HAL
open science

A Code for Unscented Kalman Filtering on Manifolds (UKF-M)

Martin Brossard, Axel Barrau, Silvere Bonnabel

To cite this version:

Martin Brossard, Axel Barrau, Silvere Bonnabel. A Code for Unscented Kalman Filtering on Manifolds (UKF-M). International Conference on Robotics and Automation (ICRA), May 2020, Paris, France. hal-02463553v1

HAL Id: hal-02463553

https://hal.science/hal-02463553v1

Submitted on 1 Feb 2020 (v1), last revised 10 Mar 2020 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

A Code for Unscented Kalman Filtering on Manifolds (UKF-M)

Martin BROSSARD†, Axel BARRAU*, Silvère BONNABEL†
†MINES ParisTech, PSL Research University, Centre for Robotics, 60 Boulevard Saint-Michel, 75006, Paris, France
*Safran Tech, Groupe Safran, Rue des Jeunes Bois-Châteaufort, 78772, Magny Les Hameaux Cedex, France

***Abstract*—The present paper introduces a novel methodology for Unscented Kalman Filtering (UKF) on manifolds that extends our previous work about UKF on Lie groups. Beyond filtering performances, the main interests of the approach are its versatility, as the method applies to numerous state estimation problems, and its simplicity of implementation for practitioners not being necessarily familiar with manifolds and Lie groups. We have developed the method on two independent open-source Python and Matlab frameworks we call *UKF-M*, for quickly implementing and testing the approach. The online repositories contain tutorials, documentations, and various relevant robotics examples that the user can readily reproduce and then adapt, for fast prototyping and benchmarking. The code is available at `https://github.com/CAOR-MINES-ParisTech/ukfm`.**

I. INTRODUCTION

Over the past fifty years, the Kalman filter has been a pervasive tool in aerospace engineering and beyond, to estimate the state of a system subject to dynamical evolution, see e.g. [1]. When system's dynamics are governed by nonlinear equations, one generally resorts to a variant called the Extended Kalman Filter (EKF), or to the more recent Unscented Kalman Filter (UKF) [2,3]. There has been various attempts to adapt the EKF and (respectively) UKF to the case where the system's state lives in a manifold $\mathcal{M}$, see respectively [4] and [5]–[8].

In this paper we introduce *UKF-M*, a novel and general method for UKF on manifolds whose versatility allows direct application to all manifolds encountered in practice. The theory is supported with independent Python and Matlab open sourced implementations. The framework is well documented, and contains a number of examples that can be readily run and then adapted, where our methodology spares the analytic computation of Jacobians (contrary in EKF) and is thus well suited to fast prototyping and benchmarking.

Filtering on manifolds is historically motivated by aerospace applications where one seeks to estimate (besides other quantities) the orientation of a body in space, much work was devoted to making the EKF work with orientations, namely quaternions. The idea is then to make the EKF estimate an error instead of the state directly, leading to error state EKFs, see [4,9]–[11] and their UKF counterparts [12]–[14]. The set of orientations of a body in space forms a Lie group, namely $SO(3)$ and efforts devoted to estimation on $SO(3)$ has paved the way to extended Kalman filtering on Lie groups, see [1,15]–[19] and unscented Kalman filtering on Lie groups, see [7,8,13,20]–[23].

Lie groups play a prominent role in robotics [24]. In the context of state estimation and localization, viewing poses as elements of the Lie group $SE(3)$ has proved relevant [25]–[31]. The use of the novel Lie group $SE_2(3)$ introduced in [19] has also led to drastic improvement of Kalman filters for robot state estimation [1,19,32]–[37]. Similarly, using the group $SE_k(n)$ introduced for Simultaneous Localization And Mapping (SLAM) in [38,39] makes EKF-SLAM consistent, see [39]–[43]. Finally, there has been attempts to devise UKFs that respect natural symmetries of the systems' dynamics, namely the invariant UKF, see [44,45].

Besides providing a comprehensive code, our main contribution in terms of methodology is to introduce a novel and general framework for UKF on manifolds that is simpler than existing methods, and whose versatility allows direct application to all manifolds encountered in practice. Indeed, [7,8] proposes UKF implementations based on the Levi-Civita connexion but mastering connections is difficult. [7,13,20,21] are reserved for $SO(3)$ and $SE(3)$, while [23] is reserved for Lie groups and requires more knowledge of Lie theory than the present paper.

In Section II, we introduce a user-friendly approach to UKF on parallelizable manifolds. Section III applies the approach in the particular case where the manifold is a Lie group and recovers [22], but without requiring much knowledge of Lie groups. Section IV describes the open sourced framework. We then show in Section V the method may actually be extended to all manifolds encountered in robotics. To this respect, some theoretical issues and clarifications related to Kalman filtering on manifolds are provided at the end of the article.

II. UNSCENTED KALMAN FILTERING ON PARALLELIZABLE MANIFOLDS

In this section we describe our simple methodology for UKF on parallelizable manifolds. Owing to space limits, we assume the reader to have approximate prior knowledge and intuition about manifolds and tangent spaces.

A. Parallelizable Manifolds

In order to "write" the equations of the extended or the unscented Kalman filter on a manifold, it may be advantageous to have a global coordinate system.

Definition 1: A smooth manifold $\mathcal{M}$ of dimension d is said parallelizable if there exists a set of smooth vector fields $\{V_1, V_2, \cdots, V_d\}$ on the manifold such that for any point $\boldsymbol{\mathcal{X}} \in \mathcal{M}$ the tangent vectors $\{V_1(\boldsymbol{\mathcal{X}}), V_2(\boldsymbol{\mathcal{X}}), \cdots, V_d(\boldsymbol{\mathcal{X}})\}$ form a basis of the tangent space at $\boldsymbol{\mathcal{X}}$.

The cylinder $\{(x, y, z) \in \mathbb{R}^3 \mid x^2 + y^2 = 1\}$ is a basic example with $d = 2$ and $V_1(x, y, z) = (y, -x, z)$, $V_2 =$

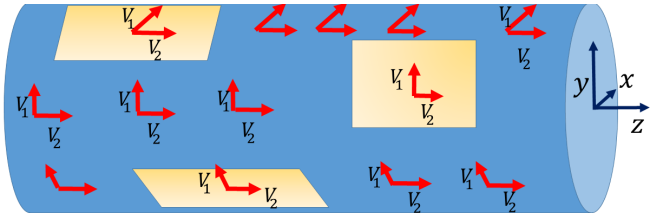

Fig. 1: The cylinder is a parallelizable manifold. We can define vector fields V_1, V_2 that form a basis of the tangent space at any point.

$(0,0,1)$ are two tangent vectors that form a local basis at (x,y,z), see Figure 1. The cylinder has null curvature but the notion of parallelizable manifolds is much broader. In particular, all Lie groups are parallelizable manifolds.

Example 1: For the rotation matrices $\mathbf{C} \in SO(3)$ let us first define the matrix

$$\boldsymbol{\omega}^\wedge = \begin{pmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{pmatrix}, \tag{1}$$

where $\boldsymbol{\omega} = (\omega_1, \omega_2, \omega_3)^T$, and choose as vector fields:

$$V_1(\mathbf{C}) = \mathbf{C}\mathbf{e}_1^\wedge,\ V_2(\mathbf{C}) = \mathbf{C}\mathbf{e}_2^\wedge,\ V_3(\mathbf{C}) = \mathbf{C}\mathbf{e}_3^\wedge, \tag{2}$$

where $\mathbf{e}_1 = (1,0,0)^T$, $\mathbf{e}_2 = (0,1,0)^T$, and $\mathbf{e}_3 = (0,0,1)^T$.

It should be noted, though, that not all manifolds fall in this category. However, we will see in Section V using an embedding into a parallelizable manifold, our methodology applies to virtually all manifolds encountered in robotics.

B. Uncertainty Representation on Parallelizable Manifolds

Our goal is to estimate the state $\boldsymbol{\mathcal{X}} \in \mathcal{M}$ given all the sensor measurements. As sensors are flawed, it is impossible to exactly reconstruct $\boldsymbol{\mathcal{X}}$. Instead a filter maintains a "belief" about the state, that is, its statistical distribution given sensor information. The Kalman filter maintains a Gaussian belief such that $\boldsymbol{\mathcal{X}} \sim \mathcal{N}\left(\hat{\boldsymbol{\mathcal{X}}}, \mathbf{P}\right)$, which may be re-written in the form:

$$\boldsymbol{\mathcal{X}} = \hat{\boldsymbol{\mathcal{X}}} + \boldsymbol{\xi},\ \boldsymbol{\xi} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{P}\right). \tag{3}$$

We see that the belief is encoded using only a "noise-free" or "mean" estimate $\hat{\boldsymbol{\mathcal{X}}}$, and a covariance matrix $\mathbf{P}$ that encodes the extent of dispersion of the belief around the estimate.

Consider a parallelizable manifold $\mathcal{M}$, and let $\{V_1, V_2, \cdots, V_d\}$ denote the associated vector fields. To devise a similar belief on $\mathcal{M}$, one needs of course local coordinates to write the mean $\hat{\boldsymbol{\mathcal{X}}} \in \mathcal{M}$. This poses no problem, though. The harder part is to find a way to encode dispersion around the estimate $\hat{\boldsymbol{\mathcal{X}}}$. It is now commonly admitted, see references above, that the tangent space at $\hat{\boldsymbol{\mathcal{X}}}$ should encode such dispersion, and that covariance $\mathbf{P}$ should hence reflect dispersion in the tangent space. As additive noise (3) makes no sense for $\boldsymbol{\mathcal{X}} \in \mathcal{M}$, we define a probability distribution $\boldsymbol{\mathcal{X}} \sim \mathcal{N}_\varphi(\hat{\boldsymbol{\mathcal{X}}}, \mathbf{P})$, for the random variable $\boldsymbol{\mathcal{X}} \in \mathcal{M}$ as

$$\boldsymbol{\mathcal{X}} = \varphi\left(\hat{\boldsymbol{\mathcal{X}}}, \boldsymbol{\xi}\right),\ \boldsymbol{\xi} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{P}\right), \tag{4}$$

where $\varphi : \mathcal{M} \times \mathbb{R}^d \to \mathcal{M}$ is a smooth function freely chosen by the user and satisfying $\varphi\left(\hat{\boldsymbol{\mathcal{X}}}, \mathbf{0}\right) = \hat{\boldsymbol{\mathcal{X}}}$, and $\boldsymbol{\xi} \in \mathbb{R}^d$ a random Gaussian vector that encodes directions of the tangent space, $\mathcal{N}(.,.)$ is the classical Gaussian distribution in Euclidean space, and $\mathbf{P} \in \mathbb{R}^{d\times d}$ the associated covariance matrix. Using the parallelizable manifold property, we implicity use coordinates in the tangent space, as $\boldsymbol{\xi} = (\xi^{(1)}, \xi^{(2)}, \cdots, \xi^{(d)})^T \in \mathbb{R}^d$ encodes in fact the tangent vector $\xi^{(1)} V_1(\hat{\boldsymbol{\mathcal{X}}}) + \cdots + \xi^{(d)} V_d(\hat{\boldsymbol{\mathcal{X}}})$. Hence φ is called a "retraction", see, e.g., [46]. In (4), the noise-free quantity $\hat{\boldsymbol{\mathcal{X}}}$ is viewed as the mean, and the dispersion arises through φ. We stress that the distribution defined at (4) is not Gaussian. It is only Gaussian in coordinates related to map φ.

Example 2: Consider Example 1. Recall tangent vectors at $\mathbf{C}$ indicate small motions around $\mathbf{C} \in SO(3)$. Tangent vector $\mathbf{C}\boldsymbol{\omega}^\wedge$ indeed writes $\omega_1 V_1(\mathbf{C}) + \omega_2 V_2(\mathbf{C}) + \omega_3 V_3(\mathbf{C})$, see (2). We can then choose for φ the following $\varphi(\mathbf{C}, \boldsymbol{\omega}) = \mathbf{C}\exp\left(\boldsymbol{\omega}^\wedge\right)$, which is the exponential map on $SO(3)$.

Finding an appropriate map φ is not always straightforward. However there exists (in theory) some "canonical" φ.

Proposition 1: One may define define $\varphi(\hat{\boldsymbol{\mathcal{X}}}, \boldsymbol{\xi})$ as the point of $\mathcal{M}$ obtained by starting from $\hat{\boldsymbol{\mathcal{X}}}$ and integrating the vector field $\sum_{i=1}^d \boldsymbol{\xi}^{(i)} V_i$ during one unit of time. In that case we call φ an "exponential map".

However, we generally have no closed form for the exponential map and one resorts to simpler retractions φ.

C. Bayesian Estimation Using the Unscented Transform

Consider a random variable $\boldsymbol{\mathcal{X}} \in \mathcal{M}$ with prior probability distribution $p\left(\boldsymbol{\mathcal{X}}\right)$. Suppose we obtain some additional information about $\boldsymbol{\mathcal{X}}$ through a measurement $\mathbf{y}$. The goal is to compute the posterior distribution $p(\boldsymbol{\mathcal{X}}|\mathbf{y})$. Let

$$\mathbf{y} = h\left(\boldsymbol{\mathcal{X}}\right) + \mathbf{v}, \tag{5}$$

be a measurement, where $h(.) : \mathcal{M} \to \mathbb{R}^p$ represents the observation function and $\mathbf{v} \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$ is a white Gaussian noise in $\mathbb{R}^p$ with known characteristics. The problem of Bayesian estimation we consider is as follows:

1) assume the prior distribution to follow (4) with known parameters $\hat{\boldsymbol{\mathcal{X}}}$ and $\mathbf{P}$;
2) assume one measurement $\mathbf{y}$ of (5) is available;
3) approximate the posterior distribution as

$$p(\boldsymbol{\mathcal{X}}|\mathbf{y}) \approx \varphi(\hat{\boldsymbol{\mathcal{X}}}^+, \boldsymbol{\xi}^+), \tag{6}$$

where $\boldsymbol{\xi}^+ \sim \mathcal{N}(\mathbf{0}, \mathbf{P}^+)$, and find parameters $\hat{\boldsymbol{\mathcal{X}}}^+$ and $\mathbf{P}^+$.

By letting $\boldsymbol{\mathcal{X}} = \varphi\left(\hat{\boldsymbol{\mathcal{X}}}, \boldsymbol{\xi}\right)$ in (5), we see $\mathbf{y}$ provides an information about $\boldsymbol{\xi} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{P}\right)$ and we may use the unscented transform of [2,3] to approximate the posterior $p(\boldsymbol{\xi}|\mathbf{y})$ for $\boldsymbol{\xi}$ as follows, see Algorithm 1: we compute a finite number of samples $\boldsymbol{\xi}_j$, $j = 1, \ldots, 2d$ (where λ is a scale parameter, see [3,22]), and pass each of these so-called sigma points through the measurement function

$$\mathbf{y}_j = h\left(\varphi(\hat{\boldsymbol{\mathcal{X}}}, (\boldsymbol{\xi}_j)\right),\ j = 1, \ldots, 2d. \tag{7}$$

Algorithm 1: Bayesian updating on parallelizable manifolds with prior (4) and observation (5)

Input: $\hat{\boldsymbol{\chi}}, \mathbf{P}, \mathbf{y}, \mathbf{R}$;

`// set sigma points`

$\boldsymbol{\xi}_j = \mathrm{col}(\sqrt{(\lambda+d)\mathbf{P}})_j,\ j = 1, \dots, d,$

$\boldsymbol{\xi}_j = -\mathrm{col}(\sqrt{(\lambda+d)\mathbf{P}})_{j-d},\ j = d+1, \dots, 2d;$

`// compute measurement sigma points`

$\mathbf{y}_0 = h(\varphi(\hat{\boldsymbol{\chi}}, \mathbf{0}));$

$\mathbf{y}_j = h(\varphi(\hat{\boldsymbol{\chi}}, \boldsymbol{\xi}_j)),\ j = 1, \dots, 2d;$

`// infer covariance matrices`

$\bar{\mathbf{y}} = w_m \mathbf{y}_0 + \sum_{j=1}^{2d} w_j \mathbf{y}_j;$

$\mathbf{P}_{\mathbf{yy}} = \sum_{j=0}^{2d} w_j (\mathbf{y}_j - \bar{\mathbf{y}})(\mathbf{y}_j - \bar{\mathbf{y}})^T + \mathbf{R};$

$\mathbf{P}_{\boldsymbol{\xi}\mathbf{y}} = \sum_{j=1}^{2d} w_j \boldsymbol{\xi}_j (\mathbf{y}_j - \bar{\mathbf{y}})^T;$

`// update state and covariance`

$\mathbf{K} = \mathbf{P}_{\boldsymbol{\xi}\mathbf{y}} \mathbf{P}_{\mathbf{yy}}^{-1}$; `// gain matrix`

$\hat{\boldsymbol{\chi}}^+ = \varphi(\hat{\boldsymbol{\chi}}, \mathbf{K}(\mathbf{y} - \bar{\mathbf{y}}));$

$\mathbf{P}^+ = \mathbf{P} - \mathbf{K}\mathbf{P}_{\mathbf{yy}}\mathbf{K}^T;$

Output: $\hat{\boldsymbol{\chi}}^+, \mathbf{P}^+;$

By noting $\mathbf{y}_0 = h(\varphi(\hat{\boldsymbol{\chi}}, \mathbf{0}))$ we then compute successively the measurement mean $\bar{\mathbf{y}} = w_m \mathbf{y}_0 + \sum_{j=1}^{2d} w_j \mathbf{y}_j$, the measurement covariance $\mathbf{P}_{\mathbf{yy}} = \sum_{j=0}^{2d} w_j (\mathbf{y}_j - \bar{\mathbf{y}})(\mathbf{y}_j - \bar{\mathbf{y}})^T + \mathbf{R}$ and the cross-covariance $\mathbf{P}_{\boldsymbol{\xi}\mathbf{y}} = \sum_{j=1}^{2d} w_j \boldsymbol{\xi}_j (\mathbf{y}_j - \bar{\mathbf{y}})^T$, where w_m and w_j are weights defined in [3,22]. We then derive the conditional distribution of $\boldsymbol{\xi} \in \mathbb{R}^d$ as

$$p(\boldsymbol{\xi}|\mathbf{y}) \sim \mathcal{N}\left(\bar{\boldsymbol{\xi}}, \mathbf{P}^+\right), \text{ where} \tag{8}$$

$$\mathbf{K} = \mathbf{P}_{\boldsymbol{\xi}\mathbf{y}} \mathbf{P}_{\mathbf{yy}}, \bar{\boldsymbol{\xi}} = \mathbf{K}\left(\mathbf{y} - \bar{\mathbf{y}}\right), \mathbf{P}^+ = \mathbf{P} - \mathbf{K}\mathbf{P}_{\mathbf{yy}}\mathbf{K}^T. \tag{9}$$

This may be viewed as a Kalman update on the error $\boldsymbol{\xi}$, in the vein of error state Kalman filtering, see e.g. [11]. The problem is then to convert this into a distribution on the manifold in the form (4). We first write $p(\boldsymbol{\xi}|\mathbf{y})$ as $\bar{\boldsymbol{\xi}} + \boldsymbol{\xi}^+$ with $\boldsymbol{\xi}^+ \sim \mathcal{N}(\mathbf{0}, \mathbf{P}^+)$, to show that the posterior may write $\varphi\left(\hat{\boldsymbol{\chi}}, \bar{\boldsymbol{\xi}} + \boldsymbol{\xi}^+\right)$ with $\boldsymbol{\xi}$ considered as a fixed noise free parameter. We suggest to define the posterior $p(\boldsymbol{\chi}|\mathbf{y})$ as

$$\boldsymbol{\chi} \approx \varphi(\hat{\boldsymbol{\chi}}^+, \boldsymbol{\xi}^+), \quad \boldsymbol{\xi}^+ \sim \mathcal{N}\left(\mathbf{0}, \mathbf{P}^+\right), \tag{10}$$

where we have let

$$\hat{\boldsymbol{\chi}}^+ = \varphi\left(\hat{\boldsymbol{\chi}}, \bar{\boldsymbol{\xi}}\right). \tag{11}$$

Note the approximation done in (10)-(11) actually consists in writing $\varphi\left(\hat{\boldsymbol{\chi}}, \bar{\boldsymbol{\xi}} + \boldsymbol{\xi}^+\right) \approx \varphi\left(\varphi\left(\hat{\boldsymbol{\chi}}, \bar{\boldsymbol{\xi}}\right), \boldsymbol{\xi}^+\right)$.

When $\mathcal{M} = \mathbb{R}^d$ the latter equality holds up to the first order in the dispersions $\bar{\boldsymbol{\xi}}$, $\boldsymbol{\xi}^+$, both assumed small. In the case where $\mathcal{M}$ is not a vector space, it may be geometrically interpreted as saying that moving from $\hat{\boldsymbol{\chi}}$ along the direction $\bar{\boldsymbol{\xi}} + \boldsymbol{\xi}^+$ approximately consists in moving from $\hat{\boldsymbol{\chi}}$ along $\bar{\boldsymbol{\xi}}$ and then from the obtained point on $\mathcal{M}$ along $\boldsymbol{\xi}^+$.

D. Unscented Kalman Filtering on Parallelizable Manifolds

Consider the dynamics

$$\boldsymbol{\chi}_n = f\left(\boldsymbol{\chi}_{n-1}, \boldsymbol{\omega}_n, \mathbf{w}_n\right), \tag{12}$$

where the state $\boldsymbol{\chi}_n$ lives in a parallelizable manifold $\mathcal{M}$, $\boldsymbol{\omega}_n$ is a known input variable and $\mathbf{w}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_n)$ is a white Gaussian noise in $\mathbb{R}^q$. We consider observations of the form

$$\mathbf{y}_n = h\left(\boldsymbol{\chi}_n\right) + \mathbf{v}_n, \tag{13}$$

where $\mathbf{v}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_n)$ is a white Gaussian noise with known covariance. For system (12)-(13), we model the state posterior conditional on past measurements using the uncertainty representation (4), that is, $\boldsymbol{\chi}_n = \varphi(\hat{\boldsymbol{\chi}}_n, \boldsymbol{\xi}_n)$ with $\boldsymbol{\xi}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{P}_n)$. To propagate the state, we need to be able to locally invert $\boldsymbol{\chi} \mapsto \varphi(\hat{\boldsymbol{\chi}}, \boldsymbol{\xi})$, i.e., to find a map we denote by $\varphi_{\hat{\boldsymbol{\chi}}}^{-1}(\cdot) : \mathcal{M} \to \mathbb{R}^d$ such that

$$\varphi_{\hat{\boldsymbol{\chi}}}^{-1}\left(\varphi(\hat{\boldsymbol{\chi}}, \boldsymbol{\xi})\right) = \boldsymbol{\xi} + O(||\boldsymbol{\xi}||^2), \tag{14}$$

that is, a map that allows one to assess the discrepancy between $\hat{\boldsymbol{\chi}}$ and $\varphi(\hat{\boldsymbol{\chi}}, \boldsymbol{\xi})$ is $\boldsymbol{\xi}$ indeed. Starting from the prior distribution $p\left(\boldsymbol{\chi}_{n-1}\right) \sim \varphi(\hat{\boldsymbol{\chi}}_{n-1}, \boldsymbol{\xi}_{n-1})$ with $\boldsymbol{\xi}_{n-1} \sim \mathcal{N}(\mathbf{0}, \mathbf{P}_{n-1})$ and $\hat{\boldsymbol{\chi}}_{n-1}$ and $\mathbf{P}_{n-1}$ known, we seek to compute the state propagated distribution in the form

$$p\left(\boldsymbol{\chi}_n | \boldsymbol{\chi}_{n-1}\right) \sim \varphi(\hat{\boldsymbol{\chi}}_n, \boldsymbol{\xi}_n) \quad \text{with} \quad \boldsymbol{\xi}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{P}_n). \tag{15}$$

Once sigma points have been defined through (4) and then (12), to find $\hat{\boldsymbol{\chi}}_n$ one is faced with an optimization problem of computing a weighted mean on the manifold $\mathcal{M}$. This route has already been advocated in [12]–[14,23]. However, to keep the implementation simple and analog to the EKF, we suggest to propagate the mean using the unnoisy state model, leading to

$$\hat{\boldsymbol{\chi}}_n = f(\hat{\boldsymbol{\chi}}_{n-1}, \boldsymbol{\omega}_n, \mathbf{0}). \tag{16}$$

To compute the covariance $\mathbf{P}_n$ from covariance $\mathbf{P}_{n-1}$ of $\boldsymbol{\xi}_{n-1}$ we use the fact $\mathbf{w}_n$ and $\boldsymbol{\xi}_{n-1}$ are uncorrelated and proceed in two steps. We first generate sigma points in $\mathbb{R}^d$ corresponding to $\mathbf{P}_{n-1}$ and pass them through the unnoisy model (16) for nonlinear propagation of $\mathbf{P}_{n-1}$ through f. We obtain points $\boldsymbol{\chi}_n^j$ on the manifold $\mathcal{M}$, and as distribution of propagated state $\hat{\boldsymbol{\chi}}_n$ is described as $\varphi\left(\hat{\boldsymbol{\chi}}_n, \boldsymbol{\xi}_n\right)$, and $\hat{\boldsymbol{\chi}}_n$ is known from (16), we may invert the latter using $\varphi_{\hat{\boldsymbol{\chi}}_n}^{-1}$ to obtain sigma points back in $\mathbb{R}^d$ and compute their empirical covariance $\boldsymbol{\Sigma}_n$. Then, we generate sigma points for process noise $\mathbf{w}_n$ similarly and obtain another covariance matrix encoding dispersion in $\mathbb{R}^d$ owed to noise, that adds up to $\boldsymbol{\Sigma}_n$ and thus clearly distinguish the contribution of the state error $\boldsymbol{\xi}_n$ from the noise $\mathbf{w}_n$. When a new measurement arrives, belief is updated via Algorithm 1. Algorithm 2 summarizes both steps, where the weights defined through `set_weights`(d, α) depend on a scale parameter α (generally set between 10^{-3} and 1), and sigma point dimension, see [3,22] and documentation in source code.

Using (16) to propagate the mean while using sigma points to compute covariance is also made in the method of [30], in the particular case of pose compounding on $SE(3)$, with φ the $SE(3)$ exponential map.

III. Application to UKF on Lie Groups

To apply the preceding methodology to any d-dimensional group $G = \mathcal{M}$, one first defines a basis of the Lie algebra. Then, to any vector $\boldsymbol{\xi} \in \mathbb{R}^d$, one may associate

Algorithm 2: UKF on parallelizable manifolds

Input: $\hat{\boldsymbol{\chi}}_{n-1}, \mathbf{P}_{n-1}, \boldsymbol{\omega}_n, \mathbf{Q}_n, \mathbf{y}_n, \mathbf{R}_n, \alpha$;

Propagation

```
// propagate mean state
```
$\hat{\boldsymbol{\chi}}_n = f(\hat{\boldsymbol{\chi}}_{n-1}, \boldsymbol{\omega}_n, \mathbf{0})$;
```
// propagate state error covariance
```
$\lambda, \{w_j\}_{j=0,\dots,2d} = \texttt{set_weights}(d, \alpha)$;

$\boldsymbol{\xi}_j = \mathrm{col}(\sqrt{(\lambda+d)\mathbf{P}_{n-1}})_j,\ j = 1, \dots, d,$

$\boldsymbol{\xi}_j = -\mathrm{col}(\sqrt{(\lambda+d)\mathbf{P}_{n-1}})_{j-d},\ j = d+1, \dots, 2d;$
```
// use retraction onto manifold
```
$\boldsymbol{\chi}_n^j = f(\varphi(\hat{\boldsymbol{\chi}}_{n-1}, \boldsymbol{\xi}_j), \boldsymbol{\omega}_n, \mathbf{0}), j = 1, \dots, 2d;$
```
// inverse retract to go back in ℝ^d
```
$\boldsymbol{\Sigma}_n = \sum_{j=1}^{2d} w_j \varphi^{-1}_{\hat{\boldsymbol{\chi}}_n}(\boldsymbol{\chi}_n^j)$;
```
// proceed similarly for noise
```
$\lambda, \{w_j\}_{j=0,\dots,2q} = \texttt{set_weights}(q, \alpha)$;

$\mathbf{w}^j = \mathrm{col}(\sqrt{(\lambda+q)\mathbf{Q}_n})_j,\ j = 1, \dots, q,$

$\mathbf{w}^j = -\mathrm{col}(\sqrt{(\lambda+q)\mathbf{Q}_n})_{j-d},\ j = q+1, \dots, 2q;$

$\boldsymbol{\chi}_n^j = f(\hat{\boldsymbol{\chi}}_{n-1}, \boldsymbol{\omega}_n, \mathbf{w}^j), j = 1, \dots, 2q;$

$\mathbf{P}_n = \boldsymbol{\Sigma}_n + \sum_{j=1}^{2q} w_j \varphi^{-1}_{\hat{\boldsymbol{\chi}}_n}(\boldsymbol{\chi}_n^j)$;

Update (when measurement $\mathbf{y}_n$ arrives)

Compute $\hat{\boldsymbol{\chi}}_n^+, \mathbf{P}_n^+$ from Algorithm 1 with $\hat{\boldsymbol{\chi}}_n, \mathbf{P}_n$;

Output: $\hat{\boldsymbol{\chi}}_n^+, \mathbf{P}_n^+$;

an element denoted $\boldsymbol{\xi}^\wedge$ of the Lie algebra $\mathfrak{g}$. Let denote the vee operator $\vee$ its inverse, as in e.g., [30]. The Lie exponential map "exp" maps elements of the Lie algebra to the group. After a basis of the Lie algebra has been defined, in (4) we may choose $\varphi(\hat{\boldsymbol{\chi}}, \boldsymbol{\xi}) := \hat{\boldsymbol{\chi}} \exp(\boldsymbol{\xi}^\wedge)$, which corresponds to left concentrated Gaussians on Lie groups [18]. Note that, in the Lie group case, choosing left invariant vector fields for the V_i's and following Proposition 1 we exactly recover the latter expression.

We may invert φ using the logarithm map $\exp^{-1}$ of G, and we get $\varphi^{-1}_{\hat{\boldsymbol{\chi}}}(\boldsymbol{\chi}) := \exp^{-1}\left((\hat{\boldsymbol{\chi}}^{-1}\boldsymbol{\chi})^\vee\right) = \log\left(\hat{\boldsymbol{\chi}}^{-1}\boldsymbol{\chi}\right)$. This may be summarized as

$$\varphi(\hat{\boldsymbol{\chi}}, \boldsymbol{\xi}) := \hat{\boldsymbol{\chi}} \exp(\boldsymbol{\xi}^\wedge),\ \varphi^{-1}_{\hat{\boldsymbol{\chi}}}(\boldsymbol{\chi}) := \log\left(\hat{\boldsymbol{\chi}}^{-1}\boldsymbol{\chi}\right). \quad (17)$$

If we alternatively privilegiate right multiplications we have

$$\varphi(\hat{\boldsymbol{\chi}}, \boldsymbol{\xi}) := \exp(\boldsymbol{\xi}^\wedge)\hat{\boldsymbol{\chi}},\ \varphi^{-1}_{\hat{\boldsymbol{\chi}}}(\boldsymbol{\chi}) := \log\left(\boldsymbol{\chi}\hat{\boldsymbol{\chi}}^{-1}\right). \quad (18)$$

A. Applications in Mobile Robotics: the Group $SE_k(d)$

It is well known that orientations of body in spaces are described by elements of $SO(3)$. It is also well known that the use of $SE(3)$ is advantageous to describe the position and the orientation of a robot (pose), especially for estimation, see [25]–[31]. In [19,47] the group of double direct isometries $SE_2(3)$ was introduced to address estimation problems for robot navigation when the motion equations are based on an Inertial Measurement Unit (IMU). In [38,39] the group of multiple spatial isometries $SE_k(d)$ was introduced in the context of SLAM. The group $SE_k(d)$, that allows recovering $SE(3)$ with $k = 1, d = 3$, $SE(2)$ with $k = 1, d = 2$ and $SO(3)$ with $k = 0, d = 3$, seems to cover virtually all robotics applications where the Lie group methodology has been so far useful (along with trivial extensions to be mentioned in Section III-B). Since it was introduced for navigation and SLAM, this group has been successfully used in various contexts, see [1,19,32]–[37,39]–[43,48,49]. For more information, see the code documentation associated with the present paper.

B. The Mixed Case

We call mixed the case where $\mathcal{M} = G \times \mathbb{R}^N$. This typically arises when one wants to estimate some additional parameters besides the state assumed to live in the group G, such as sensor biases. By decomposing the state as $\hat{\boldsymbol{\chi}} = (\hat{\boldsymbol{\chi}}_1, \hat{\boldsymbol{\chi}}_2) = G \times \mathbb{R}^N$ and letting $\boldsymbol{\xi} = (\boldsymbol{\xi}_1, \boldsymbol{\xi}_2)$, we typically define φ as

$$\varphi\left(\hat{\boldsymbol{\chi}}, \boldsymbol{\xi}\right) = \left(\exp\left(\boldsymbol{\xi}_1\right)\hat{\boldsymbol{\chi}}_1, \hat{\boldsymbol{\chi}}_2 + \boldsymbol{\xi}_2\right) \quad (19)$$

or if left multiplications are privilegiated $\varphi\left(\hat{\boldsymbol{\chi}}, \boldsymbol{\xi}\right) = \left(\hat{\boldsymbol{\chi}}_1 \exp\left(\boldsymbol{\xi}_1\right), \hat{\boldsymbol{\chi}}_2 + \boldsymbol{\xi}_2\right)$. This way, as many additional quantities as desired may be estimated along the same lines.

Remark 1: When $G = SE(3)$ for example, it is tempting to let $G' = SO(3)$ and to treat $SE(3)$ as $SO(3) \times \mathbb{R}^3$ along the lines of mixed systems. However, in robotics contexts, it has been largely argued the Lie group structure of $SE(3)$ to treat poses is more relevant than $SO(3) \times \mathbb{R}^3$, as accounting for the coupling between orientation and position leads to important properties, see [25]–[31]. In the same way, $SE_k(3)$ resembles $SO(3) \times \mathbb{R}^{3k}$ but has a special noncommutative group structure having recently led to many successes in robotics, see [1,19,32]–[37,39]–[43,48,49].

Example 3: The state $\boldsymbol{\chi}$ for fusing IMU with GNSS is generally divided into the vehicle state $\boldsymbol{\chi}_1 \in G$ (orientation, velocity and position of the vehicle) along with IMU biases $\boldsymbol{\chi}_2 = \mathbf{b} \in \mathbb{R}^6$, see e.g. our example on the KITTI dataset [50]. Independently from the vehicle state error representation the function φ always has the form of (19) for left multiplication: augmenting $\boldsymbol{\chi}_2$ with new parameters, e.g. time synchronization and force variables [51] is immediate.

IV. UKF-M Implementation

We have released both open source Python package and Matlab toolbox *UKF-M* implementations of our method at `https://github.com/CAOR-MINES-ParisTech/ukfm`. Both implementations are wholly independent, and their design guidelines pursue simplicity, intuitiveness and easy adaptation rather than optimization. We adapt the code to the user preferences as follow: the Python code follows class-object paradigm and is heavily documented through the Sphinx documentation generator [52], whereas the Matlab toolbox contains equivalent functions without class as we believe choosing well function names is best suited for the Matlab use as compared to class definition. The following code snippets are based on the Python package that we recommend using.

A. Recipe for Designing an UKF on Manifolds

To devise an UKF for any fusion problem on a parrallelizable manifold (or Lie group) $\mathcal{M}$ the ingredients required in terms of implementation are as follows, see Snippet 1.

Snippet 1: how to devise an UKF on manifolds

```
ukf = ukfm.UKF(
f=model.f,                # propagation model
h=model.h,                # observation model
phi=user.phi,             # retraction
phi_inv=user.phi_inv,     # inverse retraction
Q=model.Q,                # process  cov.
R=model.R,                # observation  cov.
alpha=user.alpha          # sigma point param.
state0=state0,            # initial state
P0=P0)                    # initial covariance
```

Snippet 2: setting φ, φ^{-1} for $\boldsymbol{\chi} := (\mathtt{Rot} \in SO(3), \mathtt{v}, \mathtt{p})$

```
def phi(state, xi):
return STATE(
Rot=state.Rot.dot(SO3.exp(xi[0:3])),
v=state.v + xi[3:6]
p=state.p + xi[6:9])

def phi_inv(state, hat_state):
return np.hstack([ # concatenate errors
SO3.log(state.Rot.dot(hat_state.Rot.T)),
state.v - hat_state.v,
state.p - hat_state.p])
```

1) A model that specifies the functions f and h used in the filter;
2) An uncertainty representation (4). This implies an expression for the function φ and its inverse φ^{-1} are defined by the user;
3) Filter parameters, that define noise covariance matrices $\mathbf{Q}_n$, $\mathbf{R}_n$ and weights (λ, w_m, and w_j) through α. Noise covariance values are commonly guided by the model and tuned by the practitioner, whereas α is generally set between 10^{-3} and 1 [3].
4) Initial state estimates $\hat{\boldsymbol{\chi}}_0$ and $\mathbf{P}_0$.

Example 4: Consider a 3D model whose state contains a rotation matrix $\mathtt{Rot} \in SO(3)$, the velocity $\mathtt{v} \in \mathbb{R}^3$ and position $\mathtt{p} \in \mathbb{R}^3$ of a moving vehicle. Defining φ and φ^{-1} allows computing (respectively) a new state and a state error. One possibility is given in Snippet 2, where $\boldsymbol{\chi} \in SO(3) \times \mathbb{R}^6$, $\varphi(\boldsymbol{\chi}, \boldsymbol{\xi}) = \left(\mathtt{Rot}\exp(\boldsymbol{\xi}^{(0:3)}), \mathtt{v} + \boldsymbol{\xi}^{(3:6)}, \mathtt{p} + \boldsymbol{\xi}^{(6:9)}\right)$ and $\varphi^{-1}(\boldsymbol{\chi}, \hat{\boldsymbol{\chi}}) = (\log(\mathtt{Rot}\hat{\mathtt{Rot}}^T), \mathtt{v} - \hat{\mathtt{v}}, \mathtt{p} - \hat{\mathtt{p}})$.

To devise an UKF for any Bayesian fusion problem on a Lie group we follow the rules given in Section IV-A and simplify step 2) as follows: we pick an uncertainty representation, either (17) or (18). This directly implies an expression for the functions $\wedge$ and its inverse $\vee$, as well as for the exponential $\exp$ and its (local) inverse $\log$. Applying the present general methodology for the particular case of Lie groups, we thus recover algorithms of [22].

Example 5: we modify the representation of the state in Example 4 for viewing the state as a element $\boldsymbol{\chi} \in SE_2(3)$, defining thus two alternative retractions, see e.g. their φ^{-1} implementation in Snippet 3. Results are clear, see Figure 2, the $SE_2(3)$-UKF with right multiplications (18) outperforms the other filters.

Snippet 3: defining φ^{-1} via (17) or (18) for $\boldsymbol{\chi} \in SE_2(3)$

```
def phi_inv(state, hat_state):
  chi = state2chi(state)
  hat_chi = state2chi(hat_state)
  # if left multiplication (17)
  return SEK3.log(SEK3.inv(hat_hat).dot(chi))
  # if right multiplication (18)
  return SEK3.log(chi.dot(SEK3.inv(hat_hat)))
```

B. Implemented Examples

We implement the frameworks on relevant vanilla robotics examples which are listed as follows:

- 2D vanilla robot localization tutorial based on odometry and GNSS measurements;
- 3D attitude estimation from an IMU equipped with gyro, accelerometer and magnetometer;
- 3D inertial navigation on flat Earth where the vehicle obtains observations of known landmarks;
- 2D SLAM where the UKFs follows [53] to limit computational complexity and adding new observed landmarks in the state;
- IMU-GNSS fusion on the KITTI dataset [50];
- an example where the state lives on the 2-sphere manifold, modeling e.g. a spherical pendulum [54].

We finally enhance code framework, documentation and examples with filter performance comparisons: for each example we simulate Monte-Carlo data and benchmark UKFs and EKFs based on different choices of uncertainty representation (4) through accuracy and consistency metrics.

Example 6: Figure 2 displays two EKFs and two UKFs for inertial navigation in the setting of [19], where initial heading and position errors are large, respectively 45 degrees and 1 m. The second UKF, whose uncertainty representation (4) is based on $SE_2(3)$ exponential, see Section III-A, clearly outperforms the EKF, the first UKF, and improves the EKF of [19] during the first 10 seconds of the trajectory.

V. Extension to General Manifolds

The main problem when $\mathcal{M}$ is not parallelizable is that one cannot define a global uncertainty representation through a map φ as in (4), since $\boldsymbol{\xi} = (\boldsymbol{\xi}^{(1)}, \cdots, \boldsymbol{\xi}^{(d)})$ shall be interpreted as the coordinates in the tangent space, which means that one needs a basis $(V_1(\boldsymbol{\chi}), \cdots, V_d(\boldsymbol{\chi}))$ of the tangent space $T_{\boldsymbol{\chi}}\mathcal{M}$ at any $\boldsymbol{\chi}$. On general manifolds, one may use a local parallelization. Indeed, it is always possible to cover the manifold with patches $\mathcal{M}_1, \cdots, \mathcal{M}_K$, such that on each patch i we have a set of vector fields $(V_1^{(i)}, \cdots, V_d^{(i)})$ allowing one to apply the methodology. For instance on the 2-sphere one could choose a North-East frame in between the polar circles, and then some other smooth set of frames beyond polar circles. However two main issues arise. First, we feel such a procedure induces discontinuities at the polar circles that will inevitably degrade the filter perfomances. Indeed by moving $\hat{\boldsymbol{\chi}}$ slightly at the polar circle, one may obtain a jump in the distribution $\mathcal{N}_\varphi(\hat{\boldsymbol{\chi}}, \mathbf{P})$ with fixed

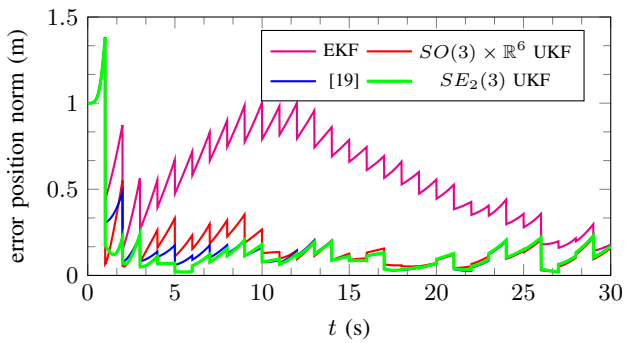

Fig. 2: Inertial navigation with heavy initial errors in the setting of [19]. $SE_2(3)$-UKF obtains the best results.

covariance $\mathbf{P}$, see Figure 3. Then, we see the obtained filter wholly depends on the way patches are chosen, which is undesirable: e.g., the use of stereographic coordinates, or Euler angles, may wholly change the behavior.

A. The Lifting "Trick"

It turns out a number of manifolds of interest called homogeneous spaces may be "lifted" to a Lie group. By simplicity[1] we consider the 2-sphere $\mathcal{M} = \mathbb{S}^2 = \{\mathbf{x} \in \mathbb{R}^3 \mid ||\mathbf{x}|| = 1\}$ with state $\mathbf{x}_n \in \mathbb{S}^2$. As $\mathbf{x}_{n+1}$ and $\mathbf{x}_n$ necessarily lie on the sphere, they are related by a rotation, that is,

$$\mathbf{x}_{n+1} = \mathbf{\Omega}_n \mathbf{x}_n \tag{20}$$

with $\mathbf{\Omega}_n \in SO(3)$ that may be written as $\exp(\boldsymbol{\omega}_n^\wedge)\exp(\mathbf{w}_n^\wedge)$ where $\boldsymbol{\omega}_n$ is known input, and $\mathbf{w}_n \sim \mathcal{N}(0, \mathbf{Q}_n)$ represents a noise, see (1) for the definition of wedge operator, and exp is the usual matrix exponential. We assume $\mathbf{x}_n$ is measured through a linear observation, that is,

$$\mathbf{y}_n = \mathbf{H}\mathbf{x}_n + \mathbf{v}_n \in \mathbb{R}^p. \tag{21}$$

Example 7: We provide a (novel) script which simulates a point of a pendulum with stiff wire living on a sphere, where two components are measured through e.g. a monocular camera, i.e. $\mathbf{H} = [\mathbf{e}_1, \mathbf{e}_2]^T$.

The dynamics can be lifted into $SO(3)$ by writing $\mathbf{x}_n$ via a rotation matrix $\mathbf{R}_n$, that is, we posit $\mathbf{x}_n = \mathbf{R}_n\mathbf{L}$ with $\mathbf{L} \in \mathbb{R}^3$. In terms of $\mathbf{R}_n$, dynamics (20) may be lifted letting $\mathbf{R}_{n+1} = \mathbf{\Omega}_n\mathbf{R}_n$ as then $\mathbf{R}_n\mathbf{L}$ satisfies (20) indeed. Similarly, the output in terms of $\mathbf{R}_n$ writes $\mathbf{y}_n = \mathbf{H}\mathbf{R}_n\mathbf{L} + \mathbf{v}_n = h(\mathbf{R}_n) + \mathbf{v}_n$. Having transposed the problem into estimation on the parallelizable manifold $SO(3)$, we can then apply the two UKFs by setting φ to either (17) or (18).

B. Covariance Retrieval

The practitioner may wonder how to retrieve the covariance in the original variables. Assume we have a Gaussian vector $\mathbf{x} \sim (\boldsymbol{\mu}, \mathbf{\Sigma})$, and we want to approximate $g(\mathbf{x})$ as a Gaussian. Of course this can be addressed resorting the unscented transform. However, a more basic and direct approach is as follows. Consider $\mathbf{A}$ a matrix and $\mathbf{b}$ a vector. Then it is known from probability theory that linear transformation preserve gaussianity and we have (exactly)

$$\mathbf{A}\mathbf{x} + \mathbf{b} \sim (\mathbf{A}\boldsymbol{\mu} + \mathbf{b}, \mathbf{A}\mathbf{\Sigma}\mathbf{A}^T). \tag{22}$$

Then, we can write $\mathbf{x} = \boldsymbol{\mu} + \mathbf{e}$ with $e \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$ and linearizing we find $g(\mathbf{x}) \approx g(\boldsymbol{\mu}) + \frac{\partial g}{\partial \mathbf{x}}(\boldsymbol{\mu})\mathbf{e}$ and applying linear Gaussian vectors transform yields approximately (i.e. as a first order approximation in the magnitude of the uncertainty) $g(\mathbf{x}) \sim (g(\boldsymbol{\mu}), \mathbf{A}\mathbf{\Sigma}\mathbf{A}^T)$, where we let $\mathbf{A} := \frac{\partial g}{\partial \mathbf{x}}(\boldsymbol{\mu})$.

In the 2-sphere example of the present section, our uncertainty representation may be taken as $\mathbf{R}_n = \hat{\mathbf{R}}_n \exp(\boldsymbol{\xi}^\wedge)$ with $\boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, \mathbf{P})$, see (17) and Example 2. As a result it is rather easy to compute the covariance matrix of $\mathbf{R}_n\mathbf{L}$ as follows. We may use linearizations to write that $\exp(\boldsymbol{\xi}^\wedge) \approx \mathbf{I} + \boldsymbol{\xi}^\wedge$ and thus $\mathbf{R}_n\mathbf{L} = \hat{\mathbf{R}}_n \exp(\boldsymbol{\xi}^\wedge)\mathbf{L} \approx \hat{\mathbf{R}}_n\mathbf{L} + \hat{\mathbf{R}}_n\boldsymbol{\xi}^\wedge\mathbf{L} = \hat{\mathbf{R}}_n\mathbf{L} - \hat{\mathbf{R}}_n\mathbf{L}^\wedge\boldsymbol{\xi} = \hat{\mathbf{R}}_n\mathbf{L} + \mathbf{A}\boldsymbol{\xi}$ with $\mathbf{A} = -\hat{\mathbf{R}}_n\mathbf{L}^\wedge$. As a result, the probability distribution of $\mathbf{R}_n\mathbf{L}$ is under a linear approximation $\mathcal{N}(\hat{\mathbf{R}}_n\mathbf{L}, \mathbf{A}\mathbf{P}\mathbf{A}^T)$.

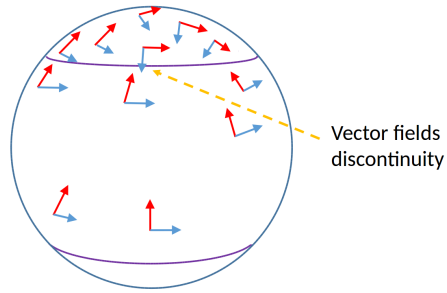

Fig. 3: We see covering the 2-sphere with 3 parallelizable patches (in between polar circles, and beyond each) inevitably induces discontinuities that may degrade filtering performances. Alluding to the "hairy ball theorem", this may be viewed as a punition for trying to comb a hairy ball.

VI. Theoretical Concluding Remarks

In fact using patches as described above, or augmenting the state to embed it in a parallelizable manifold, both serve the same purpose as using any parametrization of the state space: the user is provided with coordinates to write down the filter equations. But, similarly to a change of parameterization, each choice results in a different filter.

Once a parallelization and retraction are chosen the filter becomes independent from the coordinates, but is still dependent on choice of parallelization and retraction. The only role of parallelization, that is, retraction, is in fact to define how confidence ellipsoids are transported at the update step, and all the rest is unchanged. The fact UKF on manifolds necessarily implies transporting ellipsoids was noticed in [7,8] that advocates using the Levi-Civita connection for transportation. Again, the obtained method is independent from the coordinate system but still dependent on the chosen metric. That said, in the framework of invariant Kalman filtering devoted to EKF design, we have long advocated that one should start from the system's *dynamics* to choose the parallelization (or connection), hence retraction, see [1]. Torsion free geometry as Levi-Civita connection does not necessarily lead to best performance. To this respect note that our online code lends itself to quick benchmarking of choices of retraction, as done in Figure 2.

[1]Generalizations to the Stiefel manifold $St(p, n)$, that is, a set of p orthonormal vectors of $\mathbb{R}^n$, and hence to the set of p-dimensional subspaces of $\mathbb{R}^n$ called the Grassmann manifold are then straightforward.

REFERENCES

[1] A. Barrau and S. Bonnabel, "Invariant Kalman Filtering," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, pp. 237–257, 2018.

[2] S. J. Julier and J. K. Uhlmann, "Unscented Filtering and Nonlinear Estimation," *Proceedings of the IEEE*, vol. 92, no. 3, pp. 401–422, 2004.

[3] S. J. Julier and J. K. Uhlmann, "A New Extension of the Kalman Filter to Nonlinear Systems," in *AeroSense'97*, pp. 182–193, 1997.

[4] 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.

[5] S. Hauberg, F. Lauze, and K. S. Pedersen, "Unscented Kalman Filtering on Riemannian Manifolds," *Journal of mathematical imaging and vision*, vol. 46, no. 1, pp. 103–120, 2013.

[6] H. M. Menegaz, J. Y. Ishihara, and H. T. Kussaba, "Unscented Kalman Filters for Riemannian State-Space Systems," *IEEE Transactions on Automatic Control*, vol. 64, no. 4, pp. 1487–1502, 2018.

[7] G. Loianno, M. Watterson, and V. Kumar, "Visual Inertial Odometry for Quadrotors on SE(3)," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1544–1551, 2016.

[8] J. Svacha, G. Loianno, and V. Kumar, "Inertial Yaw-Independent Velocity and Attitude Estimation for High-Speed Quadrotor Flight," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1109–1116, 2019.

[9] E. J. Lefferts, F. L. Markley, and M. D. Shuster, "Kalman Filtering for Spacecraft Attitude Estimation," *Journal of Guidance, Control, and Dynamics*, vol. 5, no. 5, pp. 417–429, 1982.

[10] J. R. Forbes, A. H. de Ruiter, and D. E. Zlotnik, "Continuous-time Norm-constrained Kalman Filtering," *Automatica*, vol. 50, no. 10, pp. 2546–2554, 2014.

[11] J. Sola, "Quaternion Kinematics for the Error-state KF," *LAAS-CNRS, France, Tech. Rep*, 2012.

[12] E. Kraft, "A Quaternion-based Unscented Kalman Filter for Orientation Tracking," in *6th International Conference on Information Fusion (FUSION)*, vol. 1, pp. 47–54, 2003.

[13] T. Lee, "Global Unscented Attitude Estimation via the Matrix Fisher Distributions on SO(3)," in *American Control Conference (ACC), 2016*, pp. 4942–4947, IEEE, 2016.

[14] J. L. Crassidis and F. L. Markley, "Unscented Filtering for Spacecraft Attitude Estimation," *Journal of guidance, control, and dynamics*, vol. 26, no. 4, pp. 536–542, 2003.

[15] S. Bonnabel, "Left-Invariant Rxtended Kalman Filter and Attitude Estimation," in *Annual Conference on Decision and Control (CDC)*, pp. 1027–1032, IEEE, 2007.

[16] A. Barrau and S. Bonnabel, "Intrinsic filtering on SO(3) with discrete-time observations," in *Annual Conference on Decision and Control (CDC)*, pp. 3255–3260, IEEE, 2013.

[17] G. Bourmaud, R. Mégret, A. Giremus, and Y. Berthoumieu, "Discrete Extended Kalman Filter on Lie Groups," in *21st European Signal Processing Conference (EUSIPCO 2013)*, pp. 1–5, IEEE, 2013.

[18] G. Bourmaud, R. Mégret, M. Arnaudon, and A. Giremus, "Continuous-Discrete Extended Kalman Filter on Matrix Lie Groups Using Concentrated Gaussian Distributions," *Journal of Mathematical Imaging and Vision*, vol. 51, no. 1, pp. 209–228, 2015.

[19] A. Barrau and S. Bonnabel, "The Invariant Extended Kalman Filter as a Stable Observer," *IEEE Transactions on Automatic Control*, vol. 62, no. 4, pp. 1797–1812, 2016.

[20] J. Bohn and A. K. Sanyal, "Unscented State Estimation for Rigid Body Motion on SE(3)," in *Annual Conference on Decision and Control (CDC)*, pp. 7498–7503, IEEE, 2012.

[21] J. Jan, A. K. Sanyal, and E. A. Butcher, "Unscented State Estimation for Rigid Body Attitude Motion with a Finite-time Stable Observer," in *Annual Conference on Decision and Control (CDC)*, pp. 4698–4703, IEEE, 2016.

[22] M. Brossard, S. Bonnabel, and J.-P. Condomines, "Unscented Kalman filtering on Lie groups," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2485–2491, IEEE, 2017.

[23] J. R. Forbes and D. E. Zlotnik, "Sigma Point Kalman filtering on Matrix Lie Groups Applied to the SLAM Problem," in *International Conference on Geometric Science of Information*, pp. 318–328, Springer, 2017.

[24] J. Solà, J. Deray, and D. Atchuthan, "A Micro Lie Theory for State Estimation in Robotics," *CoRR*, 2018.

[25] Y. Wang and G. S. Chirikjian, "Error Propagation on the Euclidean Group with Applications to Manipulator Kinematics," *IEEE Transactions on Robotics*, vol. 22, no. 4, pp. 591–602, 2006.

[26] W. Park, Y. Liu, Y. Zhou, M. Moses, G. S. Chirikjian, *et al.*, "Kinematic State Estimation and Motion Planning for Stochastic Nonholonomic Systems Using the Exponential Map," *Robotica*, vol. 26, no. 4, pp. 419–434, 2008.

[27] G. S. Chirikjian, *Stochastic Models, Information Theory, and Lie Groups, Volume 1: Classical Results and Geometric Methods*. Applied and numerical harmonic analysis, Birkhäuser, 2009.

[28] G. S. Chirikjian and M. Kobilarov, "Gaussian Approximation of Nonlinear Measurement Models on Lie Groups," in *Annual Conference on Decision and Control (CDC)*, pp. 6401–6406, IEEE, 2014.

[29] T. Barfoot, J. R. Forbes, and P. T. Furgale, "Pose Estimation Using Linearized Rotations and Quaternion Algebra," *Acta Astronautica*, vol. 68, no. 1-2, pp. 101–112, 2011.

[30] T. D. Barfoot and P. T. Furgale, "Associating Uncertainty with Three-Dimensional Poses for Use in Estimation Problems," *IEEE Transactions on Robotics*, vol. 30, no. 3, pp. 679–693, 2014.

[31] T. D. Barfoot, *State Estimation for Robotics*. Cambridge University Press, 2017.

[32] N. Ko, W. Youn, I. Choi, G. Song, and T. Kim, "Features of Invariant Extended Kalman Filter Applied to Unmanned Aerial Vehicle Navigation," *Sensors*, vol. 18, no. 9, p. 2855, 2018.

[33] R. Hartley, M. G. Jadidi, J. Grizzle, and R. M. Eustice, "Contact-Aided Invariant Extended Kalman Filtering for Legged Robot State Estimation," in *Proceedings of Robotics: Science and Systems*, 2018.

[34] N. Y. Ko, G. Song, W. Youn, I. H. Choi, and T. S. Kim, "Improvement of Extended Kalman Filter Using Invariant Extended Kalman Filter," in *18th International Conference on Control, Automation and Systems (ICCAS)*, pp. 948–950, IEEE, 2018.

[35] M. Wang and A. Tayebi, "A Globally Exponentially Stable Nonlinear Hybrid Observer for 3D Inertial Navigation," in *Annual Conference on Decision and Control (CDC)*, pp. 1367–1372, 2018.

[36] M. Brossard, S. Bonnabel, and A. Barrau, "Unscented Kalman Filter on Lie Groups for Visual Inertial Odometry," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 649–655, IEEE, 2018.

[37] K. Wu, T. Zhang, D. Su, S. Huang, and G. Dissanayake, "An invariant-EKF VINS Algorithm for Improving Consistency," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1578–1585, IEEE, 2017.

[38] S. Bonnabel, "Symmetries in Observer Design: Review of Some Recent Results and Applications to EKF-based SLAM," in *Robot Motion and Control*, pp. 3–15, Springer, 2012.

[39] A. Barrau and S. Bonnabel, "An EKF-SLAM Algorithm with Consistency Properties," *arXiv preprint arXiv:1510.06263*, 2015.

[40] M. Brossard, A. Barrau, and S. Bonnabel, "Exploiting Symmetries to Design EKFs with Consistency Properties for Navigation and SLAM," *IEEE Sensors Journal*, vol. 19, no. 4, pp. 1572–1579, 2018.

[41] S. Heo and C. G. Park, "Consistent EKF-based Visual-inertial Odometry on Matrix Lie Group," *IEEE Sensors Journal*, vol. 18, no. 9, pp. 3780–3788, 2018.

[42] S. Heo, J. H. Jung, and C. G. Park, "Consistent EKF-based Visual-inertial Navigation Using Points and Lines," *IEEE Sensors Journal*, vol. 18, no. 18, pp. 7638–7649, 2018.

[43] T. Zhang, K. Wu, J. Song, S. Huang, and G. Dissanayake, "Convergence and Consistency Analysis for a 3-D Invariant-EKF SLAM," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 733–740, 2017.

[44] J.-P. Condomines, C. Seren, and G. Hattenberger, "Nonlinear State

Estimation Using an Invariant Unscented Kalman Filter," in *Guidance, Navigation and Control Conference*, pp. 1–15, AIAA, 2013.

[45] J.-P. Condomines, C. Seren, and G. Hattenberger, "Pi-invariant Unscented Kalman Filter for Sensor Fusion," in *Annual Conference on Decision and Control (CDC)*, pp. 1035–1040, IEEE, 2014.

[46] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, 2009.

[47] A. Barrau, *Non-linear state error based extended Kalman filters with applications to navigation*. PhD thesis, Mines Paristech, 2015.

[48] R. Mahony and T. Hamel, "A Geometric Nonlinear Observer for Simultaneous Localisation and Mapping," in *Annual Conference on Decision and Control (CDC)*, pp. 2408–2415, IEEE, 2017.

[49] M. Wang and A. Tayebi, "Geometric Nonlinear Observer Design for SLAM on a Matrix Lie Group," in *Annual Conference on Decision and Control (CDC)*, pp. 1488–1493, IEEE, 2018.

[50] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision Meets Robotics: The KITTI Dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[51] B. Nisar, P. Foehn, D. Falanga, and D. Scaramuzza, "VIMO: Simultaneous Visual Inertial Model-Based Odometry and Force Estimation," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2785–2792, 2019.

[52] G. Brandl, "Sphinx documentation," *http://sphinx-doc. org/sphinx. pdf*, 2010.

[53] 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.

[54] P. Kotaru and K. Sreenath, "Variation Based Extended Kalman Filter on S2," in *European Control Conference (ECC)*, pp. 875–882, IEEE, 2019.