\partial \mathbf{w}_\tau} \\ = & \Delta t \mathbf{G}_{\mathbf{f}_\tau} \frac{\partial \mathbf{f}\left(\mathbf{x}_{\tau|k}, \mathbf{u}_\tau, \mathbf{w}\right)}{\partial \mathbf{w}}\Big|_{\mathbf{w}=\mathbf{0}}\end{aligned} \tag{33}$$

where

$$\begin{aligned}\mathbf{G}_{\mathbf{x}_\tau} = & \frac{\partial(((\mathbf{x} \boxplus \mathbf{u}) \oplus \mathbf{v}) \boxminus \mathbf{y})}{\partial \mathbf{u}}\Big|_{\substack{\mathbf{x}=\mathbf{x}_{\tau|k}; \mathbf{u}=\mathbf{0}; \mathbf{v}=\Delta t \mathbf{f}\left(\mathbf{x}_{\tau|k}, \mathbf{u}_\tau, \mathbf{0}\right); \\ \mathbf{y}=\mathbf{x}_{\tau|k} \oplus \Delta t \mathbf{f}\left(\mathbf{x}_{\tau|k}, \mathbf{u}_\tau, \mathbf{0}\right)}} \\ \mathbf{G}_{\mathbf{f}_\tau} = & \frac{\partial(((\mathbf{x} \boxplus \mathbf{u}) \oplus \mathbf{v}) \boxminus \mathbf{y})}{\partial \mathbf{v}}\Big|_{\substack{\mathbf{x}=\mathbf{x}_{\tau|k}; \mathbf{u}=\mathbf{0}; \mathbf{v}=\Delta t \mathbf{f}\left(\mathbf{x}_{\tau|k}, \mathbf{u}_\tau, \mathbf{0}\right); \\ \mathbf{y}=\mathbf{x}_{\tau|k} \oplus \Delta t \mathbf{f}\left(\mathbf{x}_{\tau|k}, \mathbf{u}_\tau, \mathbf{0}\right)}}\end{aligned} \tag{34}$$

Finally, the covariance is propagated as

$$\mathbf{P}_{\tau+1|k} = \mathbf{F}_{\mathbf{x}_\tau} \mathbf{P}_{\tau|k} \mathbf{F}_{\mathbf{x}_\tau}^T + \mathbf{F}_{\mathbf{w}_\tau} \mathcal{Q}_\tau \mathbf{F}_{\mathbf{w}_\tau}^T \tag{35}$$

The propagation of the state in (24) and respective covariance in (35) are illustrated in Fig. 6.

3) Isolation of manifold structures: As shown by (32) and (33), the two system matrices $\mathbf{F}_{\mathbf{x}_\tau}, \mathbf{F}_{\mathbf{w}_\tau}$ are well separated into manifold-specific parts $\mathbf{G}_{\mathbf{x}_\tau}, \mathbf{G}_{\mathbf{f}_\tau}$ and system-specific parts $\frac{\partial \mathbf{f}\left(\mathbf{x}_{\tau|k} \boxplus \delta\mathbf{x}, \mathbf{u}_\tau, \mathbf{0}\right)}{\partial \delta\mathbf{x}}|_{\delta\mathbf{x}=\mathbf{0}}, \frac{\partial \mathbf{f}\left(\mathbf{x}_{\tau|k}, \mathbf{u}_\tau, \mathbf{w}\right)}{\partial \mathbf{w}}|_{\mathbf{w}=\mathbf{0}}$. The manifold-specific parts for commonly used manifolds are listed in TABLE I. Moreover, based on (15), the manifold-specific parts for any compound manifold is the concatenation of that of these primitive manifolds.

TABLE I
MANIFOLD-SPECIFIC PARTS FOR $\mathbf{G}_{\mathbf{x}_\tau}, \mathbf{G}_{\mathbf{f}_\tau}$

$\mathcal{S}$	$\mathbf{G}_{\mathbf{x}_\tau}$
$\mathbb{R}^n$	$\mathbf{I}_{n\times n}$
$SO(3)$	$\mathrm{Exp}\left(-\Delta t \mathbf{f}\left(\mathbf{x}_{\tau\|k}, \mathbf{u}_\tau, \mathbf{0}\right)\right)$
$\mathbb{S}^2(r)$	$-\frac{1}{r^2}\mathbf{B}\left(\mathbf{x}_{\tau+1|k}\right)^T \mathrm{Exp}\left(\Delta t \mathbf{f}\left(\mathbf{x}_{\tau|k}, \mathbf{u}_\tau, \mathbf{0}\right)\right)$ $\cdot \lfloor \mathbf{x}_{\tau|k} \rfloor^2 \mathbf{B}\left(\mathbf{x}_{\tau|k}\right)$

$\mathcal{S}$	$\mathbf{G}_{\mathbf{f}_\tau}$
$\mathbb{R}^n$	$\mathbf{I}_{n\times n}$
$SO(3)$	$\mathbf{A}\left(\Delta t \mathbf{f}\left(\mathbf{x}_{\tau|k}, \mathbf{u}_\tau, \mathbf{0}\right)\right)^T$
$\mathbb{S}^2(r)$	$-\frac{1}{r^2}\mathbf{B}\left(\mathbf{x}_{\tau+1|k}\right)^T \mathrm{Exp}\left(\Delta t \mathbf{f}\left(\mathbf{x}_{\tau|k}, \mathbf{u}_\tau, \mathbf{0}\right)\right)$ $\cdot \lfloor \mathbf{x}_{\tau|k} \rfloor^2 \mathbf{A}\left(\Delta t \mathbf{f}\left(\mathbf{x}_{\tau|k}, \mathbf{u}_\tau, \mathbf{0}\right)\right)^T$

D. State update

1) Prior distribution: Assume a measurement arrives at step $\tau > k$. Without the loss of generality, we assume $\tau = k+1$, i.e., the measurement rate is equal to the input rate. The propagated error state $\delta\mathbf{x}_{k+1|k}$ and its covariance $\mathbf{P}_{k+1|k}$ create a prior distribution for $\mathbf{x}_{k+1}$:

$$\delta\mathbf{x}_{k+1|k} = \mathbf{x}_{k+1} \boxminus \mathbf{x}_{k+1|k} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{P}_{k+1|k}\right) \tag{36}$$

2) Iterated update: Now assume the new measurement at $k+1$ is $\mathbf{z}_{k+1}$. In the j-th iteration, the state estimate is $\mathbf{x}^j_{k+1|k+1}$, where $\mathbf{x}^j_{k+1|k+1} = \mathbf{x}_{k+1|k}$ (i.e., the priori estimate) for $j = 0$, then define the residual

$$\begin{aligned}\mathbf{r}^j_{k+1} &\triangleq \mathbf{z}_{k+1} \boxminus \mathbf{h}(\mathbf{x}^j_{k+1|k+1}, \mathbf{0}) \\ &= \mathbf{h}(\mathbf{x}_{k+1}, \mathbf{v}_{k+1}) \boxminus \mathbf{h}(\mathbf{x}^j_{k+1|k+1}, \mathbf{0}) \\ &= \mathbf{h}(\mathbf{x}^j_{k+1|k+1} \boxplus \delta\mathbf{x}_j, \mathbf{v}_{k+1}) \boxminus \mathbf{h}(\mathbf{x}^j_{k+1|k+1}, \mathbf{0}) \\ &\approx \mathbf{D}^j_{k+1}\mathbf{v}_{k+1} + \mathbf{H}^j_{k+1}\delta\mathbf{x}_j\end{aligned} \tag{37}$$

where $\delta\mathbf{x}_j \triangleq \mathbf{x}_{k+1} \boxminus \mathbf{x}^j_{k+1|k+1}$ is the error between the ground true state $\mathbf{x}_{k+1}$ and its most recent estimate $\mathbf{x}^j_{k+1|k+1}$, and

$$\begin{aligned}\mathbf{H}^j_{k+1} &= \frac{\partial\left(\mathbf{h}(\mathbf{x}^j_{k+1|k+1} \boxplus \delta\mathbf{x}, \mathbf{0}) \boxminus \mathbf{h}(\mathbf{x}^j_{k+1|k+1}, \mathbf{0})\right)}{\partial \delta\mathbf{x}}|_{\delta\mathbf{x}=\mathbf{0}} \\ &= \frac{\partial \mathbf{h}(\mathbf{x}^j_{k+1|k+1} \boxplus \delta\mathbf{x}, \mathbf{0})}{\partial \delta\mathbf{x}}|_{\delta\mathbf{x}=\mathbf{0}}, \text{ for } \mathcal{M} = \mathbb{R}^m, \\ \mathbf{D}^j_{k+1} &= \frac{\partial\left(\mathbf{h}(\mathbf{x}^j_{k+1|k+1}, \mathbf{v}) \boxminus \mathbf{h}(\mathbf{x}^j_{k+1|k+1}, \mathbf{0})\right)}{\partial \mathbf{v}}|_{\mathbf{v}=\mathbf{0}} \\ &= \frac{\partial \mathbf{h}(\mathbf{x}^j_{k+1|k+1}, \mathbf{v})}{\partial \mathbf{v}}|_{\mathbf{v}=\mathbf{0}}, \text{ for } \mathcal{M} = \mathbb{R}^m\end{aligned} \tag{38}$$

Equation (37) defines a posteriori distribution for $\delta\mathbf{x}_j$

$$\begin{aligned}&(\mathbf{D}^j_{k+1}\mathbf{v}_{k+1}) | \delta\mathbf{x}_j = \mathbf{r}^j_{k+1} - \mathbf{H}^j_{k+1}\delta\mathbf{x}_j \sim \mathcal{N}\left(\mathbf{0}, \bar{\mathcal{R}}_{k+1}\right); \\ &\bar{\mathcal{R}}_{k+1} = \mathbf{D}^j_{k+1}\mathcal{R}_{k+1}(\mathbf{D}^j_{k+1})^T\end{aligned} \tag{39}$$

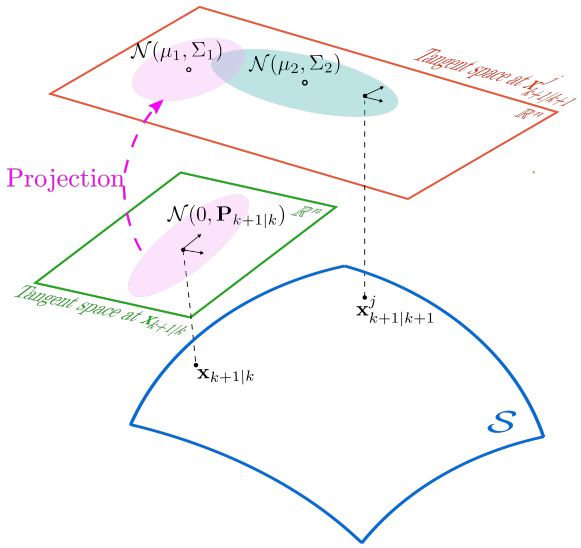

Fig. 7. Prior distribution $\mathcal{N}(\mathbf{0}, \mathbf{P}_{k+1|k})$, its projection $\mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$, and posteriori distribution $\mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$, where $\boldsymbol{\mu}_1 = -\mathbf{J}^j_{k+1}(\mathbf{x}^j_{k+1|k+1} \boxminus \mathbf{x}_{k+1|k})$, $\boldsymbol{\Sigma}_1 = \mathbf{J}^j_{k+1}\mathbf{P}_{k+1|k}(\mathbf{J}^j_{k+1})^T$ and $\boldsymbol{\mu}_2 = (\mathbf{H}^j_{k+1})^{-1}\mathbf{r}^j_{k+1}$, $\boldsymbol{\Sigma}_2 = (\mathbf{H}^j_{k+1})^{-1}\bar{\mathcal{R}}_{k+1}(\mathbf{H}^j_{k+1})^{-T}$.

On the other hand, (36) defines a distribution for the prior estimation error $\delta\mathbf{x}_{k+1|k} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{P}_{k+1|k}\right)$, which is in the tangent space of $\mathbf{x}_{k+1|k}$. As shown in Fig. 7, projecting $\delta\mathbf{x}_{k+1|k}$ to the tangent space of $\mathbf{x}^j_{k+1|k+1}$ leads to

$$\begin{aligned}\delta\mathbf{x}_{k+1|k} &= \mathbf{x}_{k+1} \boxminus \mathbf{x}_{k+1|k} = (\mathbf{x}^j_{k+1|k+1} \boxplus \delta\mathbf{x}_j) \boxminus \mathbf{x}_{k+1|k} \\ &= (\mathbf{x}^j_{k+1|k+1} \boxminus \mathbf{x}_{k+1|k}) + (\mathbf{J}^j_{k+1})^{-1}\delta\mathbf{x}_j\end{aligned} \tag{40}$$

where

$$\mathbf{J}^j_{k+1} = \frac{\partial(((\mathbf{x} \boxplus \mathbf{u}) \oplus \mathbf{v}) \boxminus \mathbf{y})}{\partial \mathbf{u}}\Big|_{\substack{\mathbf{x}=\mathbf{x}_{k+1|k}, \mathbf{u}=\mathbf{x}^j_{k+1|k+1} \boxminus \mathbf{x}_{k+1|k}, \\ \mathbf{v}=\mathbf{0}, \mathbf{y}=\mathbf{x}^j_{k+1|k+1}}} \tag{41}$$

is the inverse Jacobian of $\delta\mathbf{x}_{k+1|k}$ with repect to (w.r.t.) $\delta\mathbf{x}_j$ evaluated at zero. Then, the equivalent prior distribution for $\delta\mathbf{x}_j$ is

$$\delta\mathbf{x}_j \sim \mathcal{N}(-\mathbf{J}^j_{k+1}(\mathbf{x}^j_{k+1|k+1} \boxminus \mathbf{x}_{k+1|k}), \mathbf{J}^j_{k+1}\mathbf{P}_{k+1|k}(\mathbf{J}^j_{k+1})^T) \tag{42}$$

Combing the prior distribution (42) and posteriori distribution (39) leads to the maximum a-posteriori estimate (MAP) of $\delta\mathbf{x}_j$ (see Fig. 7):

$$\begin{aligned}&\arg\max_{\delta\mathbf{x}_j} \log\left(\mathcal{N}(\delta\mathbf{x}_j)\mathcal{N}\left((\mathbf{D}^j_{k+1}\mathbf{v}_{k+1}) | \delta\mathbf{x}_j\right)\right) \\ &= \arg\min_{\delta\mathbf{x}_j} g(\delta\mathbf{x}_j); \; g(\delta\mathbf{x}_j) = \|\mathbf{r}^j_{k+1} - \mathbf{H}^j_{k+1}\delta\mathbf{x}_j\|^2_{\bar{\mathcal{R}}^{-\frac{1}{2}}_{k+1}} \\ &\quad + \|(\mathbf{x}^j_{k+1|k+1} \boxminus \mathbf{x}_{k+1|k}) + (\mathbf{J}^j_{k+1})^{-1}\delta\mathbf{x}_j\|^2_{\mathbf{P}^{-\frac{1}{2}}_{k+1|k}}\end{aligned} \tag{43}$$

where $\|\mathbf{x}\|^2_{\mathbf{A}} = \|\mathbf{A}\mathbf{x}\|^2 = \mathbf{x}^T\mathbf{A}^T\mathbf{A}\mathbf{x}$. The optimal solution

$\delta\mathbf{x}^o$ for (43) leads to the Kalman update [35] as below:

$$\begin{aligned}\delta\mathbf{x}_j^o &= -\mathbf{J}_{k+1}^j(\mathbf{x}_{k+1|k+1}^j \boxminus \mathbf{x}_{k+1|k}) \\ &\quad + \mathbf{K}_{k+1}^j(\mathbf{r}_{k+1}^j + \mathbf{H}_{k+1}^j\mathbf{J}_{k+1}^j(\mathbf{x}_{k+1|k+1}^j \boxminus \mathbf{x}_{k+1|k})) \\ \mathbf{K}_{k+1}^j &= \left(\mathbf{Q}_{k+1}^j\right)^{-1}(\mathbf{H}_{k+1}^j)^T\bar{\mathcal{R}}_{k+1}^{-1} \\ &= \mathbf{J}_{k+1}^j\mathbf{P}_{k+1|k}(\mathbf{J}_{k+1}^j)^T(\mathbf{H}_{k+1}^j)^T(\mathbf{S}_{k+1}^j)^{-1} \\ \mathbf{Q}_{k+1}^j &= (\mathbf{H}_{k+1}^j)^T\bar{\mathcal{R}}_{k+1}^{-1}\mathbf{H}_{k+1}^j + (\mathbf{J}_{k+1}^j)^{-T}\mathbf{P}_{k+1|k}^{-1}\left(\mathbf{J}_{k+1}^j\right)^{-1} \\ \mathbf{S}_{k+1}^j &= \mathbf{H}_{k+1}^j\mathbf{J}_{k+1}^j\mathbf{P}_{k+1|k}(\mathbf{J}_{k+1}^j)^T(\mathbf{H}_{k+1}^j)^T + \bar{\mathcal{R}}_{k+1}\end{aligned} \tag{44}$$

where $\mathbf{Q}_{k+1}^j$ is the Hessian matrix of (43) and its inverse represents the covariance of $\delta\mathbf{x}_j - \delta\mathbf{x}_j^o$, which can be furthermore written into the form below [35]

$$\begin{aligned}\mathbf{P}_{k+1}^j &= (\mathbf{Q}_{k+1}^j)^{-1} \\ &= (\mathbf{I} - \mathbf{K}_{k+1}^j\mathbf{H}_{k+1}^j)\mathbf{J}_{k+1}^j\mathbf{P}_{k+1|k}(\mathbf{J}_{k+1}^j)^T\end{aligned} \tag{45}$$

With the optimal $\delta\mathbf{x}_j^o$, the update of $\mathbf{x}_{k+1}$ estimate is then

$$\mathbf{x}_{k+1|k+1}^{j+1} = \mathbf{x}_{k+1|k+1}^j \boxplus \delta\mathbf{x}_j^o \tag{46}$$

The above process iterates until convergence or exceeding the maximum steps.

3) Covariance reset:

Assume the iterated update stops after $\kappa \geq 0$ iterations, resulting in a MAP estimate $\mathbf{x}_{k+1|k+1}^{\kappa+1}$ and covariance matrix $\mathbf{P}_{k+1}^{\kappa}$. Then $\mathbf{x}_{k+1|k+1}^{\kappa+1}$ becomes the Kalman update of $\mathbf{x}_{k+1}$

$$\mathbf{x}_{k+1|k+1} = \mathbf{x}_{k+1|k+1}^{\kappa+1} \tag{47}$$

which is passed to the next step of the Kalman filter. For the $\mathbf{P}_{k+1}^{\kappa}$, note that it describes the covariance of $\delta\mathbf{x}_\kappa - \delta\mathbf{x}_\kappa^o$ which is in the tangent space of $\mathbf{x}_{k+1|k+1}^{\kappa}$, while what required at the next step of the Kalman filter should be the covariance $\mathbf{P}_{k+1|k+1}$ describing error $\delta\mathbf{x}_{k+1|k+1}$ that is in the tangent space of $\mathbf{x}_{k+1|k+1}$ (see Section V-A). This discrepancy necessitates a projection step as shown in Fig. 8. According to the definition of the error state in (25), we have

$$\begin{aligned}\delta\mathbf{x}_{k+1|k+1} &= \mathbf{x}_{k+1} \boxminus \mathbf{x}_{k+1|k+1} = \mathbf{x}_{k+1} \boxminus \mathbf{x}_{k+1|k+1}^{\kappa+1} \\ \delta\mathbf{x}_\kappa &= \mathbf{x}_{k+1} \boxminus \mathbf{x}_{k+1|k+1}^{\kappa}\end{aligned} \tag{48}$$

which leads to

$$\begin{aligned}\delta\mathbf{x}_{k+1|k+1} &= (\mathbf{x}_{k+1|k+1}^{\kappa} \boxplus \delta\mathbf{x}_\kappa) \boxminus \mathbf{x}_{k+1|k+1}^{\kappa+1} \\ &= \mathbf{L}_{k+1}(\delta\mathbf{x}_\kappa - \delta\mathbf{x}_\kappa^o)\end{aligned} \tag{49}$$

where

$$\mathbf{L}_{k+1} = \left.\frac{\partial(((\mathbf{x}\boxplus\mathbf{u})\oplus\mathbf{v})\boxminus\mathbf{y})}{\partial\mathbf{u}}\right|_{\substack{\mathbf{x}=\mathbf{x}_{k+1|k+1}^{\kappa},\mathbf{u}=\delta\mathbf{x}_\kappa^o,\\ \mathbf{v}=\mathbf{0},\mathbf{y}=\mathbf{x}_{k+1|k+1}^{\kappa+1}}} \tag{50}$$

is the Jacobian of $\delta\mathbf{x}_{k+1|k+1}$ w.r.t. $\delta\mathbf{x}_\kappa$ evaluated at $\delta\mathbf{x}_\kappa^o$.

Finally, the covariance for $\delta\mathbf{x}_{k+1|k+1}$ is

$$\mathbf{P}_{k+1|k+1} = \mathbf{L}_{k+1}\mathbf{P}_{k+1}^{\kappa}\mathbf{L}_{k+1}^T \tag{51}$$

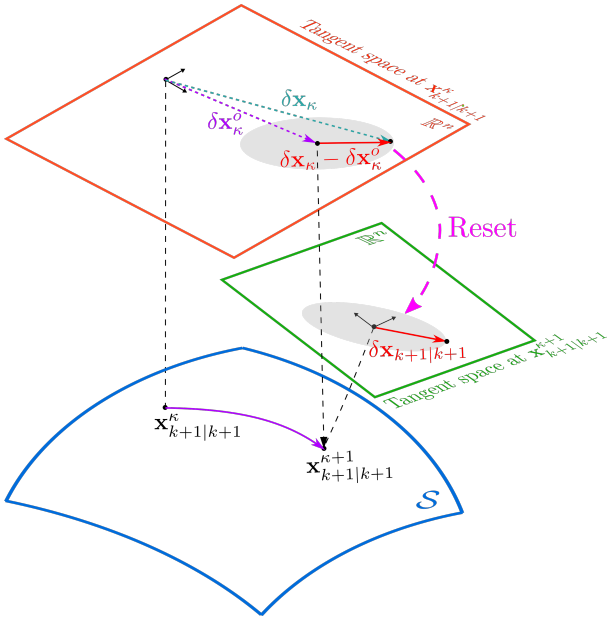

Fig. 8. Reset of covariance. The red and green $\mathbb{R}^n$ spaces are tangent with the $\mathcal{S}$ space at points $\mathbf{x}_{k+1|k+1}^{\kappa}$ and $\mathbf{x}_{k+1|k+1}^{\kappa+1}$ respectively.

4) Isolation of manifold structures: Notice that the two matrices $\mathbf{J}_{k+1}^j$and $\mathbf{L}_{k+1}$ required in the Kalman upudate only depend on the manifold $\mathcal{S}$ thus being manifold-specific matrices. Their values for commonly used manifolds are summarized in TABLE II. Again, the manifold-specific parts for any compound manifolds are the concatenation of these primitive manifolds. In particular, for an extended Kalman filter (i.e., $\kappa = 0$), $\mathbf{J}_{k+1}^{\kappa} = \mathbf{I}$ while $\mathbf{L}_{k+1} \neq \mathbf{I}$; for a fully converged iterated Kalman filter (i.e., κ is sufficiently large), $\mathbf{J}_{k+1}^{\kappa} \neq \mathbf{I}$ while $\mathbf{L}_{k+1} = \mathbf{I}$.

TABLE II
MANIFOLD-SPECIFIC PARTS FOR $\mathbf{J}_{k+1}^j$, $\mathbf{L}_{k+1}$

$\mathcal{S}$	$\mathbf{J}_{k+1}^j$
$\mathbb{R}^n$	$\mathbf{I}_{n\times n}$
$SO(3)$	$\mathbf{A}\left(\delta\mathbf{x}_{k+1|k+1}^j\right)^T$
$\mathbb{S}^2(r)$	$\frac{-1}{r^2}\mathbf{B}\left(\mathbf{x}_{k+1|k+1}^j\right)^T \mathrm{Exp}\left(\mathbf{B}\left(\mathbf{x}_{k+1|k}\right)\delta\mathbf{x}_{k+1|k+1}^j\right)$ $\cdot\lfloor\mathbf{x}_{k+1|k}\rfloor^2\mathbf{A}\left(\mathbf{B}\left(\mathbf{x}_{k+1|k}\right)\delta\mathbf{x}_{k+1|k+1}^j\right)^T\mathbf{B}\left(\mathbf{x}_{k+1|k}\right)$

$\mathcal{S}$	$\mathbf{L}_{k+1}$
$\mathbb{R}^n$	$\mathbf{I}_{n\times n}$
$SO(3)$	$\mathbf{A}\left(\delta\mathbf{x}_\kappa^o\right)^T$
$\mathbb{S}^2(r)$	$\frac{-1}{r^2}\mathbf{B}\left(\mathbf{x}_{k+1|k+1}^{\kappa+1}\right)^T \mathrm{Exp}\left(\mathbf{B}\left(\mathbf{x}_{k+1|k+1}^{\kappa}\right)\delta\mathbf{x}_\kappa^o\right)$ $\cdot\lfloor\mathbf{x}_{k+1|k+1}^{\kappa}\rfloor^2\mathbf{A}\left(\mathbf{B}\left(\mathbf{x}_{k+1|k+1}^{\kappa}\right)\delta\mathbf{x}_\kappa^o\right)^T\mathbf{B}\left(\mathbf{x}_{k+1|k+1}^{\kappa}\right)$

[1] $\delta\mathbf{x}_{k+1|k+1}^j = \mathbf{x}_{k+1|k+1}^j \boxminus \mathbf{x}_{k+1|k}$.

E. Error-state iterated Kalman filter on Manifolds

Summarizing all the above procedures in Section (V-A, V-B, V-C, V-D) leads to the full error-state iterated Kalman filter operating on manifolds (see Algorithm 1). Setting the number

of iteration $N_{\max}$ to zero leads to the error-state extended Kalman filter used in [6, 16].

Algorithm 1: Iterated error-state Kalman filter on manifolds

Input: $\mathbf{x}_{k|k}$, $\mathbf{P}_{k|k}$, $\mathbf{u}_k$, $\mathbf{z}_{k+1}$
Output: State update $\mathbf{x}_{k+1|k+1}$ and covariance $\mathbf{P}_{k+1|k+1}$
Prediction:

$$\mathbf{x}_{k+1|k} = \mathbf{x}_{k|k} \oplus \left(\Delta t \mathbf{f}\left(\mathbf{x}_{k|k}, \mathbf{u}_k, \mathbf{0}\right)\right);$$
$$\mathbf{P}_{k+1|k} = \mathbf{F}_{\mathbf{x}_k}\mathbf{P}_{k|k}\mathbf{F}_{\mathbf{x}_k}^T + \mathbf{F}_{\mathbf{w}_k}\mathcal{Q}_k\mathbf{F}_{\mathbf{w}_k}^T;$$

Update:

$j = -1$; $\mathbf{x}_{k+1|k+1}^0 = \mathbf{x}_{k+1|k}$;
while Not Converged and $j \leq N_{\max} - 1$ **do**
  $j = j + 1$;
  Calculate $\mathbf{r}_{k+1}^j$, $\mathbf{D}_{k+1}^j$, $\mathbf{H}_{k+1}^j$ as in (37) and (38);
  Calculate $\mathbf{J}_{k+1}^j$ as in (41);
  Calculate $\mathbf{K}_{k+1}^j$ and $\delta\mathbf{x}_j^o$ as in (44);
  $\mathbf{x}_{k+1|k+1}^{j+1} = \mathbf{x}_{k+1|k+1}^j \boxplus \delta\mathbf{x}_j^o$;
end while
$\mathbf{P}_{k+1}^j = (\mathbf{I} - \mathbf{K}_{k+1}^j\mathbf{H}_{k+1}^j)\mathbf{J}_{k+1}^j\mathbf{P}_{k+1|k}(\mathbf{J}_{k+1}^j)^T$;
$\mathbf{x}_{k+1|k+1} = \mathbf{x}_{k+1|k+1}^{j+1}$;
$\mathbf{P}_{k+1|k+1} = \mathbf{L}_{k+1}\mathbf{P}_{k+1}^j\mathbf{L}_{k+1}^T$;

VI. Embedding Manifold Structures into Kalman Filters and Toolkit Development

Shown in Section V, the derived Kalman filter is formulated in symbolic representations and it is seen that each step of the Kalman filter is nicely separated into manifold-specific parts and system-specific parts. More specifically, state propagation (24) breaks into the manifold-specific operation $\oplus$ and system-specific part $\Delta t\mathbf{f}\left(\mathbf{x}, \mathbf{u}, \mathbf{w}\right)$, the two matrices $\mathbf{F_x}$ and $\mathbf{F_w}$ used in the covariance propagation (35) breaks into the manifold-specific parts $\mathbf{G_x}$, $\mathbf{G_f}$ and system-specific parts $\frac{\partial \mathbf{f}(\mathbf{x}\boxplus\delta\mathbf{x},\mathbf{u},\mathbf{0})}{\partial\delta\mathbf{x}}|_{\delta\mathbf{x}=\mathbf{0}}$, $\frac{\partial \mathbf{f}(\mathbf{x},\mathbf{u},\mathbf{w})}{\partial\mathbf{w}}|_{\mathbf{w}=\mathbf{0}}$, the state update (44) breaks into the manifold-specific operation $\boxplus$, manifold-specific part $\mathbf{J}_{k+1}^j$ and system-specific parts, i.e., $\mathbf{h}(\mathbf{x}, \mathbf{v})$, $\frac{\partial(\mathbf{h}(\mathbf{x}\boxplus\delta\mathbf{x},\mathbf{0})\boxminus\mathbf{h}(\mathbf{x},\mathbf{0}))}{\partial\delta\mathbf{x}}|_{\delta\mathbf{x}=\mathbf{0}}$, and $\frac{\partial(\mathbf{h}(\mathbf{x},\mathbf{v})\boxminus\mathbf{h}(\mathbf{x},\mathbf{0}))}{\partial\mathbf{v}}|_{\mathbf{v}=\mathbf{0}}$. And covariance reset only involves the manifold-specific part $\mathbf{L}_{k+1}$. Note that these system-specific descriptions are often easy to derive even for robotic systems of high dimension (see Section VII).

The nice separation property between the manifold-specific parts and system-specific descriptions allows the embedding of the manifold structures into the Kalman filter framework, and only leaves system-specific parts to be filled for specific systems. Moreover, enabled by the manifold composition in (14) and (15), we only need to do so for simple *primitive manifolds* while those for larger *compound manifolds* can be automatically constructed. These two properties enabled us to develop a *C++* toolkit that encapsulates the manifold-specific operations with a Kalman filter. With this toolkit, users need only to specify the manifold of state $\mathcal{S}$, measurement $\mathcal{M}$, and system-specific descriptions (i.e., function $\mathbf{f}, \mathbf{h}$ and their derivatives), and call the respective Kalman filter operations (i.e., propagation and update) according to the current event (e.g., reception of an input or a measurement).

The current toolkit implementation is a full multi-rate iterated Kalman filter naturally operating on manifolds and is thus termed as *IKFoM*. Furthermore, it supports three *primitive manifolds*: $\mathbb{R}^n, SO(3)$ and $\mathbb{S}^2(r)$, but extendable to other types of primitive manifolds with proper definition of the operation $\boxplus, \oplus$, and differentiations $\frac{\partial(((\mathbf{x}\boxplus\mathbf{u})\oplus\mathbf{v})\boxminus\mathbf{y})}{\partial\mathbf{u}}$, $\frac{\partial(((\mathbf{x}\boxplus\mathbf{u})\oplus\mathbf{v})\boxminus\mathbf{y})}{\partial\mathbf{v}}$. The toolkit is open sourced and more details about the implementation can be found at https://github.com/hku-mars/IKFoM.

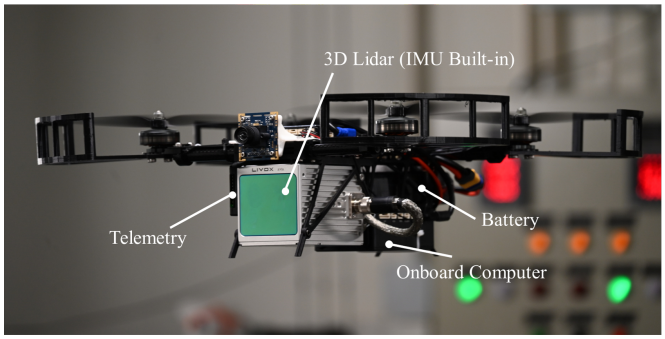

Fig. 9. Configuration of the lidar-inertial system from [15]: A small scale ($280mm$ wheelbase) quadrotor UAV carrying a Livox AVIA lidar and a DJI Manifold 2C computer. The onboard camera is for visualization only.

VII. Experiments

In this section, we apply our developed Kalman filter framework and toolkit implementations to a tightly-coupled lidar-inertial navigation system taken from [15]. The overall system, shown in Fig. 9, consists of a solid-state lidar (Livox AVIA) with a built-in IMU and an onboard computer. The lidar provides a $10Hz$ scan rate and $200Hz$ gyro and accelerometer measurements. Unlike conventional spinning lidars (e.g., Velodyne lidars), the Livox AVIA has only 70° Field of View (FoV), making the lidar-inertial odometry rather challenging. The onboard computer is configured with a $1.8GHz$ quad-core Intel i7-8550U CPU and $8GB$ RAM. Besides the original state estimation problem considered in [15], we further consider the online estimation of the extrinsic between the lidar and IMU.

A. System modeling

The global frame is denoted as G (i.e. the initial frame of the IMU), the IMU frame is taken as the body frame (denoted as I), and the lidar frame is denoted as L. Assuming the lidar is rigidly attached to the IMU with an unknown extrinsic ${}^I\mathbf{T}_L = \left({}^I\mathbf{R}_L, {}^I\mathbf{p}_L\right)$, the objective of this system is to 1) estimate kinematics states of the IMU including its position (${}^G\mathbf{p}_I$), velocity (${}^G\mathbf{v}_I$), and rotation (${}^G\mathbf{R}_I \in SO(3)$) in the global frame; 2) estimate the biases of the IMU (i.e., $\mathbf{b_a}$ and $\mathbf{b_\omega}$; 3) estimate the gravity vector (${}^G\mathbf{g}$) in the global frame; 4) estimate the extrinsic ${}^I\mathbf{T}_L = \left({}^I\mathbf{R}_L, {}^I\mathbf{p}_L\right)$ online; and 5) build a global point cloud map of the observed environment.

Augmenting the state formulation in [15] with the lidar-IMU extrinsic, we have:

$$
\begin{aligned}
&{}^G\dot{\mathbf{p}}_I = {}^G\mathbf{v}_I,\ {}^G\dot{\mathbf{v}}_I = {}^G\mathbf{R}_I\left(\mathbf{a}_m - \mathbf{b_a} - \mathbf{n_a}\right) + {}^G\mathbf{g} \\
&{}^G\dot{\mathbf{R}}_I = {}^G\mathbf{R}_I\lfloor\boldsymbol{\omega}_m - \mathbf{b}_{\boldsymbol{\omega}} - \mathbf{n}_{\boldsymbol{\omega}}\rfloor,\ \dot{\mathbf{b}}_{\boldsymbol{\omega}} = \mathbf{n}_{\mathbf{b}_{\boldsymbol{\omega}}},\ \dot{\mathbf{b}}_{\mathbf{a}} = \mathbf{n}_{\mathbf{b_a}} \\
&{}^G\dot{\mathbf{g}} = \mathbf{0},\ {}^I\dot{\mathbf{R}}_L = \mathbf{0},\ {}^I\dot{\mathbf{p}}_L = \mathbf{0}
\end{aligned} \tag{52}
$$

where $\mathbf{a}_m, \boldsymbol{\omega}_m$ are the IMU measurements, $\mathbf{n_a}$, $\mathbf{n}_{\boldsymbol{\omega}}$ are IMU noises, $\mathbf{n}_{\mathbf{b}_{\boldsymbol{\omega}}}$ and $\mathbf{n}_{\mathbf{b_a}}$ are zero mean Gaussian white noises that drive the IMU biases $\mathbf{b}_{\boldsymbol{\omega}}$ and $\mathbf{b_a}$ respectively. The gravity vector ${}^G\mathbf{g}$ is of fixed length $9.81m/s^2$.

The measurement model is identical to [15]: for a new scan of lidar raw points, we extract the plane and edge points (i.e., feature points) based on the local curvature [36]. Then for a measured feature point ${}^L\mathbf{p}_{f_i}, i = 1, ..., m$, its true location in the global frame should lie on the corresponding plane (or edge) in the map built so far. More specifically, we represent the corresponding plane (or edge) in the map by its normal direction (or direction of the edge) $\mathbf{u}_i$ and a point ${}^G\mathbf{q}_i$ lying on the plane (or edge). Since the point ${}^L\mathbf{p}_{f_i}, i = 1, ..., m$ is measured in the lidar local frame (thus denoted as L) and contaminated by measurement noise $\mathbf{n}_i$, the true point location in the global frame is ${}^G\mathbf{T}_I{}^I\mathbf{T}_L\left({}^L\mathbf{p}_{f_i} - \mathbf{n}_i\right)$. Since this true location lies on the plane (or edge) defined by $\mathbf{u}_i$ and ${}^G\mathbf{q}_i$, its distance to the plane (or edge) should be zero, i.e.,

$$
\mathbf{G}_i\left({}^G\mathbf{T}_I{}^I\mathbf{T}_L\left({}^L\mathbf{p}_{f_i} - \mathbf{n}_i\right) - {}^G\mathbf{q}_i\right) = \mathbf{0},\ i = 1, \cdots, m \tag{53}
$$

where $\mathbf{G}_i = \mathbf{u}_i^T$ for a planar feature and $\mathbf{G}_i = \lfloor\mathbf{u}_i\rfloor$ for an edge feature. This equation defines an implicit measurement model which relates the measurement ${}^L\mathbf{p}_{f_i}$, measurement noise $\mathbf{n}_i$, and the ground-truth state ${}^G\mathbf{T}_I$ and ${}^I\mathbf{T}_L$.

To obtain $\mathbf{u}_i, {}^G\mathbf{q}_i$ of the corresponding plane (or edge) in the map, we use the state estimated at the current iteration to project the feature point ${}^L\mathbf{p}_{f_i}$ to the global frame and find the closest five feature points (of the same type) in the map built so far. After convergence of the iterated Kalman filter, the optimal state update is used to project the feature point ${}^L\mathbf{p}_{f_i}$ to the global frame and append it to the map. The updated map is finally used in the next to register new scans.

B. Canonical representation:

Using the zero-order hold discretization described in Section IV, the system with state model (52) and measurement model (53) can be discretized and cast into the canonical form as follows:

$$
\begin{aligned}
&\mathcal{S} = \mathbb{R}^3 \times \mathbb{R}^3 \times SO(3) \times \mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{S}^2 \times SO(3) \times \mathbb{R}^3, \\
&\mathcal{M} = \underbrace{\mathbb{R}^1 \times \cdots \times \mathbb{R}^1 \times \mathbb{R}^1 \times \cdots \times \mathbb{R}^1}_{m}, \\
&\mathbf{x}^T = \begin{bmatrix} {}^G\mathbf{p}_I & {}^G\mathbf{v}_I & {}^G\mathbf{R}_I & \mathbf{b_a} & \mathbf{b}_{\boldsymbol{\omega}} & {}^G\mathbf{g} & {}^I\mathbf{R}_L & {}^I\mathbf{p}_L \end{bmatrix}, \\
&\mathbf{u}^T = \begin{bmatrix} \mathbf{a}_m & \boldsymbol{\omega}_m \end{bmatrix} \\
&\mathbf{f}\left(\mathbf{x}, \mathbf{u}, \mathbf{w}\right)^T = \big[{}^G\mathbf{v}_I \quad {}^G\mathbf{R}_I(\mathbf{a}_m - \mathbf{b_a} - \mathbf{n_a}) + {}^G\mathbf{g} \\
&\qquad\qquad \boldsymbol{\omega}_m - \mathbf{b}_{\boldsymbol{\omega}} - \mathbf{n}_{\boldsymbol{\omega}} \quad \mathbf{n}_{\mathbf{b_a}} \quad \mathbf{n}_{\mathbf{b}_{\boldsymbol{\omega}}} \quad \mathbf{0} \quad \mathbf{0} \quad \mathbf{0} \big], \\
&\mathbf{h}_i(\mathbf{x}, \mathbf{v})^T = \mathbf{G}_i\left({}^G\mathbf{T}_I{}^I\mathbf{T}_L\left({}^L\mathbf{p}_{f_i} - \mathbf{n}_i\right) - {}^G\mathbf{q}_i\right), \\
&\mathbf{w}^T = \begin{bmatrix} \mathbf{n_a} & \mathbf{n}_{\boldsymbol{\omega}} & \mathbf{n}_{\mathbf{b_a}} & \mathbf{n}_{\mathbf{b}_{\boldsymbol{\omega}}} \end{bmatrix}, \\
&\mathbf{v}^T = \begin{bmatrix} \cdots & \mathbf{n}_i & \cdots \end{bmatrix}, i = 1, ..., m.
\end{aligned} \tag{54}
$$

with equivalent measurement $\mathbf{z}$ being constantly zero.

The system-specific partial differentions are therefore: partial differentiations for $\mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{w})$:

$$
\frac{\partial \mathbf{f}(\mathbf{x} \boxplus \delta\mathbf{x}, \mathbf{u}, \mathbf{0})}{\partial \delta\mathbf{x}}\Big|_{\delta\mathbf{x}=\mathbf{0}} = \begin{bmatrix}
\mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{U}_{23}^F & -{}^G\mathbf{R}_I & \mathbf{0} & \mathbf{U}_{26}^F & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0}
\end{bmatrix}
$$

$$
\frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{w})}{\partial \mathbf{w}}\Big|_{\mathbf{w}=\mathbf{0}} = \begin{bmatrix}
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
-{}^G\mathbf{R}_I & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & -\mathbf{I} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0}
\end{bmatrix} \tag{55}
$$

where $\mathbf{U}_{23}^F = -{}^G\mathbf{R}_I\lfloor\mathbf{a}_m - \mathbf{b_a}\rfloor$ and $\mathbf{U}_{26}^F = -\lfloor{}^G\mathbf{g}\rfloor\mathbf{B}({}^G\mathbf{g})$, $\mathbf{B}(\cdot)$ is defined in the equation (13). And partial differentiations for $\mathbf{h}(\mathbf{x}, \mathbf{v})$:

$$
\begin{aligned}
&\frac{\partial(\mathbf{h}(\mathbf{x} \boxplus \delta\mathbf{x}, \mathbf{0}) \boxminus \mathbf{h}(\mathbf{x}, \mathbf{0}))}{\partial \delta\mathbf{x}}\Big|_{\delta\mathbf{x}=\mathbf{0}} = \\
&\begin{bmatrix}
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\mathbf{G}_i & \mathbf{0} & \mathbf{U}_{i3}^H & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{U}_{i7}^H & \mathbf{G}_i{}^G\mathbf{R}_I \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots
\end{bmatrix}, \\
&\frac{\partial(\mathbf{h}(\mathbf{x}, \mathbf{v}) \boxminus \mathbf{h}(\mathbf{x}, \mathbf{0}))}{\partial \mathbf{v}}\Big|_{\mathbf{v}=\mathbf{0}} = \mathrm{diag}(\cdots, -\mathbf{G}_i{}^G\mathbf{R}_I{}^I\mathbf{R}_L, \cdots)
\end{aligned} \tag{56}
$$

where $\mathbf{U}_{i3}^H = -\mathbf{G}_i{}^G\mathbf{R}_I\lfloor{}^I\mathbf{T}_L{}^L\mathbf{p}_{f_i}\rfloor$, and $\mathbf{U}_{i7}^H = -\mathbf{G}_i{}^G\mathbf{R}_I{}^I\mathbf{R}_L\lfloor{}^L\mathbf{p}_{f_i}\rfloor$.

Supplying the canonical representation of the system (54) and the respective partial differentiations in (55) and (56) to our toolkit leads to a tightly-coupled lidar-inertial navigation system.

C. Experiment results

We verify the tightly-coupled lidar-inertial navigation system implemented by our toolkit in three different scenarios, i.e., indoor UAV flight, indoor quick-shake experiment, and outdoor random walk experiment. They are denoted as V1, V2, and V3 respectively. For each scenario, we test the implementation on two trials of data, one collected by ourselves and the other from the original paper [15]. The six datasets are denoted as V1-01, V1-02, V2-01, V2-02, V3-01, and V3-02, respectively. In all experiments, the maximal number of iteration in the iterated Kalman filter (see Algorithm 1) is set to 4, i.e., $N_{\max} = 4$.

1) Indoor UAV flight: For the UAV fight experiment, we only show the data collected in this work (i.e., V1-01). The experiment is conducted in an indoor environment (see Fig. 10 (A)) where the UAV took off from the ground and flied in a circle path. During the path following, the UAV is constantly facing at a cluttered office area behind a safety net (see Fig.

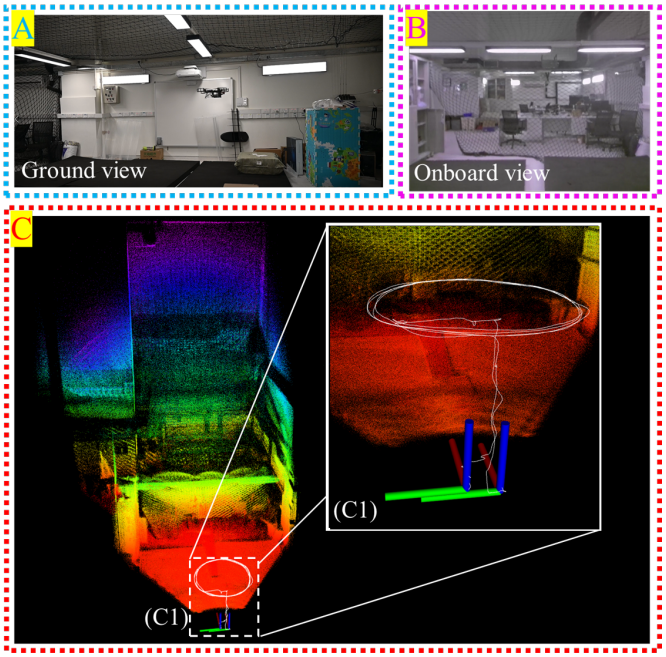

Fig. 10. Real-time mapping result of dataset V1-01, an indoor UAV flight experiment of lidar-inertial navigation system. A: Photo from ground view; B: Snapshot of onboard FPV video; C: Map result, (C1) Trajectory and poses of the UAV at beginning and end of the experiment.

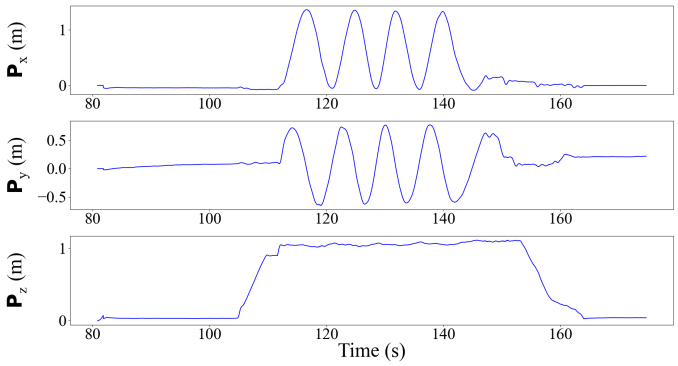

Fig. 11. Estimation of the position of the UAV for dataset V1-01, an indoor UAV flight experiment of lidar-inertial navigation system.

10 (B)). After the path following, a human pilot took over the UAV and landed it manually to ensure that the landing position coincides with the take-off point.

Fig. 10 (C) shows the real-time mapping results overlaid with the 3D trajectory estimated by our system. It can be seen that our system achieves consistent mapping even in the cluttered indoor environment. The position drift is less than 0.9692% (i.e., $0.2211m$ drift over the $22.81m$ path, see Fig. 10 (C1)). This drift is caused, in part, by the accumulation of odometry error, which is common in SLAM systems, and in part by inaccurate manual landing.

We show the estimated trajectory of position (${}^G\mathbf{p}_I$), rotation (${}^G\mathbf{R}_I$), and velocity (${}^G\mathbf{v}_I$) in Fig. 11, Fig. 12 and Fig. 13, respectively, where the experiment starts from $80.8393s$ and ends at $174.6590s$. Our system achieves smooth state estimation that is suitable for onboard feedback control. All the estimated state variables agree well with the actual motions.

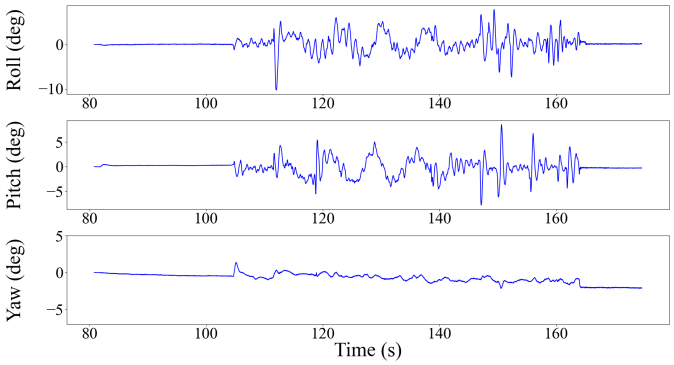

Fig. 12. Estimation of the rotation of the UAV for dataset V1-01, an indoor UAV flight experiment of lidar-inertial navigation system.

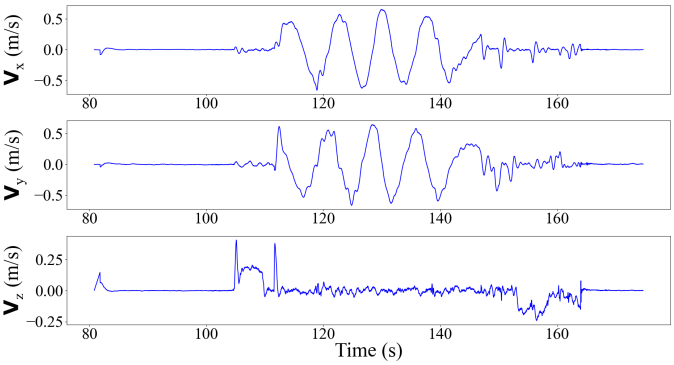

Fig. 13. Estimation of the velocity of the UAV for dataset V1-01, an indoor UAV flight experiment of lidar-inertial navigation system.

The V1-02 dataset has similar performance (0.6872% position drift) and the results are not included in this paper due to space limit. For further experiment demonstration, we refer readers to the videos at https://youtu.be/sz_ZlDkl6fA.

2) Indoor quick shake: The second experiment is conducted in a cluttered office area (see Fig. 14 (A)). In the experiment, the UAV containing the lidar sensor and onboard computer is handheld (see Fig. 14 (B)) and quickly shaken, creating a large rotation up to $356.85deg/s$ (see onboard FPV images from Fig. 14 (A) and raw IMU measurements in Fig. 15). The UAV ends at the starting position to enable the computation of odometry drift.

Fig. 14 (C) shows the real-time mapping result on dataset V2-01. It is seen that our system achieves consistent mapping even in fast rotational movements that are usually challenging for visual-inertial odometry due to image defocus and/or motion blur (see Fig. 14 (A4) and (A5)). As shown in Fig. 14 (C3), the estimated final position of the UAV coincides with the beginning position, leading to a position drift less than 0.1113% (i.e., $0.1232m$ drift over $110.64m$ path).

Fig. 16, Fig. 17 and Fig. 18 show the estimates of the position (${}^G\mathbf{p}_I$), Euler angles of the rotation (${}^G\mathbf{R}_I$) and velocity (${}^G\mathbf{V}_I$) of the UAV, where the experiment starts from $80.5993s$ and ends at $303.499s$. Those estimates are changing in a high frequency, which is consistent with the actual motions of the UAV. The noticeable translation around $275s$ is the actual UAV motion. We refer readers to the videos at https:

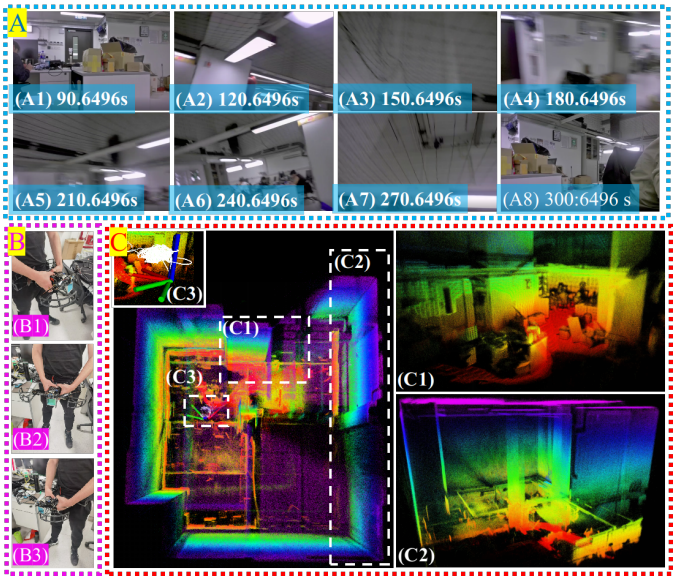

Fig. 14. Real-time mapping result of dataset V2-01, an indoor quick-shake experiment of lidar-inertial navigation system. A: Snapshots of onboard FPV video; B: The UAV containing the lidar sensor and onboard computer is handheld to creat large rotations; C: Map result, (C1) Local zoom-in map, (C2) Side view of the map, (C3) Poses of the UAV at the beginning and end of the experiment.

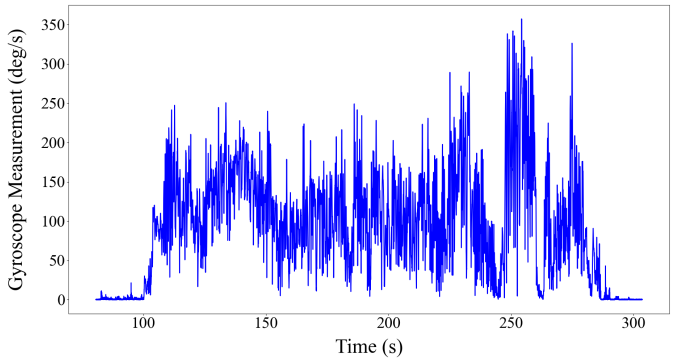

Fig. 15. Magnitude of gyroscope measurements for dataset V2-01, an indoor quick-shake experiment of lidar-inertial navigation system.

//youtu.be/sz_ZlDkl6fA for further experiment demonstration.

3) Outdoor random walk: The third experiment is conducted in a structured outdoor environment which is a corridor between a slope and the Hawking Wong building of the University of Hong Kong. In the experiment, the UAV is handheld to move along the road and then return to the beginning position (see Fig. 19 (A)).

The real-time mapping results of dataset V3-01 estimated by our toolkit is shown in Fig. 19 (B), which clearly shows the building on one side and the cars and bricks on the slope. The position drift is less than 0.0003538% (i.e., $0.0007260m$ drift over $205.22m$ path, see Fig. 19 (B3)). This extremely small drift, although seemly supports the efficacy of our system, should not be interpreted as the ground true drift since the actual landing cannot be made this accurate in practice. A more indicative result is obtained from V3-02, which leads to a position drift of 0.1575%. The rest results of V3-02 is very similar to V3-01, hence are omitted in this paper.

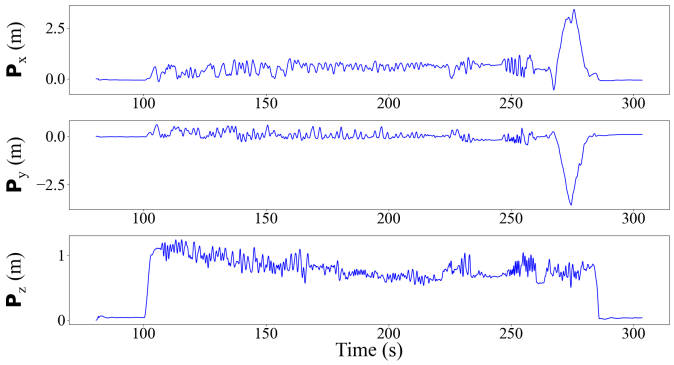

Fig. 16. Estimation of position on dataset V2-01, an indoor quick-shake experiment of lidar-inertial navigation system.

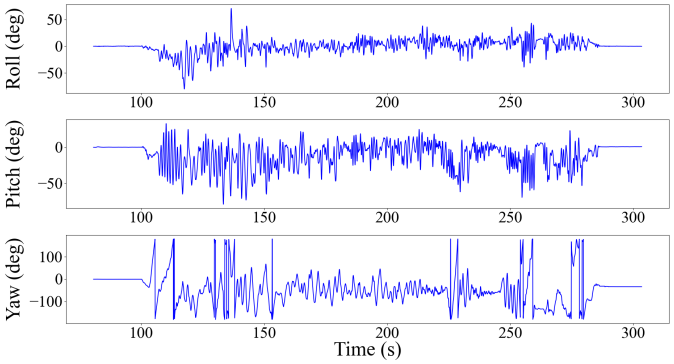

Fig. 17. Estimation of rotation on dataset V2-01, an indoor quick-shake experiment of lidar-inertial navigation system.

The estimations of the kinematics parameters are shown in Fig. 20, Fig. 21 and Fig. 22, where the experiment starts from $353.000s$ and ends at $509.999s$. The trajectory is approximately symmetric about the middle time in X and Z direction, which agrees with the actual motion profile where the sensor is moved back on the same road. For further experiment demonstration, we refer readers to the videos at https://youtu.be/sz_ZlDkl6fA.

4) Online estimation of extrinsic, gravity, and IMU bias: To verify our developed method being a properly functioning filter, the online calibration parameters, which are composed of gravity in the global frame, IMU biases and the lidar-IMU

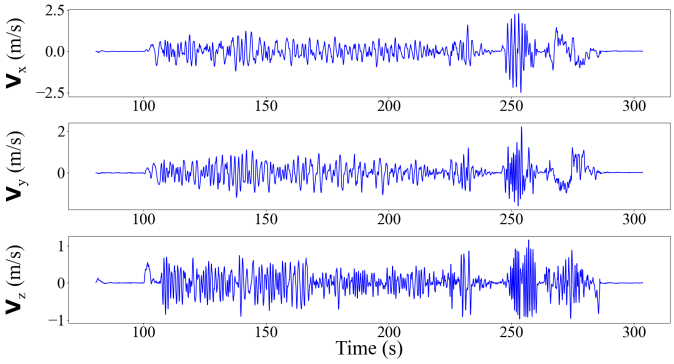

Fig. 18. Estimation of velocity on dataset V2-01, an indoor quick-shake experiment of lidar-inertial navigation system.

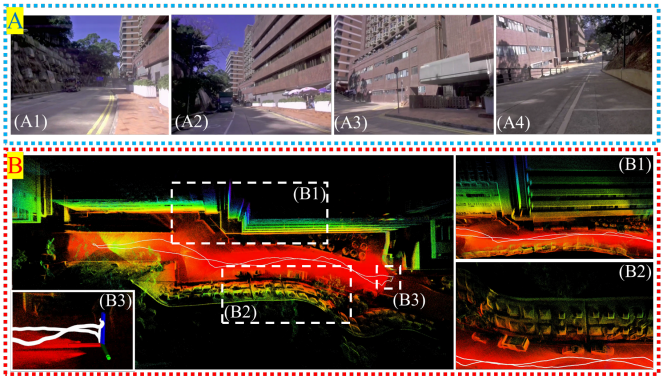

Fig. 19. Real-time mapping result of dataset V3-01, an outdoor random walk experiment of lidar-inertial navigation system. A: Photos of the environment of this experiment; B: Mapping result, (B1) Local zoom-in map result of one side of the road, (B2) Local zoom-in map result of the other side of the road, (B3) Poses of the UAV at the beginning and end of the experiment.

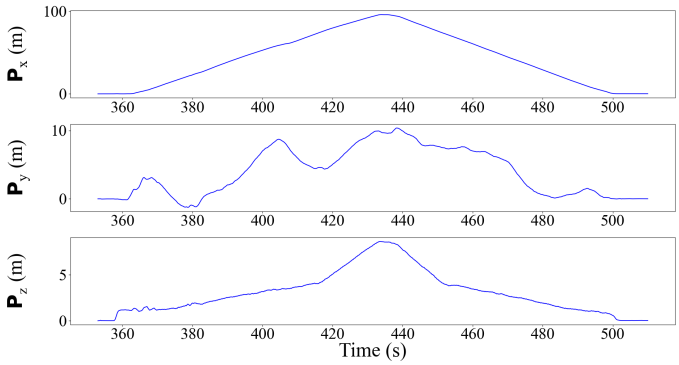

Fig. 20. Estimation of position on dataset V3-01, an outdoor random walk experiment of lidar-inertial navigation system.

extrinsics have to converge. Moreover, the extrinsic estimate should be close across different datasets with the same sensor setup, and we can thus evaluate the extrinsics on multiple datasets and compare the values they have converged to. Fig. 23 shows the final estimate of the rotational and translational parts of the extrinsics by running the proposed toolkit on all the six datasets. The initial values of the extrinsics were read from the manufacturer datasheet. As seen in Fig. 23, the extrinsic estimates (both rotation and translation) over

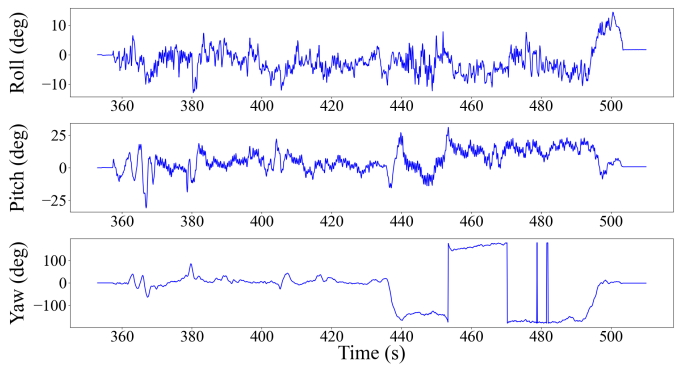

Fig. 21. Estimation of rotation on dataset V3-01, an outdoor random walk experiment of lidar-inertial navigation system.

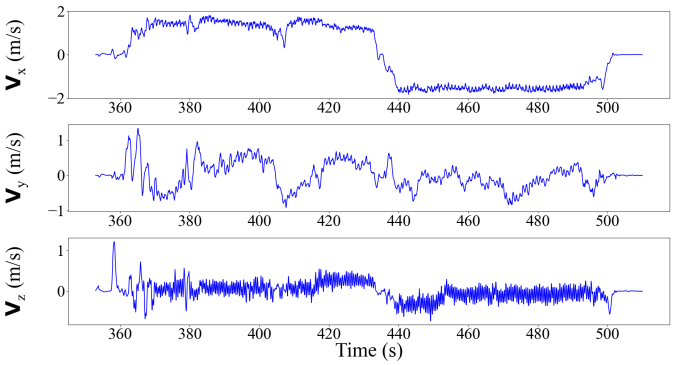

Fig. 22. Estimation of velocity on dataset V3-01, an outdoor random walk experiment of lidar-inertial navigation system.

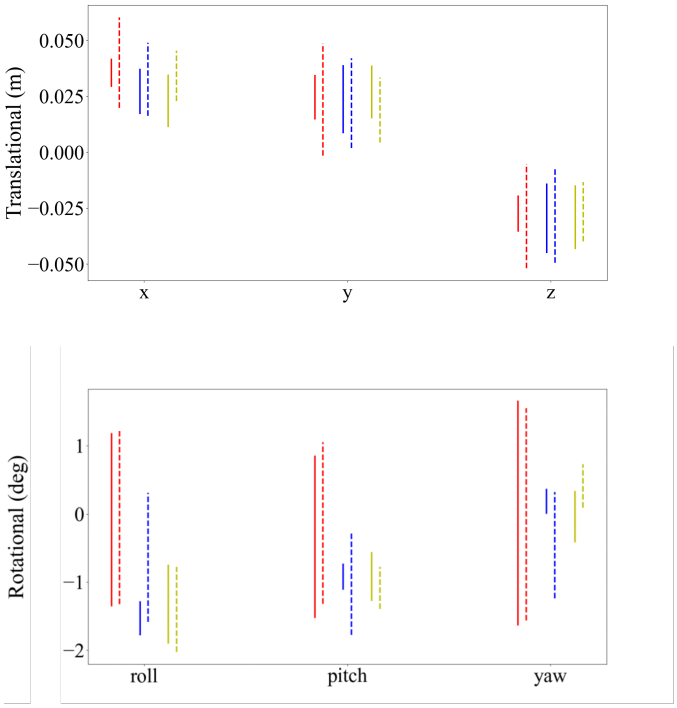

Fig. 23. Estimation of the extrinsics between lidar and IMU for 6 datasets of the lidar-inertial system. The length of each line corresponds to the 3σ bounds the Kalman filter converged to. Red, blue, and yellow lines indicate datasets V1 (i.e., indoor UAV flight), V2 (i.e., indoor quick-shake), and V3 (i.e., outdoor random walk), respectively. For each dataset, the solid and the dashed lines respectively indicate the trial 01 (i.e., data collected in this work) and trial 02 (i.e., data from [15]).

different dataset show great agreement. The uncertainty in translation is $1cm - 2cm$ while that in rotation is less than 1°. In particular, the variance of the rotational extrinsic on dataset V1 is significantly larger than the others. This is because the slow and smooth movement in the flight experiment, which creates insufficient excitation in parameter estimation. On the other hand, the motion profile of the two handheld experiments V2 and V3 has much more excitation as shown previously. The other possible reason is the enlarged IMU noises due to the constant propeller rotation in UAV flight. Moreover, as indicated by the blue lines in Fig. 23, we notice a larger variance in V2-02 than V2-01. This is resulted from the fact

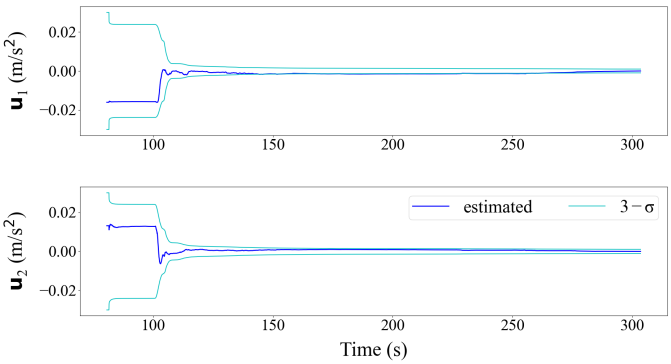

Fig. 24. Gravity estimation error (dark blue lines) and its 3σ bounds (bright green lines) on dataset V2-01.

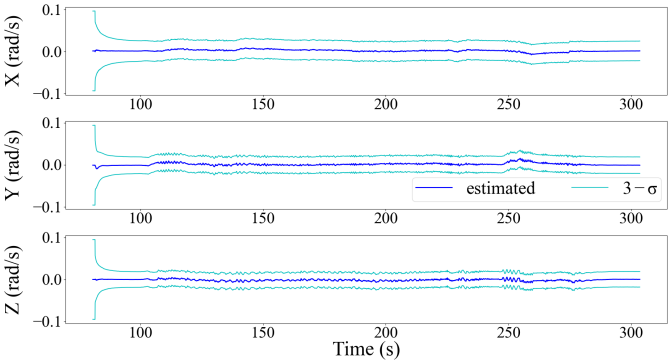

Fig. 25. Estimated gyroscope biases on dataset V2-01. Estimates (dark blue lines) together with the 3σ bounds (bright green lines) are depicted. The estimates converge very quickly due to the large rotational movement.

that V2-01 has constant excitation over $222.85s$ while V2-02 only ran for $48.001s$ where the Kalman filter has not fully converged (e.g., see Fig. 24).

We further inspect the convergence of the gravity estimation. Due to the space limit, we show the result on dataset V2-01 only. Fig. 24 shows the gravity estimation error $\mathbf{u} = {}^G\bar{\mathbf{g}} \boxminus {}^G\hat{\mathbf{g}}_k \in \mathbb{R}^2$, where ${}^G\bar{\mathbf{g}}$ is the ground-true gravity vector and ${}^G\hat{\mathbf{g}}_k$ is the estimate at step k. Since the ground-true gravity vector is unknown, we use the converged gravity estimation as ${}^G\bar{\mathbf{g}}$. Fig. 24 further shows the 3σ bounds for $\mathbf{u}$ and is estimated by the Kalman filter. It is shown that the error constantly falls within the 3σ bounds, which indicates the consistency of the Kalman filter.

Finally, we investigate the convergence of the IMU bias estimation. We show the results on dataset V2-01 only. The results are depicted in Fig. 25 and Fig. 26, where the estimates over time are plotted together with the 3σ bounds. In particular, the gyroscope biases converge rapidly due to the large rotational movement. Also the accelerometer biases converge with sufficient excitation of the system. They typically converge faster along the gravity direction due to the large vertical movement at the beginning of the dataset (see Fig. 14).

5) Running time: To further evaluate the practicability of the developed toolkit, its running time on the three datasets V1-02, V2-02, and V3-02 are evaluated and compared against [15]. Note that the work in [15] also used an iterated Kalman filter but differs with our implementations in two

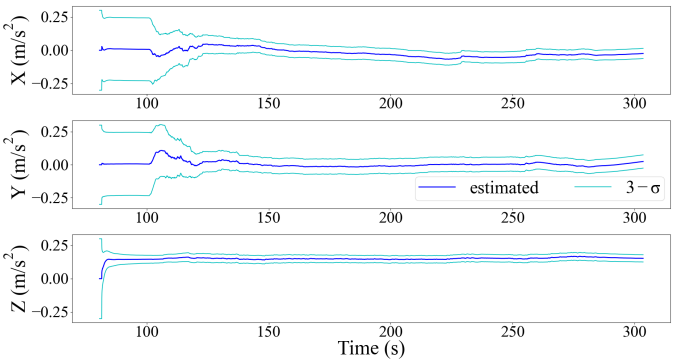

Fig. 26. Estimated accelerometer biases on dataset V2-01. Estimates (dark blue lines) together with the 3σ bounds (bright green lines) are depicted. The accelerometer bias converges quicker along the gravity direction which is mostly along the z-axis.

aspects: (1) The iterated Kalaman filter in [15] is manually derived and the respective matrices (e.g., $\mathbf{F}_{\mathbf{x}_\tau}, \mathbf{F}_{\mathbf{w}_\tau}$ in (35)) used in the Kalman filter are directly coded. Matrices sparsity are carefully exploited for computation efficiency. In contrast, our implementation directly uses the toolkit which separates the computation of manifold-specific parts and system-specific parts; (2) The original work in [15] did not consider the estimate of extrinsic between lidar and IMU, hence has six fewer state variables. Other than these two aspects, the rest implementations are identical. Both implementations are tested on the UAV onboard computer (see Fig. 9).

The running time comparison is shown in Table. III, which shows the average time for completing one step of Kalman filter (both propagation and update). As expected, the toolkit-based implementation takes more computation time due to the higher state dimension and the toolkit overhead. However, this time overhead is acceptable and both implementations run sufficiently fast in real-time.

TABLE III
COMPARISON OF THE AVERAGE RUNNING TIME

	IKFoM-based implementation	Hand-derived implementation in [15]
V1-02:	$7.586ms$	$5.166ms$
V2-02:	$43.22ms$	$33.16ms$
V3-02:	$53.70ms$	$44.40ms$

[1] Running time is recorded as the time consumed by one propagation step and one update step.

VIII. CONCLUSION

This paper proposed a canonical representation of robot systems and developed a symbolic error-state iterated Kalman filter. The canonical representation employs manifolds to represent the system states and uses $\boxplus\backslash\boxminus$ and $\oplus$ operations to describe the system model. Based on the canonical representation of a robotic system, we showed the separation principle between the manifold-specific descriptions and the system-specific descriptions in a Kalman filter framework. This separation enables us to encapsulate manifold structures

into Kalman filters by developing a C++ toolkit, facilitating the quick deployment of Kalman filters to generic robotic systems operating on manifolds. The proposed method and the developed toolkit are verified on a tightly-coupled lidar-inertial navigation system in three different scenarios.

APPENDIX

A. Partial differentiation of $SO(3)$

Lemma 1. *If $\mathbf{x}, \mathbf{y} \in SO(3)$, $\mathbf{u}, \mathbf{v} \in \mathbb{R}^3$, then*

$$\frac{\partial(((\mathbf{x}\boxplus\mathbf{u})\oplus\mathbf{v})\boxminus\mathbf{y})}{\partial\mathbf{u}}=\mathbf{A}(((\mathbf{x}\boxplus\mathbf{u})\oplus\mathbf{v})\boxminus\mathbf{y})^{-T}\operatorname{Exp}(-\mathbf{v})\,\mathbf{A}(\mathbf{u})^{\mathrm{T}}$$

where the operations $\boxplus$, $\boxminus$, and $\oplus$ are defined in (6) and $\mathbf{A}(\cdot)$ is defined in (7).

Proof. Denote $\mathbf{w}=((\mathbf{x}\boxplus\mathbf{u})\oplus\mathbf{v})\boxminus\mathbf{y}$, we have

$$\operatorname{Exp}(\mathbf{w}) = \mathbf{y}^{-1}\cdot\mathbf{x}\cdot\operatorname{Exp}(\mathbf{u})\cdot\operatorname{Exp}(\mathbf{v})$$

Hence a small variation $\Delta\mathbf{u}$ in $\mathbf{u}$ causes a small variation $\Delta\mathbf{w}$ in $\mathbf{w}$, which is subject to

$$\operatorname{Exp}(\mathbf{w}+\Delta\mathbf{w}) = \mathbf{y}^{-1}\cdot\mathbf{x}\cdot\operatorname{Exp}(\mathbf{u}+\Delta\mathbf{u})\cdot\operatorname{Exp}(\mathbf{v}) \quad (57)$$

Using the fact $\operatorname{Exp}(\mathbf{u}+\Delta\mathbf{u}) = \operatorname{Exp}(\mathbf{u})\cdot\left(\mathbf{I}+\lfloor\mathbf{A}(\mathbf{u})^{\mathrm{T}}\Delta\mathbf{u}\rfloor\right)$ as shown in [33], it is derived that the left hand side of (57)

$$\operatorname{Exp}(\mathbf{w}+\Delta\mathbf{w}) = \operatorname{Exp}(\mathbf{w})\cdot\left(\mathbf{I}+\lfloor\mathbf{A}(\mathbf{w})^{\mathrm{T}}\Delta\mathbf{w}\rfloor\right)$$

and the right hand side of (57)

$$\begin{aligned}&\mathbf{y}^{-1}\cdot\mathbf{x}\cdot\operatorname{Exp}(\mathbf{u}+\Delta\mathbf{u})\cdot\operatorname{Exp}(\mathbf{v})\\ &=\mathbf{y}^{-1}\cdot\mathbf{x}\cdot\operatorname{Exp}(\mathbf{u})\cdot\left(\mathbf{I}+\lfloor\mathbf{A}(\mathbf{u})^{\mathrm{T}}\Delta\mathbf{u}\rfloor\right)\cdot\operatorname{Exp}(\mathbf{v})\\ &=\operatorname{Exp}(\mathbf{w})\operatorname{Exp}(-\mathbf{v})\cdot\left(\mathbf{I}+\lfloor\mathbf{A}(\mathbf{u})^{\mathrm{T}}\Delta\mathbf{u}\rfloor\right)\cdot\operatorname{Exp}(\mathbf{v})\end{aligned}$$

Equating the two sides of (57) leads to

$$\mathbf{A}(\mathbf{w})^{T}\Delta\mathbf{w} = \operatorname{Exp}(-\mathbf{v})\cdot\mathbf{A}(\mathbf{u})^{\mathrm{T}}\Delta\mathbf{u}$$

and as a result,

$$\frac{\partial(((\mathbf{x}\boxplus\mathbf{u})\oplus\mathbf{v})\boxminus\mathbf{y})}{\partial\mathbf{u}} = \frac{\Delta\mathbf{w}}{\Delta\mathbf{u}} = \mathbf{A}(\mathbf{w})^{-T}\operatorname{Exp}(-\mathbf{v})\,\mathbf{A}(\mathbf{u})^{\mathrm{T}}$$

■

B. Partial differentiation of compound manifolds

Lemma 2. *If $\mathbf{x}_1, \mathbf{y}_1 \in \mathcal{S}_1$; $\mathbf{x}_2, \mathbf{y}_2 \in \mathcal{S}_2$; $\mathbf{u}_1, \mathbf{v}_1 \in \mathbb{R}^{n_1}$ and $\mathbf{u}_2, \mathbf{v}_2 \in \mathbb{R}^{n_2}$; where n_1, n_2 are dimensions of $\mathcal{S}_1$, $\mathcal{S}_2$ respectively, define compound manifold $\mathcal{S} = \mathcal{S}_1 \times \mathcal{S}_2$, and its elements $\mathbf{x} = \begin{bmatrix}\mathbf{x}_1 & \mathbf{x}_2\end{bmatrix}^T \in \mathcal{S}$; $\mathbf{y} = \begin{bmatrix}\mathbf{y}_1 & \mathbf{y}_2\end{bmatrix}^T \in \mathcal{S}$; $\mathbf{u} = \begin{bmatrix}\mathbf{u}_1 & \mathbf{u}_2\end{bmatrix}^T \in \mathbb{R}^{n_1+n_2}$ and $\mathbf{v} = \begin{bmatrix}\mathbf{v}_1 & \mathbf{v}_2\end{bmatrix}^T \in \mathbb{R}^{l_1+l_2}$, where $l_1 = n_1$ when $\mathcal{S}_1$ is a Lie group and $l_2 = n_2$ when $\mathcal{S}_2$ is a Lie group, then*

$$\frac{\partial(((\mathbf{x}\boxplus_{\mathcal{S}}\mathbf{u})\oplus_{\mathcal{S}}\mathbf{v})\boxminus_{\mathcal{S}}\mathbf{y})}{\partial\mathbf{u}} = \begin{bmatrix}\frac{\partial(((\mathbf{x}_1\boxplus_{\mathcal{S}_1}\mathbf{u}_1)\oplus_{\mathcal{S}_1}\mathbf{v}_1)\boxminus_{\mathcal{S}_1}\mathbf{y}_1)}{\partial\mathbf{u}_1} & \mathbf{0}\\ \mathbf{0} & \frac{\partial(((\mathbf{x}_2\boxplus_{\mathcal{S}_2}\mathbf{u}_2)\oplus_{\mathcal{S}_2}\mathbf{v}_2)\boxminus_{\mathcal{S}_2}\mathbf{y}_2)}{\partial\mathbf{u}_2}\end{bmatrix}$$

Proof. Define $\mathbf{w} = ((\mathbf{x}\boxplus_{\mathcal{S}}\mathbf{u})\oplus_{\mathcal{S}}\mathbf{v})\boxminus_{\mathcal{S}}\mathbf{y}$, then according to the composition of operation $\boxplus$, $\boxminus$ and $\oplus$ in (14), we have

$$\begin{aligned}\mathbf{w}&=((\mathbf{x}\boxplus_{\mathcal{S}}\mathbf{u})\oplus_{\mathcal{S}}\mathbf{v})\boxminus_{\mathcal{S}}\mathbf{y}\\ &=\left(\left(\begin{bmatrix}\mathbf{x}_1\\ \mathbf{x}_2\end{bmatrix}\boxplus_{\mathcal{S}}\begin{bmatrix}\mathbf{u}_1\\ \mathbf{u}_2\end{bmatrix}\right)\oplus_{\mathcal{S}}\mathbf{v}\right)\boxminus_{\mathcal{S}}\mathbf{y}\\ &=\left(\begin{bmatrix}\mathbf{x}_1\boxplus_{\mathcal{S}_1}\mathbf{u}_1\\ \mathbf{x}_2\boxplus_{\mathcal{S}_2}\mathbf{u}_2\end{bmatrix}\oplus_{\mathcal{S}}\begin{bmatrix}\mathbf{v}_1\\ \mathbf{v}_2\end{bmatrix}\right)\boxminus_{\mathcal{S}}\mathbf{y}\\ &=\begin{bmatrix}(\mathbf{x}_1\boxplus_{\mathcal{S}_1}\mathbf{u}_1)\oplus_{\mathcal{S}_1}\mathbf{v}_1\\ (\mathbf{x}_2\boxplus_{\mathcal{S}_2}\mathbf{u}_2)\oplus_{\mathcal{S}_2}\mathbf{v}_2\end{bmatrix}\boxminus_{\mathcal{S}}\begin{bmatrix}\mathbf{y}_1\\ \mathbf{y}_2\end{bmatrix}\\ &=\begin{bmatrix}((\mathbf{x}_1\boxplus_{\mathcal{S}_1}\mathbf{u}_1)\oplus_{\mathcal{S}_1}\mathbf{v}_1)\boxminus_{\mathcal{S}_1}\mathbf{y}_1\\ ((\mathbf{x}_2\boxplus_{\mathcal{S}_2}\mathbf{u}_2)\oplus_{\mathcal{S}_2}\mathbf{v}_2)\boxminus_{\mathcal{S}_2}\mathbf{y}_2\end{bmatrix}\triangleq\begin{bmatrix}\mathbf{w}_1\\ \mathbf{w}_2\end{bmatrix}\end{aligned}$$

As a result, the differentiation is

$$\begin{aligned}\frac{\partial\mathbf{w}}{\partial\mathbf{u}}&=\begin{bmatrix}\frac{\partial\mathbf{w}_1}{\partial\mathbf{u}_1} & \frac{\partial\mathbf{w}_1}{\partial\mathbf{u}_2}\\ \frac{\partial\mathbf{w}_2}{\partial\mathbf{u}_1} & \frac{\partial\mathbf{w}_2}{\partial\mathbf{u}_2}\end{bmatrix}=\begin{bmatrix}\frac{\partial\mathbf{w}_1}{\partial\mathbf{u}_1} & \mathbf{0}\\ \mathbf{0} & \frac{\partial\mathbf{w}_2}{\partial\mathbf{u}_2}\end{bmatrix}\\ &=\begin{bmatrix}\frac{\partial(((\mathbf{x}_1\boxplus_{\mathcal{S}_1}\mathbf{u}_1)\oplus_{\mathcal{S}_1}\mathbf{v}_1)\boxminus_{\mathcal{S}_1}\mathbf{y}_1)}{\partial\mathbf{u}_1} & \mathbf{0}\\ \mathbf{0} & \frac{\partial(((\mathbf{x}_2\boxplus_{\mathcal{S}_2}\mathbf{u}_2)\oplus_{\mathcal{S}_2}\mathbf{v}_2)\boxminus_{\mathcal{S}_2}\mathbf{y}_2)}{\partial\mathbf{u}_2}\end{bmatrix}\end{aligned}$$

■

REFERENCES

[1] F. L. Markley, "Attitude error representations for kalman filtering," *Journal of guidance, control, and dynamics*, vol. 26, no. 2, pp. 311–317, 2003.

[2] N. Trawny and S. I. Roumeliotis, "Indirect kalman filter for 3d attitude estimation," *University of Minnesota, Dept. of Comp. Sci. & Eng., Tech. Rep*, vol. 2, p. 2005, 2005.

[3] F. L. Markley and J. L. Crassidis, *Fundamentals of spacecraft attitude determination and control*. Springer, 2014, vol. 33.

[4] F. M. Mirzaei and S. I. Roumeliotis, "A kalman filter-based algorithm for imu-camera calibration: Observability analysis and performance evaluation," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. p.1143–1156, 2008.

[5] J. Kelly and G. S. Sukhatme, "Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration," *The International Journal of Robotics Research*, vol. 30, no. 1, pp. 56–79, 2011.

[6] J. Sola, "Quaternion kinematics for the error-state kalman filter," *arXiv preprint arXiv:1711.02508*, 2017.

[7] M. Kleinert and S. Schleith, "Inertial aided monocular slam for gps-denied navigation," in *2010 IEEE Conference on Multisensor Fusion and Integration*, 2010, pp. 20–25.

[8] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint kalman filter for vision-aided inertial navigation," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, 2007, pp. 3565–3572.

[9] M. Li and A. Mourikis, "High-precision, consistent ekf-based visual–inertial odometry," *The International Journal of Robotics Research*, vol. 32, pp. 690–711, 05 2013.

[10] S. Lynen, M. W. Achtelik, S. Weiss, M. Chli, and R. Siegwart, "A robust and modular multi-sensor fusion approach applied to mav navigation," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 3923–3929.

[11] M. Bloesch, M. Burri, S. Omari, M. Hutter, and R. Siegwart, "Iterated extended kalman filter based visual-inertial odometry using direct photometric feedback," *The International Journal of Robotics Research*, vol. 36, no. 10, pp. 1053–1072, 2017.

[12] Z. Huai and G. Huang, "Robocentric visual-inertial odometry," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 6319–6326.

[13] J. A. Hesch, F. M. Mirzaei, G. L. Mariottini, and S. I. Roumeliotis, "A laser-aided inertial navigation system (l-ins) for human localization in unknown indoor environments," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 5376–5382.

[14] C. Qin, H. Ye, C. E. Pranata, J. Han, S. Zhang, and M. Liu, "Lins: A lidar-inertial state estimator for robust and efficient navigation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 8899–8906.

[15] W. Xu and F. Zhang, "Fast-lio: A fast, robust lidar-inertial odometry package by tightly-coupled iterated kalman filter," *arXiv preprint arXiv:2010.08196*, 2020.

[16] G. Lu and F. Zhang, "Imu-based attitude estimation in the presence of narrow-band noise," *IEEE / ASME Transactions on Mechatronics*, vol. 24, no. 2, pp. 841–852, 2019.

[17] J. Lin, X. Liu, and F. Zhang, "A decentralized framework for simultaneous calibration, localization and mapping with multiple lidars," 2020.

[18] G. P. Huang, A. I. Mourikis, and S. I. Roumeliotis, "A quadratic-complexity observability-constrained unscented kalman filter for slam," *IEEE Transactions on Robotics*, vol. 29, no. 5, pp. 1226–1243, 2013.

[19] S. Huang and G. Dissanayake, "Convergence and consistency analysis for extended kalman filter based slam," *IEEE Transactions on robotics*, vol. 23, no. 5, pp. 1036–1049, 2007.

[20] M. S. Grewal, L. R. Weill, and A. P. Andrews, *Global positioning systems, inertial navigation, and integration*. John Wiley & Sons, 2007.

[21] K. M. Lynch and F. C. Park, *Modern Robotics*. Cambridge University Press, 2017.

[22] A. R. Klumpp, "Apollo lunar descent guidance," *Automatica*, vol. 10, no. 2, pp. 133 – 146, 1974. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0005109874900193

[23] A. J. Davison, "Real-time simultaneous localisation and mapping with a single camera," in *null*. IEEE, 2003, p. 1403.

[24] C. Hertzberg, R. Wagner, U. Frese, and L. Schröder, "Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds," *Information Fusion*, vol. 14, no. 1, pp. 57–77, 2013.

[25] C. Hertzberg, "A framework for sparse, non-linear least squares problems on manifolds," 2008.

[26] V. Wüest, V. Kumar, and G. Loianno, "Online estimation of geometric and inertia parameters for multirotor aerial vehicles," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 1884–1890.

[27] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g 2 o: A general framework for graph optimization," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 3607–3613.

[28] S. Agarwal, K. Mierle, and Others, "Ceres solver," http://ceres-solver.org.

[29] R. Wagner, O. Birbach, and U. Frese, "Rapid development of manifold-based graph optimization systems for multi-sensor calibration and slam," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 3305–3312.

[30] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based slam," *IEEE Intelligent Transportation Systems Magazine*, vol. 2, no. 4, pp. 31–43, 2010.

[31] M. Burri, M. Bloesch, Z. Taylor, R. Siegwart, and J. Nieto, "A framework for maximum likelihood parameter identification applied on mavs," *Journal of Field Robotics*, vol. 35, no. 1, pp. 5–22, 2018. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21729

[32] R. M. Murray, Z. Li, S. S. Sastry, and S. S. Sastry, *A mathematical introduction to robotic manipulation*. CRC press, 1994.

[33] F. Bullo and R. M. Murray, "Proportional derivative (pd) control on the euclidean group," in *European control conference*, vol. 2, 1995, pp. 1091–1097.

[34] C. Hertzberg, R. Wagner, U. Frese, and L. Schröder, "Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds," *CoRR*, vol. abs/1107.1119, 2011. [Online]. Available: http://arxiv.org/abs/1107.1119

[35] B. M. Bell and F. W. Cathey, "The iterated kalman filter update as a gauss-newton method," *IEEE Transactions on Automatic Control*, vol. 38, no. 2, pp. 294–297, 1993.

[36] J. Lin and F. Zhang, "Loam livox: A fast, robust, high-precision lidar odometry and mapping package for lidars of small fov," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 3126–3131.