Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds

Christoph Hertzberg [a,b,*], René Wagner [c,b], Udo Frese [a,b,c], Lutz Schröder [c,b]

[a] SFB TR/8 Spatial Cognition, Universität Bremen, Postfach 330 440, 28334 Bremen, Germany
[b] Fachbereich 3 – Mathematik und Informatik, Universität Bremen, Postfach 330 440, 28334 Bremen, Germany
[c] Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI), Sichere Kognitive Systeme, Enrique-Schmidt-Str. 5, 28359 Bremen, Germany

ARTICLE INFO

ABSTRACT

Common estimation algorithms, such as least squares estimation or the Kalman filter, operate on a state in a state space $\mathcal{S}$ that is represented as a real-valued vector. However, for many quantities, most notably orientations in 3D, $\mathcal{S}$ is not a vector space, but a so-called manifold, i.e. it behaves like a vector space locally but has a more complex global topological structure. For integrating these quantities, several ad hoc approaches have been proposed.

Here, we present a principled solution to this problem where the structure of the manifold $\mathcal{S}$ is encapsulated by two operators, state displacement $\boxplus : \mathcal{S} \times \mathbb{R}^n \to \mathcal{S}$ and its inverse $\boxminus : \mathcal{S} \times \mathcal{S} \to \mathbb{R}^n$. These operators provide a local vector-space view $\delta \mapsto x \boxplus \delta$ around a given state x. Generic estimation algorithms can then work on the manifold $\mathcal{S}$ mainly by replacing $+/-$ with $\boxplus/\boxminus$ where appropriate. We analyze these operators axiomatically, and demonstrate their use in least-squares estimation and the Unscented Kalman Filter. Moreover, we exploit the idea of encapsulation from a software engineering perspective in the *Manifold Toolkit*, where the $\boxplus/\boxminus$ operators mediate between a "flat-vector" view for the generic algorithm and a "named-members" view for the problem specific functions.

© 2011 Elsevier B.V. All rights reserved.

1. Introduction

Sensor fusion is the process of combining information obtained from a variety of different sensors into a joint belief over the system state. In the design of a sensor fusion system, a key engineering task lies in finding a state representation that (a) adequately describes the relevant aspects of reality and is (b) compatible with the sensor fusion algorithm in the sense that the latter yields meaningful or even optimal results when operating on the state representation.

Satisfying both these goals at the same time has been a longstanding challenge. Standard sensor fusion algorithms typically operate on real valued vector state representations ($\mathbb{R}^n$) while mathematically sound representations often form more complex, non-Euclidean topological spaces. A very common example of this comes up, e.g. within the context of inertial navigation systems (INS) where a part of the state space is $SO(3)$, the group of orientations in $\mathbb{R}^3$. To estimate variables in $SO(3)$, there are generally two different approaches. The first uses a parametrization of minimal dimension, i.e. with three parameters (e.g. Euler angles), and operates on the parameters like on $\mathbb{R}^3$. This parametrization has singularities, i.e. a situation analogous to the well-known *gimbal lock* problem in gimballed INS [13] can occur where very large changes in the parametrization are required to represent small changes in the state space. Workarounds for this exist that try to avoid these parts of the state space, as was most prominently done in the guidance system of the Apollo Lunar Module [22], or switch between alternative orderings of the parametrization each of which exhibit singularities in different areas of the state space. The second alternative is to overparametrize states with a non-minimal representation such as unit quaternions or rotation matrices which are treated as $\mathbb{R}^4$ or $\mathbb{R}^{3 \times 3}$ respectively and re-normalized as needed [45,30]. This has other disadvantages such as redundant parameters or degenerated, non-normalized variables.

Both approaches require representation-specific modifications of the sensor fusion algorithm and tightly couple the state representation and the sensor fusion algorithm, which is then no longer a generic black box but needs to be adjusted for every new state representation.

Our approach is based on the observation that sensor fusion algorithms employ operations which are inherently local, i.e. they compare and modify state variables in a local neighborhood around some reference. We thus arrive at a generic solution that bridges the gap between the two goals above by viewing the state space as a manifold. Informally speaking, every point of a manifold has a neighborhood that can be mapped bi-directionally to $\mathbb{R}^n$. This

* Corresponding author at: Fachbereich 3 – Mathematik und Informatik, Universität Bremen, Postfach 330 440, 28334 Bremen, Germany.
  E-mail address: chtz@informatik.uni-bremen.de (C. Hertzberg).

enables us to use an arbitrary manifold $\mathcal{S}$ as the state representation while the sensor fusion algorithm only sees a locally mapped part of $\mathcal{S}$ in $\mathbb{R}^n$ at any point in time. For the unit sphere S^2 this is illustrated in Fig. 1.

We propose to implement the mapping by means of two encapsulation operators ⊞ ("boxplus") and ⊟ ("boxminus") where

$$\boxplus : \mathcal{S} \times \mathbb{R}^n \to \mathcal{S}, \qquad (1)$$

$$\boxminus : \mathcal{S} \times \mathcal{S} \to \mathbb{R}^n. \qquad (2)$$

Here, ⊞ takes a manifold state and a small change expressed in the mapped local neighborhood in $\mathbb{R}^n$ and applies this change to the state to yield a new, modified state. Conversely, ⊟ determines the mapped difference between two states.

The encapsulation operators capture an important duality: The generic sensor fusion algorithm uses ⊟ and ⊞ in place of the corresponding vector operations − and + to compare and to modify states, respectively, based on flattened perturbation vectors, and otherwise treats the state space as a black box. Problem-specific code such as measurement models, on the other hand, can work inside the black box, and use the most natural representation for the state space at hand. The operators ⊞ and ⊟ translate between these alternative views. We will later show how this can be modeled in an implementation framework such that manifold representations (with matching operators) even of very sophisticated compound states can be generated automatically from a set of manifold primitives (in our C++ and MATLAB implementation currently $\mathbb{R}^n$, $SO(2)$, $SO(3)$, and S^2).

This paper extends course material [11] and several master's theses [25,5,16,43].

It starts with a discussion of related work in Section 2. Section 3 then introduces the ⊞-method and 3D orientations as the most important application, and Section 4 lays out how least squares optimization and Kalman filtering can be modified to operate on these so-called ⊞-manifolds. Section 5 introduces the aforementioned software toolkit, and Section 6 shows practical experiments. Finally, the appendices prove the properties of ⊞-manifolds claimed in Section 3 and give ⊞-manifold representations of the most relevant manifolds $\mathbb{R}/2\pi\mathbb{Z}$, $SO(n)$, S^n and $\mathbb{P}^n$ along with proofs.

2. Related work

2.1. Ad hoc solutions

Several ad hoc methods are available to integrate manifolds into estimation algorithms working on $\mathbb{R}^n$ [37,42]. All of them have some drawbacks, as we now discuss using 3D orientations as a running example.

The most common workaround is to use a minimal parametrization (e.g. Euler angles) [37] [42, p. 6] and place the singularities

in some part of the workspace that is not used (e.g. facing 90° upwards). This, however, creates an unnecessary constraint between the application and the representation leading to a failure mode that is easily forgotten. If it is detected, it requires a recovery strategy, in the worst case manual intervention as in the aforementioned Apollo mission [22].

Alternatively, one can switch between several minimal parametrizations with different singularities. This works but is more complicated.

For overparametrizations (e.g. quaternions), a normalization constraint must be maintained (unit length), e.g. by normalizing after each step [45,30]. The step itself does not know about the constraint, and tries to improve the fit by violating the constraint, which is then undone by normalization. This kind of counteracting updates is inelegant and at least slows down convergence. Lagrange multipliers could be used to enforce the constraint exactly [42, p. 40] but lead to a more difficult equation system. Alternatively, one could add the normalization constraint as a "measurement" but this is only an approximation, since it would need to have uncertainty zero [42, p. 40].

Instead, one could allow non-normalized states and apply a normalization function every time before using them [27,42]. Then the normalization degree of freedom is redundant having no effect on the cost function and the algorithm does not try to change it. Some algorithms can handle this (e.g. Levenberg–Marquardt [35, Chapter 15]) but many (e.g. Gauss–Newton [35, Chapter 15]) fail due to singular equations. This problem can be solved by adding the normalization constraint as a "measurement" with some arbitrary uncertainty [42, p. 40]. Still, the estimated state has more dimensions than necessary and computation time is increased.

2.2. Reference state with perturbations

Yet another way is to set a reference state and work relative to it with a minimal parametrization [37,42, p. 6]. Whenever the parametrization becomes too large, the reference system is changed accordingly [1,7]. This is practically similar to the ⊞-method, where in $s \boxplus \delta$ the reference state is s and δ is the minimal parametrization.

In particular Ref. [7] has triggered our investigation, so we compare to their SPmap in detail in Appendix D. The vague idea of applying perturbations in a way more specific than by simply adding has been around in the literature for some time:

> "We write this operation [the state update] as $x \to x + \delta x$, even though it may involve considerable more than vector addition." Triggs et al. [42, p. 7]

Very recently, Strasdat et al. [39, §II.C] have summarized this technique for $SO(3)$ under the label Lie group/algebra representation. This means the state is viewed as an element of the Lie group $SO(3)$, i.e. an orthonormal matrix or quaternion but every step of the estimation algorithm operates in the Lie group's tangential space, i.e. the Lie algebra. The Lie's group's exponential maps a perturbation δ from the Lie algebra to the Lie group. For Lie groups this is equivalent to our approach, with $s \boxplus \delta = s \cdot \exp \delta$ (Section A.6).

Our contribution here compared to Refs. [7,42,39] is to embed this idea into a mathematical and algorithmic framework by means of an explicit axiomatization, where the operators ⊞ and ⊟ make it easy to adapt algorithms operating on $\mathbb{R}^n$. Moreover, our framework is more generic, being applicable also to manifolds which fail to be Lie groups, such as S^2.

In summary, the discussion shows the need for a principled solution that avoids singularities and needs neither normalization constraints nor redundant degrees of freedom in the estimation algorithm.
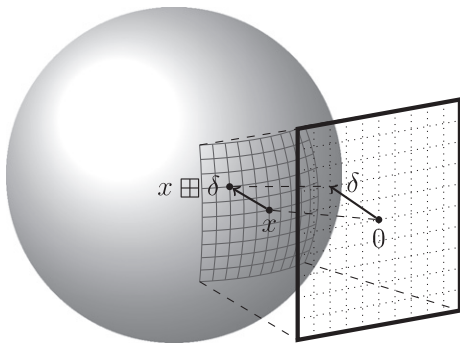

Fig. 1. Mapping a local neighborhood in the state space (here: on the unit sphere S^2) into $\mathbb{R}^n$ (here: the plane) allows for the use of standard sensor fusion algorithms without explicitly encoding the global topological structure.

2.3. Quaternion-based Unscented Kalman Filter

A reasonable ad hoc approach for handling quaternions in an EKF/UKF is to normalize, updating the covariance with the Jacobian of the normalization as in the EKF's dynamic step [8]. This is problematic, because it makes the covariance singular. Still the EKF operates as the innovation covariance remains positive definite. Nevertheless, it is unknown if this phenomenon causes problems, e.g. the zero-uncertainty degree of freedom could create overconfidence in another degree of freedom after a non-linear update.

With a similar motivation, van der Merwe [33] included a dynamic, $\dot{q} = \eta(1 - |q|^2)q$ that drives the quaternion q towards normalization. In this case the measurement update can still violate normalization, but the violation decays over time.

Kraft [24] and Sipos [38] propose a method of handling quaternions in the state space of an Unscented Kalman Filter (UKF). Basically, they modify the *unscented transform* to work with their specific quaternion state representation using a special operation for adding a perturbation to a quaternion and for taking the difference of two quaternions.

Our approach is related to Kraft's work in so far as we perform the same computations for states containing quaternions. It is more general in that we work in a mathematical framework where the special operations are encapsulated in the $\boxplus/\boxminus$-operators. This allows us to handle not just quaternions, but general manifolds as representations of both states and measurements, and to routinely adapt estimation algorithms to manifolds without sacrificing their genericity.

2.4. Distributions on unit spheres

The above-mentioned methods as well as our present work treat manifolds locally as $\mathbb{R}^n$ and take care to accumulate small steps in a way that respects the global manifold structure. This is an approximation as most distributions, e.g. Gaussians, are strictly speaking not local but extend to infinity. In view of this problem, several generalizations of normal distributions have been proposed that are genuinely defined on a unit sphere.

The von Mises–Fisher distribution on S^{n-1} was proposed by Fisher [10] and is given by

$$p(x) = C_n(\kappa) \exp(\kappa \mu^\top x), \tag{3}$$

with $\mu, x \in S^{n-1} \subset \mathbb{R}^n$, $\kappa > 0$ and a normalization constant $C_n(\kappa)$. μ is called the *mean direction* and κ the *concentration parameter*.

For the unit-circle S^1 this reduces to the von Mises distribution

$$p(x) = C_2(\kappa) \exp(\kappa \cos(x - \mu)), \tag{4}$$

with $x, \mu \in [-\pi, \pi)$.

The von Mises–Fisher distribution locally looks like a normal distribution $\mathcal{N}(0, \frac{1}{\kappa} I_{n-1})$ (viewed in the tangential space), where I_{n-1} is the $(n - 1)$-dimensional unit matrix. As only a single parameter exists to model the covariance, only isotropic distributions can be modeled, i.e. contours of constant probability are always circular. Especially for posterior distributions arising in sensor fusion this is usually not the case.

To solve this, i.e. to represent general multivariate normal distributions, Kent proposed the Fisher–Bingham distribution on the sphere [21]. It is defined as:

$$p(x) = \frac{1}{c(\kappa, \beta)} \exp\left(\kappa \gamma_1^\top x + \beta \left[(\gamma_2^\top x)^2 - (\gamma_3^\top x)^2 \right] \right).$$

It requires γ_i to be a system of orthogonal unit vectors, with γ_1 denoting the mean direction (as μ in the von Mises–Fisher case), γ_2 and γ_3 describing the semi-major and semi-minor axis of the

covariance. κ and β describe the concentration and eccentricity of the covariance.

Though being mathematically more profound, these distributions have the big disadvantage that they cannot be easily combined with other normal distributions. Presumably, a completely new estimation algorithm would be needed to compensate for this, whereas our approach allows us to adapt established algorithms.

3. The $\boxplus$-method

In this paper, we propose a method, the $\boxplus$-method, which integrates generic sensor fusion algorithms with sophisticated state representations by encapsulating the state representation structure in an operator $\boxplus$.

In the context of sensor fusion, a *state* refers to those aspects of reality that are to be estimated. The state comprises the quantities needed as output for a specific application, such as the guidance of an aircraft. Often, additional quantities are needed in the state in order to model the behavior of the system by mathematical equations, e.g. velocity as the change in position.

The first step in the design of a sensor fusion system (cf. Fig. 2) is to come up with a state model based on abstract mathematical concepts such as position or orientation in three-dimensional space. In the aircraft example, the INS state would consist of three components: the position ($\mathbb{R}^3$), orientation ($SO(3)$), and velocity ($\mathbb{R}^3$), each in some Earth-centric coordinate system, i.e.

$$S = \mathbb{R}^3 \times SO(3) \times \mathbb{R}^3. \tag{5}$$

3.1. The standard approach to state representation in generic sensor fusion algorithms

The second step (cf. Fig. 2) is to turn the abstract state model into a concrete representation which is suitable for the sensor fusion algorithm to work with. Standard generic sensor fusion algorithms require the state representation to be $\mathbb{R}^n$. Thus, one needs to translate the state model into an $\mathbb{R}^n$ representation. In the INS example, choosing Euler angles as the representation of $SO(3)$ we obtain the following translation:

$$\mathbb{R}^3 \times SO(3) \times \mathbb{R}^3 \to \mathbb{R}^9. \tag{6}$$

This translation step *loses information* in two ways: Firstly and most importantly, it forgets the mathematical structure of the state space – as discussed in the introduction, the Euler-angle representation of three-dimensional orientations has singularities and there are multiple parametrizations which correspond to the same orientation. This is particularly problematic, because the generic sensor fusion algorithm will exhibit erratic behavior near the singularities of this representation. If, instead, a non-minimal state representation such as quaternions or rotation matrices was used, the translation loses
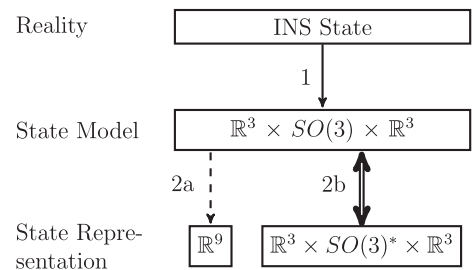

Fig. 2. From *reality* to *state representation* in the INS example (see text for details). Step 1 models relevant aspects of reality in the *state model*. Step 2a uses the standard approach to obtain a *lossy state representation*. Step 2b uses the $\boxplus$-method to obtain a *state representation preserving the topological structure* of the state space (using a faithful representation $SO(3)^*$ of $SO(3)$).

no information. However, later a generic algorithm would give inconsistent results, not knowing about the unit and orthonormality constraints, unless it was modified in a representation-specific way accordingly.

The second issue is that states are now treated as flat vectors with the only differentiation between different components being the respective index in a vector. All entries in the vector look the same, and for each original component of a state (e.g. position, orientation, velocity) one needs to keep track of the chosen representation and the correct index. This may seem like a mere book-keeping issue but in practice tends to be a particularly cumbersome and error-prone task when implementing process or measurement models that need to know about these representation-specifics.

3.2. Manifolds as state representations

We propose to use manifolds as a central tool to solve both issues. In this section, we regard a manifold as a black box encapsulating a certain (topological) structure. In practice it will be a *subset of $\mathbb{R}^s$ subject to constraints* such as the orthonormality of a 3×3 rotation matrix that can be used as a representation of $SO(3)$.

As for the representation of states, in the simple case where a state consists of a single component only, e.g. a three-dimensional orientation ($SO(3)$), we can represent a state as a single manifold. If a state consists of multiple different components we can also represent it as a manifold since the Cartesian product (Section A.7) of the manifolds representing each individual component yields another, *compound manifold*. Essentially, we can build sophisticated compound manifolds starting with a set of *manifold primitives*. As we will discuss in Section 5, this mechanism can be used as the basis of an object-oriented software toolkit which automatically generates compound manifold classes from a concise specification.

3.3. ⊞-Manifolds

The second important property of manifolds is that they are locally homeomorphic to $\mathbb{R}^n$, i.e. informally speaking, we can establish a bi-directional mapping from a local neighborhood in an n-manifold to $\mathbb{R}^n$. The ⊞-method uses two encapsulation operators ⊞ ("boxplus") and ⊟ ("boxminus") to implement this mapping:

$$\boxplus_S : S \times \mathbb{R}^n \to S, \tag{7}$$
$$\boxminus_S : S \times S \to \mathbb{R}^n. \tag{8}$$

When clear from the context, the subscript S is omitted. The operation $y = x \boxplus \delta$ adds a small perturbation expressed as a vector $\delta \in \mathbb{R}^n$ to the state $x \in S$. Conversely, $\delta = y \boxminus x$ determines the perturbation vector δ which yields y when ⊞-added to x. Axiomatically, this is captured by the definition below. We will discuss its properties here intuitively, formal proofs are given in Appendix A.

Definition 1 (⊞-*Manifold*). A ⊞-*manifold* is a quadruple $(S, \boxplus, \boxminus, V)$ (usually referred to as just S), consisting of a subset $S \subset \mathbb{R}^s$, operators

$$\boxplus : S \times \mathbb{R}^n \to S, \tag{9}$$
$$\boxminus : S \times S \to \mathbb{R}^n, \tag{10}$$

and an open neighborhood $V \subset \mathbb{R}^n$ of 0. These data are subject to the following requirements. To begin, $\delta \mapsto x \boxplus \delta$ must be smooth on $\mathbb{R}^n$, and for all $x \in S$, $y \mapsto y \boxminus x$ must be smooth on U_x, where $U_x = x \boxplus V$. Moreover, we impose the following axioms to hold for every $x \in S$:

$$x \boxplus 0 = x, \tag{11a}$$
$$\forall y \in S : \quad x \boxplus (y \boxminus x) = y, \tag{11b}$$
$$\forall \delta \in V : \quad (x \boxplus \delta) \boxminus x = \delta, \tag{11c}$$

$$\forall \delta_1, \delta_2 \in \mathbb{R}^n : \quad \|(x \boxplus \delta_1) \boxminus (x \boxplus \delta_2)\| \leqslant \|\delta_1 - \delta_2\|. \tag{11d}$$

One can show (Section A.1) that a ⊞-manifold is indeed a manifold, with additional structure useful for sensor fusion algorithms. The operators ⊞ and ⊟ allow a generic algorithm to modify and compare manifold states as if they were flat vectors without knowing the internal structure of the manifold, which thus appears as a black box to the algorithm.

Axiom (11a) makes 0 the neutral element of ⊞. Axiom (11b) ensures that from an element x, every other element $y \in S$ can be reached via ⊞, thus making $\delta \mapsto x \boxplus \delta$ surjective. Axiom (11c) makes $\delta \mapsto x \boxplus \delta$ injective on V, which defines the range of perturbations for which the parametrization by ⊞ is unique. Obviously, this axiom cannot hold globally in general, since otherwise we could have used $\mathbb{R}^n$ as a universal state representation in the first place. Instead, ⊞ and ⊟ create a local vectorized view of the state space. Intuitively x is a reference point which defines the "center" of a local neighborhood in the manifold and thus also the coordinate system of δ in the part of $\mathbb{R}^n$ onto which the local neighborhood in the manifold is mapped (cf. Fig. 1). Axiom (11d) allows to define a metric and is discussed later.

Additionally, we demand that the operators are *smooth* (i.e. sufficiently often differentiable, cf. Appendix A) in δ and y (for $y \in U_x$). This makes limits and derivatives of δ correspond to limits and derivatives of $x \boxplus \delta$, essential for any estimation algorithm (formally, $\delta \mapsto x \boxplus \delta$ is a diffeomorphism from V to U_x). It is important to note here that we require neither $x \boxplus \delta$ nor $y \boxminus x$ to be smooth in x. Indeed it is sometimes impossible for these expressions to be even continuous in x for all x (see Appendix B.5).

Returning to the INS example, we can now essentially keep the state model as the state representation (Fig. 2, Step 2b):

$$\mathbb{R}^3 \times SO(3) \times \mathbb{R}^3 \to \mathbb{R}^3 \times SO(3)^* \times \mathbb{R}^3,$$

where $SO(3)^*$ refers to any mathematically sound ("lossless") representation of $SO(3)$ expressed as a set of numbers to enable a computer to process it. Commonly used examples would be quaternions ($\mathbb{R}^4$ with unit constraints) or rotation matrices ($\mathbb{R}^{3 \times 3}$ with orthonormality constraints). Additionally, we need to define matching representation-specific ⊞ and ⊟ operators which replace the static, lossy translation of the state model into an $\mathbb{R}^n$ state representation that we saw in the standard approach above with an on-demand, lossless mapping of a manifold state representation into $\mathbb{R}^n$ in our approach.

In the INS example, ⊞ would simply perform vector-arithmetic on the $\mathbb{R}^n$ components and multiply a small, minimally parametrized rotation into the $SO(3)^*$ component (details follow soon).

3.4. Probability distributions on ⊞-manifolds

So far we have developed a new way of representing states – as compound manifolds – and a method that allows generic sensor fusion algorithms to work with them – the encapsulation operators ⊞/⊟. Both together form a ⊞-manifold. However, sensor fusion algorithms rely on the use of probability distributions to represent uncertain and noisy sensor data. Thus, we will now define probability distributions on ⊞-manifolds.

The general idea is to use a manifold element as the mean μ which defines a reference point. A multivariate probability distribution which is well-defined on $\mathbb{R}^n$ is then lifted into the manifold by mapping it into the neighborhood around $\mu \in S$ via ⊞. That is, for $X : \Omega \to \mathbb{R}^n$ and $\mu \in S$ (with $\dim S = n$), we can define $Y : \Omega \to S$ as $Y := \mu \boxplus X$, with probability distribution given by

$$p(Y = y) = p(\mu \boxplus X = y). \tag{12}$$

In particular, we extend the notion of a Gaussian distribution to ⊞-manifolds by

$$\mathcal{N}(\mu, \Sigma) := \mu \boxplus \mathcal{N}(0, \Sigma), \tag{13}$$

where $\mu \in \mathcal{S}$ is an element of the $\boxplus$-manifold but $\Sigma \in \mathbb{R}^{n \times n}$ just a matrix as for regular Gaussians (Appendix A.9).

3.5. Mean and covariance on $\boxplus$-manifolds

Defining the expected value on a manifold is slightly more involved than one might assume: we would, of course, expect that $EX \in \mathcal{S}$ for $X : \Omega \to \mathcal{S}$, which however would fail for a naive definition such as

$$EX \overset{?}{=} \int_{\mathcal{S}} x \cdot p(X = x) dx. \tag{14}$$

Instead, we need a definition that is equivalent to the definition on $\mathbb{R}^n$ and well-defined for $\boxplus$-manifolds. Therefore, we define the expected value as the value minimizing the expected mean squared error:

$$EX = \mathrm{argmin}_{x \in \mathcal{S}} E(\|X \boxminus x\|^2). \tag{15}$$

This also implies the implicit definition

$$E(X \boxminus EX) = 0 \tag{16}$$

as we will prove in Appendix A.9.

One method to compute this value is to start with an initial guess μ_0 and iterate [24]:

$$\mu_{k+1} = \mu_k \boxplus E(X \boxminus \mu_k), \tag{17}$$
$$EX = \lim_{k \to \infty} \mu_k. \tag{18}$$

Care must be taken that μ_0 is sufficiently close to the true expected value. In practice, however, this is usually not a problem as sensor fusion algorithms typically modify probability distributions only slightly at each time step such that the previous mean can be chosen as μ_0.

Also, closed form solutions exist for some manifolds – most trivially for $\mathbb{R}^n$. For rotation matrices, Markley et al. [31] give a definition similar to (15) but use Frobenius distance. They derive an analytical solution. Lemma 9 shows that both definitions are roughly equivalent, so this can be used to compute an initial guess for rotations.

The same method can be applied to calculate a weighted mean value of a number of values, because in vector spaces the weighted mean

$$\bar{x} = \sum_i w_i x_i \quad \text{with} \quad \sum_i w_i = 1 \tag{19}$$

can be seen as the expected value of a discrete distribution with $P(X = x_i) = w_i$.

The definition of the covariance of a $\boxplus$-manifold distribution, on the other hand, is straight-forward. As in the $\mathbb{R}^n$ case, it is an $n \times n$ matrix. Specifically, given a mean value $E X$ of a distribution $X : \Omega \to \mathcal{S}$, we define its covariance as

$$\mathrm{Cov}\, X = E((X \boxminus EX)(X \boxminus EX)^\top), \tag{20}$$

because $X \boxminus EX \in \mathbb{R}^n$ and the standard definition can be applied.

The $\boxminus$-operator induces a metric (proof in Lemma 2)

$$d(x, y) := \|y \boxminus x\|, \tag{21}$$

that is important in interpreting $\mathrm{Cov}\, X$. First,

$$\mathrm{tr}\, \mathrm{Cov}\, X = E(d(X, EX)^2). \tag{22}$$

i.e. $\sqrt{\mathrm{tr}\, \mathrm{Cov}\, X}$ is the rms d-distance of X to the mean. Second, the states $y \in \mathcal{S}$ with $d(\mu, y) = \sigma$ are the 1σ contour of $\mathcal{N}(\mu, \sigma^2 I)$. Hence, to interpret the state covariance or to define measurement or dynamic noise for a sensor fusion algorithm intuitively it is impor-

tant that the metric induced by $\boxminus$ has an intuitive meaning. For the $\boxplus$-manifold representing orientation in the INS example, $d(x, y)$ will be the angle between two orientations x and y.

In the light of (21), axiom (11d) means that the actual distance $d(x \boxplus \delta_1, x \boxplus \delta_2)$ is less or equal to the distance $\|\delta_1 - \delta_2\|$ in the parametrization (Fig. 3), i.e. the map $x \mapsto x \boxplus \delta$ is 1-Lipschitz. This axiom is needed for all concepts explained in this subsection, as can be seen in the proofs in Appendix A. In so far, (11d) is the deepest insight in the axiomatization (11).

Appendix A also discusses a slight inconsistency in the definitions above, where the covariance of $\mathcal{N}(\mu, \Sigma)$ defined by (13) and (20) is slightly smaller than Σ. For the usual case that Σ is significantly smaller than the range of unique parameters V (i.e. for angles $\ll \pi$), these inconsistencies are very small and can be practically ignored.

3.6. Practically important $\boxplus$-manifolds

We will now define $\boxplus$-operators for the practically most relevant manifolds: $\mathbb{R}^n$ as well as 2D and 3D orientations. Appendix B proves the ones listed here to fulfill the Axioms (11), Appendices A.5 and A.6 derive general techniques for constructing a $\boxplus$-operator.

3.6.1. Vectorspace $\mathbb{R}^n$

For $\mathcal{S} = \mathbb{R}^n$, the $\boxplus$- and $\boxminus$- operators are, of course, simple vector addition and subtraction

$$x \boxplus \delta = x + \delta, \quad y \boxminus x = y - x. \tag{23}$$

3.6.2. 2D orientation as angles modulo 2π

A planar rotation is the simplest case that requires taking its manifold structure into account. It can be represented by the rotation angle interpreted modulo 2π. Mathematically, this is $\mathcal{S} = \mathbb{R}/2\pi\mathbb{Z}$, a set of equivalence classes. Practically, simply a real number $\alpha \in \mathbb{R}$ is stored, and periodic equivalents $\alpha + 2\pi k$, $k \in \mathbb{Z}$ are treated as the same. $\boxplus$ is, then, simply plus. It could be normalized but that is not necessary. The difference in $\boxminus$ however, being a plain real value, must be normalized to $[-\pi, \pi)$ using a function v_π:

$$\alpha \boxplus \delta = \alpha + \delta, \quad \beta \boxminus \alpha = v_\pi(\beta - \alpha),$$
$$\text{where } v_\pi(\delta) := \delta - 2\pi \left\lfloor \frac{\delta + \pi}{2\pi} \right\rfloor. \tag{24}$$

With this definition, $\beta \boxminus \alpha$ is the smallest angle needed to rotate α into β, respecting the periodic interpretation of angles and giving the induced metric $d(x, y)$ an intuitive meaning. The parametrization is unique for angles of modulus $< \pi$, i.e. $V = (-\pi, \pi)$.
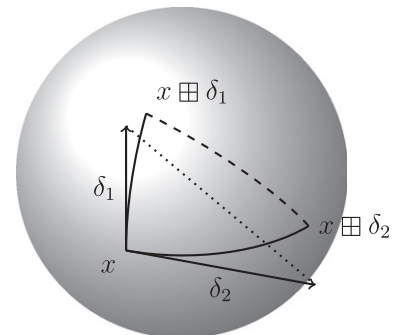

Fig. 3. Axiom (11d): The d-distance between $x \boxplus \delta_1$ and $x \boxplus \delta_2$ (dashed line) is less or equal to the distance in the parametrized space around x (dotted line).

3.6.3. 3D orientation as an orthonormal matrix

Rotations in 3D can be readily represented using orthonormal matrices $S \subset \mathbb{R}^{3 \times 3}$ with determinant 1. $x \boxplus \delta$ performs a rotation around axis δ in x coordinates with angle $\|\delta\|$. This is also called matrix exponential representation and implemented by the Rodriguez formula [23, pp. 147]

$$x \boxplus \delta = x \exp \delta, \quad y \boxminus x = \log(x^{-1} y), \tag{25}$$

$$\exp \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \cos\theta + cx^2 & -sz + cxy & sy + cxz \\ sz + cxy & \cos\theta + cy^2 & -sx + cyz \\ -sy + cxz & sx + cyz & \cos\theta + cz^2 \end{bmatrix}, \tag{26}$$

$$\theta = \sqrt{x^2 + y^2 + z^2}, \quad s = \operatorname{sinc}\theta, \quad c = \frac{1 - \cos\theta}{\theta^2},$$

$$\log x = \frac{1}{2\operatorname{sinc}\theta} \begin{bmatrix} x_{32} - x_{23} \\ x_{13} - x_{31} \\ x_{21} - x_{12} \end{bmatrix}, \qquad \theta = \operatorname{acos}\frac{\operatorname{tr} x - 1}{2}. \tag{27}$$

The induced metric $d(x, y)$ is the angle of a rotation necessary to rotate x onto y. There is also a monotonic relation to the widely used Frobenius distance $\|x - y\|_F$ (Lemma 9, Appendix C). The parametrization is again unique for angles $< \pi$, i.e. $V = B_\pi(0)$, where, as usual, we denote by

$$B_\varepsilon(v) = \{w \in \mathbb{R}^n \mid \|w - v\| < \varepsilon\} \tag{28}$$

the open ε-ball around $v \in \mathbb{R}^n$ for $\varepsilon > 0$.

3.6.4. 3D Orientation as a Unit Quaternion

The same geometrical construction of rotating around δ by $\|\delta\|$ also works with unit quaternions $S \subset \mathbb{H}$, where q and $-q$ are considered equivalent as they represent the same orientation.

$$q \boxplus \delta = q \cdot \exp\frac{\delta}{2}, \quad q \boxminus p = 2\overline{\log}(p^{-1} \cdot q) \tag{29}$$

$$\exp \delta = \begin{bmatrix} \cos\|\delta\| \\ \operatorname{sinc}\|\delta\|\delta \end{bmatrix} \tag{30}$$

$$\overline{\log}[wv] = \begin{cases} 0 & v = 0 \\ \frac{\operatorname{atan}(\|v\|/w)}{\|v\|} v & v \neq 0, \quad w \neq 0 \\ \pm\frac{\pi/2}{\|v\|} v & w = 0 \end{cases} \tag{31}$$

The factor 2 is introduced so that the induced metric $d(p, q)$ is the angle between two orientations and the quaternion and matrix $\boxplus$-manifolds are isomorphic. It originates from the fact that the quaternion is multiplied to the left and right of a vector when applying a rotation. To distinguish from the more general logarithm defined in Section B.2, we use the notation $\overline{\log}$ to express that $\overline{\log} q = \overline{\log}(-q)$.

3.6.5. Compound $\boxplus$-manifolds

Two (or several) $\boxplus$-manifolds S_1, S_2 can be combined into a single $\boxplus$-manifold by simply taking the Cartesian product $S = S_1 \times S_2$ and defining the $\boxplus$-and $\boxminus$-operator component-wise as

$$(x_1, x_2) \boxplus \begin{bmatrix} \delta_1 \\ \delta_2 \end{bmatrix} := (x_1 \boxplus_{S_1} \delta_1, x_2 \boxplus_{S_2} \delta_2) \tag{32}$$

$$(y_1, y_2) \boxminus (x_1, x_2) := \begin{bmatrix} y_1 \boxminus_{S_1} x_1 \\ y_2 \boxminus_{S_2} x_2 \end{bmatrix} \tag{33}$$

4. Least squares optimization and Kalman filtering on $\boxplus$-manifolds

One of the goals of the $\boxplus$-method was to easily adapt estimation algorithms to work on arbitrary manifolds. Essentially, this can be done by replacing + with $\boxplus$ when adding perturbations to a state, and replacing − with $\boxminus$ when comparing two states or measurements. However, some pitfalls may arise, which we will deal with in this section.

We show how the $\boxplus$-method can be applied to convert least squares optimization algorithms and the Unscented Kalman Filter such that they can operate on $\boxplus$-manifolds rather than just $\mathbb{R}^n$.

4.1. Least squares optimization

Least squares optimization dates back to the late 18th century where it was used to combine measurements in astronomy and in geodesy. Initial publications were made by Legendre [29] and Gauss [12] in the early 19th century. The method is commonly used to solve overdetermined problems, i.e. problems having more "equations" or measurements than unknown variables.

When combining all unknowns into a single state $X \in \mathbb{R}^n$, and all measurement functions into a single function $f : \mathbb{R}^n \to \mathbb{R}^m$, the basic idea is to find X such that given a combined measurement z,

$$\frac{1}{2}\|f(X) - z\|^2 = \min! \tag{38}$$

(where we write "=min!" to denote that the left-hand side becomes minimal). (38) assumes all measurement errors to be identically, normally distributed with identity covariance. For a positive definite covariance Σ between the measurements, this becomes

$$\frac{1}{2}\|f(X) - z\|_\Sigma^2 = \min! \tag{39}$$

using the notation $\|x\|_\Sigma^2 := x^\top \Sigma^{-1} x$. Under the assumption that $f(X) = z + \varepsilon$, with $\varepsilon \sim \mathcal{N}(0, \Sigma)$, this leads to a maximum likelihood solution.

If now our measurement function $f : S \to \mathcal{M}$ maps from a state manifold S to a measurement manifold $\mathcal{M}$, we can write analogously:

$$\frac{1}{2}\|f(X) \boxminus z\|_\Sigma^2 = \min! \tag{40}$$

which again leads to a maximum likelihood solution, for $f(X) \boxminus z \sim \mathcal{N}(0, \Sigma)$, as we will prove in Appendix A.9.

Even for classical least squares problems, non-linear functions f usually allow only local, iterative solutions, i.e. starting from an initial guess x_0 we construct a sequence of approximations x_i by calculating a refinement δ_i such that $x_{i+1} = x_i + \delta_i$ is a better solution than x_i.

This approach can be adapted to the manifold case, where every iteration takes place on a new local function

$$f_{x_i}^z : \mathbb{R}^n \to \mathbb{R}^m, \tag{41}$$

$$\delta \mapsto f(x_i \boxplus \delta) \boxminus z, \tag{42}$$

which for smooth f is a smooth function and, as it is an ordinary vector function, a local refinement δ_i can be found analogously to the classical case. This refinement can then be added to the previous state using $x_{i+1} := x_i \boxplus \delta_i$. The key difference is that now the refinements are accumulated in S, not in $\mathbb{R}^n$. In Table 1 we show how this can be done using the popular Gauss–Newton method with finite-difference Jacobian calculation. Other least squares methods like Levenberg–Marquardt can be applied analogously.

Table 1
Only small changes are necessary to adapt a classical least squares algorithm (left column) to work on $\boxplus$-manifolds (right column). Adding perturbations to the state is done using $\boxplus$, comparing values in the measurement space is done using $\boxminus$. Note that the $\boxminus z$ term does not cancel out when calculating the Jacobian. Also note that the equivalence marked by $*$ holds only approximately, as we will derive in Appendix A.9.

Classical Gauss–Newton	Gauss–Newton on a $\boxplus$-manifold	
$f : \mathbb{R}^n \to \mathbb{R}^m$	$f : \mathcal{S} \to \mathcal{M}$	(34)
$f(X) \sim \mathcal{N}(z, \Sigma) \Longleftrightarrow f(X) - z \sim \mathcal{N}(0, \Sigma)$	$f(X) \sim z \boxplus \mathcal{N}(0, \Sigma) \overset{*}{\Longleftrightarrow} f(X) \boxminus z \sim \mathcal{N}(0, \Sigma)$	(35)
Iterate with initial guess x_0 until x_i converges		
$J_{\bullet k} := \frac{f(x_i + \varepsilon e_k) - f(x_i - \varepsilon e_k)}{2\varepsilon}$	$J_{\bullet k} := \frac{(f(x_i \boxplus \varepsilon e_k) \boxminus z) - (f(x_i \boxplus -\varepsilon e_k) \boxminus z)}{2\varepsilon}$	(36)
$x_{i+1} := x_i - (J^\top \Sigma^{-1} J)^{-1} J^\top \Sigma^{-1} (f(x_i) - z)$	$x_{i+1} := x_i \boxplus - (J^\top \Sigma^{-1} J)^{-1} J^\top \Sigma^{-1} (f(x_i) \boxminus z)$	(37)

4.2. Kalman filtering

Since its inception in the late 1950s, the Kalman filter [20] and its many variants have successfully been applied to a wide variety of state estimation and control problems. In its original form, the Kalman filter provides a framework for continuous state and discrete time state estimation of linear Gaussian systems. Many real-world problems, however, are intrinsically non-linear, which gives rise to the idea of modifying the Kalman filter algorithm to work with non-linear process models (mapping old to new state) and measurement models (mapping state to expected measurements) as well.

The two most popular extensions of this kind are the Extended Kalman Filter (EKF) [4, Chapter 5.2] and more recently the Unscented Kalman Filter (UKF) [18]. The EKF linearizes the process and measurement models through first order Taylor series expansion. The UKF, on the other hand, is based on the *unscented transform* which approximates the respective probability distributions through deterministically chosen samples, so-called *sigma points*, propagates these directly through the non-linear process and measurement models and recovers the statistics of the transformed distribution from the transformed samples. Thus, intuitively, the EKF relates to the UKF as a tangent to a secant.

We will focus on the UKF here since it is generally better at handling non-linearities and does not require (explicit or numerically approximated) Jacobians of the process and measurement models, i.e. it is a *derivative-free* filter. Although the UKF is fairly new, it has been used successfully in a variety of robotics applications ranging from ground robots [41] to unmanned aerial vehicles (UAVs) [33].

The UKF algorithm has undergone an evolution from early publications on the *unscented transform* [36] to the work by Julier and Uhlmann [17,18] and by van der Merwe et al. [32,33]. The following is based on the consolidated UKF formulation by Thrun et al. [40] with parameters chosen as discussed in Ref. [43]. The modification of the UKF algorithm for use with manifolds is based on Refs. [11,25,5,43].

4.2.1. Non-linear process and measurement models

UKF process and measurement models need not be linear but are assumed to be subject to additive white Gaussian noise, i.e.

$$x_t = g(u_t, x_{t-1}) + \varepsilon_t, \tag{43}$$
$$z_t = h(x_t) + \delta_t, \tag{44}$$

where $g : T \times \mathbb{R}^n \to \mathbb{R}^n$ and $h : \mathbb{R}^n \to \mathbb{R}^m$ are arbitrary (but sufficiently nice) functions, T is the space of controls, $\varepsilon_t \sim \mathcal{N}(0, R_t)$, $\delta_t \sim \mathcal{N}(0, Q_t)$, and all ε_t and δ_t are independent.

4.2.2. Sigma points

The set of $2n + 1$ *sigma points* that are used to approximate an n-dimensional Gaussian distribution with mean μ and covariance Σ is computed as follows:

$$\mathcal{X}^{[0]} = \mu, \tag{45}$$
$$\mathcal{X}^{[i]} = \mu + \left(\sqrt{\Sigma}\right)_{\bullet i} \qquad \text{for } i = 1, \ldots, n, \tag{46}$$
$$\mathcal{X}^{[i+n]} = \mu - \left(\sqrt{\Sigma}\right)_{\bullet i} \qquad \text{for } i = 1, \ldots, n, \tag{47}$$

where $\left(\sqrt{\Sigma}\right)_{\bullet i}$ denotes the ith column of a matrix square root $\sqrt{\Sigma}\sqrt{\Sigma}^\top = \Sigma$ implemented by Cholesky decomposition. The name *sigma points* reflects the fact that all $\mathcal{X}^{[k]}$ lie on the 1σ-contour for $k > 0$.

In the following we will use the abbreviated notation

$$\mathcal{X} = (\mu \quad \mu + \sqrt{\Sigma} \quad \mu - \sqrt{\Sigma}) \tag{48}$$

to describe the generation of the *sigma points*.

4.2.3. Modifying the UKF algorithm for use with $\boxplus$-manifolds

The complete UKF algorithm is given in Table 2. Like other Bayes filter instances, it consists of two alternating steps – the prediction and the correction step. The prediction step of the UKF takes the previous belief represented by its mean μ_{t-1} and covariance Σ_{t-1} and a control u_t as input, calculates the corresponding set of *sigma points*, applies the process model to each *sigma point*, and recovers the statistics of the transformed distribution as the predicted belief with added process noise R_t ((52)–(55)).

To convert the prediction step of the UKF for use with $\boxplus$-manifolds we need to consider operations that deal with states. These add a perturbation vector to a state in (52), determine the difference between two states in (55), and calculate the mean of a set of sigma points in (54). In the manifold case, perturbation vectors are added via $\boxplus$ and the difference between two states is simply determined via $\boxminus$. The mean of a set of manifold *sigma points* can be computed analogously to the definition of the expected value from (17) and (18); the definition of the corresponding function MEANOFSIGMAPOINTS is shown in Table 3.

The UKF correction step first calculates the new set of *sigma points* (56), propagates each through the measurement model to obtain the *sigma points* corresponding to the expected measurement distribution in (57), and recovers its mean $\hat{z}_t$ in (58) and covariance S_t with added measurement noise Q_t in (59). Similarly, the cross-covariance $\overline{\Sigma}_t^{x,z}$ between state and expected measurement is calculated in (60). The latter two are then used in (61) to compute the Kalman gain K, which determines how the innovation is to be used to update the mean in (62) and how much uncertainty can be removed from the covariance matrix in (63) to reflect the information gained from the measurement.

The conversion of the UKF correction step for use with $\boxplus$-manifolds generally follows the same strategy as that of the prediction step but is more involved in detail. Firstly, this is because we use $\boxplus$-manifolds to represent both states and measurements so that the advantages introduced for states above also apply to measurements. Secondly, the update of the mean in (62) cannot be implemented as a simple application of $\boxplus$: This might result in an

Table 2
Classical UKF vs. UKF on ⊞-manifolds algorithms. See text for details.

Classical UKF	UKF on ⊞-Manifolds	

Input, Process and Measurement Models:

$$\mu_{t-1} \in \mathbb{R}^n, \Sigma_{t-1} \in \mathbb{R}^{n\times n}, u_t \in T, z_t \in \mathbb{R}^m \qquad \mu_{t-1} \in \mathcal{S}, \Sigma_{t-1} \in \mathbb{R}^{n\times n}, u_t \in T, z_t \in \mathcal{M} \qquad (49)$$

$$g: T\times\mathbb{R}^n \to \mathbb{R}^n, X_t = g(u_t, X_{t-1}) + \mathcal{N}(0, R_t) \qquad g: T\times\mathcal{S} \to \mathcal{S}, X_t = g(u_t, X_{t-1}) \boxplus \mathcal{N}(0, R_t) \qquad (50)$$

$$h: \mathbb{R}^n \to \mathbb{R}^m, z_t = h(X_t) + \mathcal{N}(0, Q_t) \qquad h: \mathcal{S} \to \mathcal{M}, z_t = h(X_t) \boxplus_{\mathcal{M}} \mathcal{N}(0, Q_t) \qquad (51)$$

Prediction Step:

$$\mathcal{X}_{t-1} = (\mu_{t-1} \;\; \mu_{t-1} + \sqrt{\Sigma_{t-1}} \;\; \mu_{t-1} - \sqrt{\Sigma_{t-1}}) \qquad \mathcal{X}_{t-1} = (\mu_{t-1} \;\; \mu_{t-1} \boxplus \sqrt{\Sigma_{t-1}} \;\; \mu_{t-1} \boxplus -\sqrt{\Sigma_{t-1}}) \quad (52)$$

$$\bar{\mathcal{X}}_t^* = g(u_t, \mathcal{X}_{t-1}) \qquad\qquad \bar{\mathcal{X}}_t^* = g(u_t, \mathcal{X}_{t-1}) \qquad (53)$$

$$\bar{\mu}_t = \frac{1}{2n+1}\sum_{i=0}^{2n} \bar{\mathcal{X}}_t^{*[i]} \qquad\qquad \bar{\mu}_t = \text{MeanOfSigmaPoints}(\bar{\mathcal{X}}_t^*) \qquad (54)$$

$$\bar{\Sigma}_t = \frac{1}{2}\sum_{i=0}^{2n} (\bar{\mathcal{X}}_t^{*[i]} - \bar{\mu}_t)(\bar{\mathcal{X}}_t^{*[i]} - \bar{\mu}_t)^\top + R_t \qquad \bar{\Sigma}_t = \frac{1}{2}\sum_{i=0}^{2n} (\bar{\mathcal{X}}_t^{*[i]} \boxminus \bar{\mu}_t)(\bar{\mathcal{X}}_t^{*[i]} \boxminus \bar{\mu}_t)^\top + R_t \qquad (55)$$

Correction Step:

$$\bar{\mathcal{X}}_t = (\bar{\mu}_t \;\; \bar{\mu}_t + \sqrt{\bar{\Sigma}_t} \;\; \bar{\mu}_t - \sqrt{\bar{\Sigma}_t}) \qquad \bar{\mathcal{X}}_t = (\bar{\mu}_t \;\; \bar{\mu}_t \boxplus \sqrt{\bar{\Sigma}_t} \;\; \bar{\mu}_t \boxplus -\sqrt{\bar{\Sigma}_t}) \qquad (56)$$

$$\bar{\mathcal{Z}}_t = h(\bar{\mathcal{X}}_t) \qquad\qquad \bar{\mathcal{Z}}_t = h(\bar{\mathcal{X}}_t) \qquad (57)$$

$$\hat{z}_t = \frac{1}{2n+1}\sum_{i=0}^{2n} \bar{\mathcal{Z}}_t^{[i]} \qquad\qquad \hat{z}_t = \text{MeanOfSigmaPoints}(\bar{\mathcal{Z}}_t) \qquad (58)$$

$$S_t = \frac{1}{2}\sum_{i=0}^{2n} (\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t)(\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t)^\top + Q_t \qquad S_t = \frac{1}{2}\sum_{i=0}^{2n} (\bar{\mathcal{Z}}_t^{[i]} \boxminus_{\mathcal{M}} \hat{z}_t)(\bar{\mathcal{Z}}_t^{[i]} \boxminus_{\mathcal{M}} \hat{z}_t)^\top + Q_t \qquad (59)$$

$$\bar{\Sigma}_t^{x,z} = \frac{1}{2}\sum_{i=0}^{2n} (\bar{\mathcal{X}}_t^{[i]} - \bar{\mu}_t)(\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t)^\top \qquad \bar{\Sigma}_t^{x,z} = \frac{1}{2}\sum_{i=0}^{2n} (\bar{\mathcal{X}}_t^{[i]} \boxminus \bar{\mu}_t)(\bar{\mathcal{Z}}_t^{[i]} \boxminus_{\mathcal{M}} \hat{z}_t)^\top \qquad (60)$$

$$K_t = \bar{\Sigma}_t^{x,z} S_t^{-1} \qquad\qquad K_t = \bar{\Sigma}_t^{x,z} S_t^{-1} \qquad (61)$$

$$\mu_t = \bar{\mu}_t + K_t(z_t - \hat{z}_t) \qquad\qquad \delta = K_t(z_t \boxminus_{\mathcal{M}} \hat{z}_t) \qquad (62)$$

$$\Sigma_t = \bar{\Sigma}_t - K_t S_t K_t^\top \qquad\qquad \Sigma_t' = \bar{\Sigma}_t - K_t S_t K_t^\top \qquad (63)$$

$$\mathcal{X}_t' = (\bar{\mu}_t \boxplus \delta \;\; \bar{\mu}_t \boxplus (\delta + \sqrt{\Sigma_t'}) \;\; \bar{\mu}_t \boxplus (\delta - \sqrt{\Sigma_t'})) \qquad (64)$$

$$\mu_t = \text{MeanOfSigmaPoints}(\mathcal{X}_t') \qquad (65)$$

$$\Sigma_t = \frac{1}{2}\sum_{i=0}^{2n} (\mathcal{X}_t'^{[i]} \boxminus \mu_t)(\mathcal{X}_t'^{[i]} \boxminus \mu_t)^\top \qquad (66)$$

inconsistency between the covariance matrix and the mean since in general

$$\mu \boxplus \mathcal{N}(\delta, \Sigma') \neq (\mu \boxplus \delta) \boxplus \mathcal{N}(0, \Sigma') \qquad (71)$$

i.e. the mean would be modified while the covariance is still formulated in terms of the coordinate system defined by the old mean as the reference point. Thus, we need to apply an additional *sigma point* propagation as follows. The manifold variant of (62) only determines the perturbation vector δ by which the mean is to be changed and the manifold variant of (63) calculates a temporary covariance matrix Σ_t' still relative to the old mean $\bar{\mu}_t$. (64) then adds the sum of δ and the respective columns of Σ_t' to $\bar{\mu}_t$ in a single ⊞ operation to generate the set of *sigma points* $\mathcal{X}_t'$. Therefrom, the new mean μ_t in (65) and covariance Σ_t in (66) is computed.

The overhead of the additional *sigma point* propagation can be avoided by storing $\mathcal{X}_t'$ for reuse in (52) or (56). If $x \boxplus \delta$ is continuous in x, the step can also be replaced by $\mu_t = \bar{\mu}_t \boxplus \delta$ as an approximation.

A final word auf caution: *Sigma point* propagation fails for a standard deviation larger than the range V of unique parametrization,

Table 3
MeanOfSigmaPoints computes the mean of a set of ⊞-manifold sigma points $\mathcal{Y}$. In practice, the limit in (70) can be implemented by an iterative loop that is terminated if the norm of the most recent summed error vector is below a certain threshold. The number of *sigma points* is not necessarily the same as the dimension of each.

⊞-Manifold-MeanOfSigmaPoints

Input:

$$\mathcal{Y}^{[i]}, \; i = 0, \dots, 2n \qquad (67)$$

Determine mean μ':

$$\mu_0' = \mathcal{Y}^{[0]} \qquad (68)$$

$$\mu_{k+1}' = \mu_k' \boxplus \frac{1}{2n+1}\sum_{i=0}^{2n} \mathcal{Y}^{[i]} \boxminus \mu_k' \qquad (69)$$

$$\mu' = \lim_{k\to\infty} \mu_k' \qquad (70)$$

where even propagation through the identity function results in a reduced covariance. To prevent this, the standard deviation must be within $V/2$, so all *sigma points* are mutually within a range of V. Hence, for 2D and 3D orientations, an angular standard deviation of $\pi/2$ is allowed. This is no practical limitation, because filters usually fail much earlier because of non-linearity.

5. ⊞-Manifolds as a software engineering tool

As discussed in Section 3, the ⊞-method simultaneously provides two alternative views of a ⊞-manifold. On the one hand, generic algorithms access *primitive* or *compound manifolds* via flattened perturbation vectors, on the other hand, the user implementing process and measurement models needs direct access to the underlying state representation (such as a quaternion) and for *compound manifolds* wants to access individual components by a descriptive name.

In this section we will use our Manifold Toolkit (*MTK*) to illustrate how the ⊞-method can be modeled in software. The current version of *MTK* is implemented in C++ and uses the Boost Preprocessor library [2] and the Eigen Matrix library [15]. A port to MATLAB is also available [44]. Similar mechanisms can be applied in other (object-oriented) programming languages.

5.1. Representing manifolds in software

In *MTK*, we represent ⊞-manifolds as C++ classes and require every manifold to provide a common interface to be accessed by the generic sensor fusion algorithm, i.e. implementations of ⊞ and ⊟. The corresponding C++ interface is fairly straight-forward. Defining a manifold requires an `enum DOF` and two methods

```
struct MyManifold {
  enum {DOF = n};
  typedef double scalar;
  void boxplus (vectview < const double, DOF > delta,
      double scale = 1);
  void boxminus (vectview < double, DOF > delta,
          const MyManifold & other) const;
};
```

where *n* is the degrees of freedom, `x.boxplus (delta, s)` implements $x := x \boxplus (s \cdot \delta)$ and `y.boxminus (delta, x)` implements $\delta := y \boxminus x$. `vectview` maps a `double` array of size `DOF` to an expres-
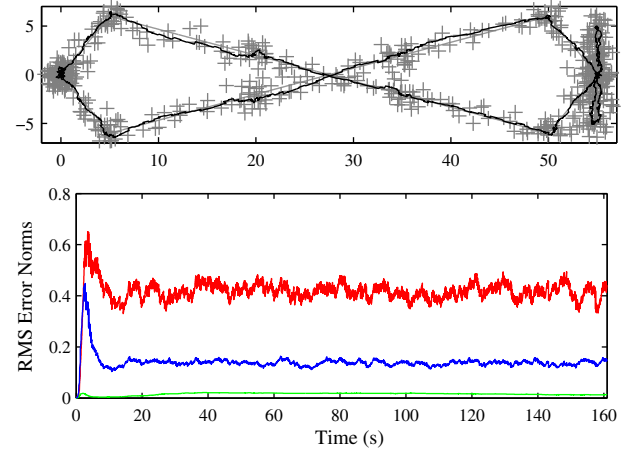

Fig. 5. **Top**: Trajectory estimated by the minimal INS-GPS filter (black) vs. ground truth (solid gray) and position measurements (gray crosses) from a single filter run. The plot shows the *x*–*y*-plane (m). **Bottom**: RMS error norms for position (red, m), orientation (green, rad) and velocity (blue, m/s) estimates from 50 Monte Carlo runs. The time averaged error is 0.415 m, 0.0158 rad and 0.141 m/s, respectively. This magnitude seems plausible for a σ_p = 0.75 m GPS. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

sion directly usable in Eigen expressions. The scaling factor in `boxplus` can be set to -1 to conveniently implement $x \boxplus (-\delta)$.

Additionally, a ⊞-manifold class can provide arbitrary member variables and methods (which, e.g. rotate vectors in the case of orientation) that are specific to the particular manifold.

MTK already comes with a library of readily available *manifold primitive* implementations of $\mathbb{R}^n$ as `vect < n>`, $SO(2)$ and $SO(3)$ as `SO2` and `SO3` respectively, and S^2 as `S2` It is possible to provide alternative implementations of these or to add new implementations of other manifolds basically by writing appropriate ⊞ and ⊟ methods.

5.2. Automatically generated compound manifold representations

In practice, a single *manifold primitive* is usually insufficient to represent states (or measurements). Thus, we also need to cover *compound ⊞-manifolds* consisting of several manifold components in software.

A user-friendly approach would encapsulate the manifold in a class with members for each individual submanifold which,
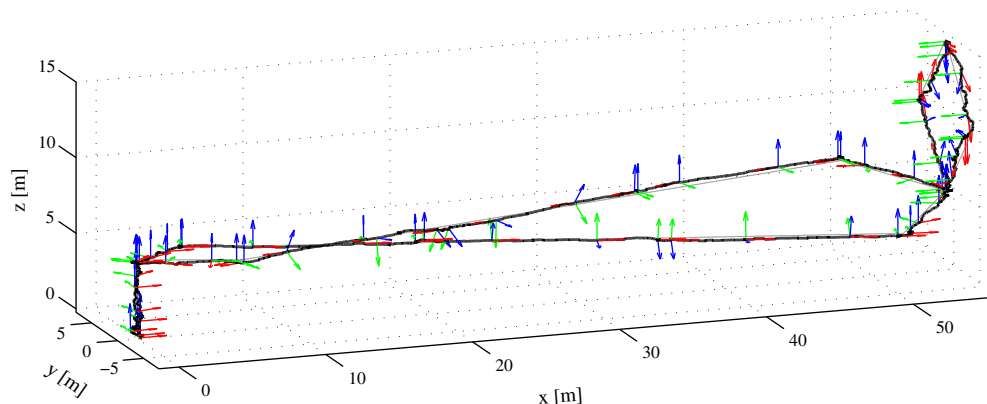

Fig. 4. Trajectory estimated from a synthetic helicopter dataset by the *UKFoM*-based minimal INS-GPS filter. The trajectory starts and ends at (0, 0) and consists of a stylized figure eight with an embedded stylized loop (right). While on the long segments of the figure eight the helicopter rotates about its roll axis.
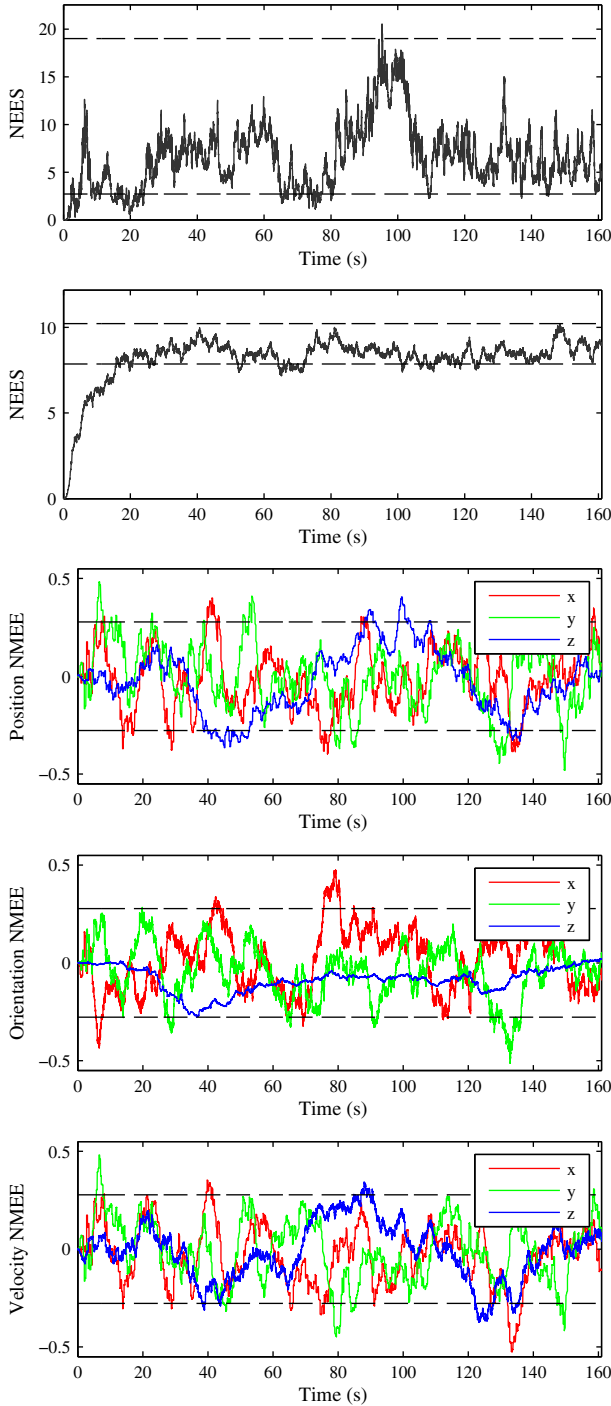
Fig. 7. Performance of a UKF using Euler angles for the orientation (top to bottom): Trajectory estimated, error norms as in Fig.5, NEES, from a single run. The filter operates normally until $t = 60$ s where the orientation approaches the singularity, the filter becomes inconsistent and the error rapidly increases with the estimate leaving the workspace.

a list of simple manifolds. The way how *MTK* does this automatically and hides all details from the user is a main contribution of *MTK*.

Returning to the INS example from the introduction where we need to represent a state consisting of a position, an orientation, and aP velocity, we would have a nine-degrees-of-freedom state

$$\mathcal{S} = \mathbb{R}^3 \times SO(3) \times \mathbb{R}^3. \tag{72}$$

Using our toolkit this can be constructed as:

```
MTK_BUILD_MANIFOLD (state,
  ((vect<3>, pos))
  ((SO3, orient))
  ((vect<3>, vel))
)
```

Given this code snippet the preprocessor will generate a `class state`, having public members `pos`, `orient`, and `vel`. Also generated are the manifold operations `boxplus` and `boxminus` as well as the total degrees of freedom `state::DOF = 9` of the *compound manifold*.

The macro also addresses another technical problem: Kalman filters in particular require covariance matrices to be specified which represent process and measurement noise. Similarly, individual parts of covariance matrices often need to be analyzed. In both cases the indices of individual components in the flattened vector view need to be known. *MTK* makes this possible by generating an enum `IDX` reflecting the index corresponding to the start of the respective part of the vector view. In the above example, the start of the vectorized orientation can be determined using e.g. `s.orient.IDX` for a `state s`. The size of this part is given by `s.orient.DOF`. *MTK* also provides convenience methods to access and modify corresponding sub-vectors or sub-matrices using member pointers.

Fig. 6. Plots for filter consistency evaluation (top to bottom): NEES $\|\mu_t \boxminus X_t\|_{\Sigma_t}^2$ from a single run; averaged NEES from 50 Monte Carlo runs; $\frac{(\mu_t \boxminus X_t)_k}{\sqrt{\Sigma_{tkk}}}$ averaged over 50 Monte Carlo runs (NMEE) with k being x, y, and z of position, orientation and velocity. Note how each largely remains within its 95% probability region (dashed), as should be the case for a consistent filter.

mathematically, corresponds to a Cartesian product. Following this approach, $\boxplus/\boxminus$ operators on the *compound manifold* are needed, which use the $\boxplus/\boxminus$ operators of the components as described in Section 3.6.5. Using this method, the user can access members of the manifold by name, and the algorithm just sees the compound manifold.

This can be done by hand in principle, but becomes quite error-prone when there are many components. Therefore, *MTK* provides a preprocessor macro which generates a compound manifold from
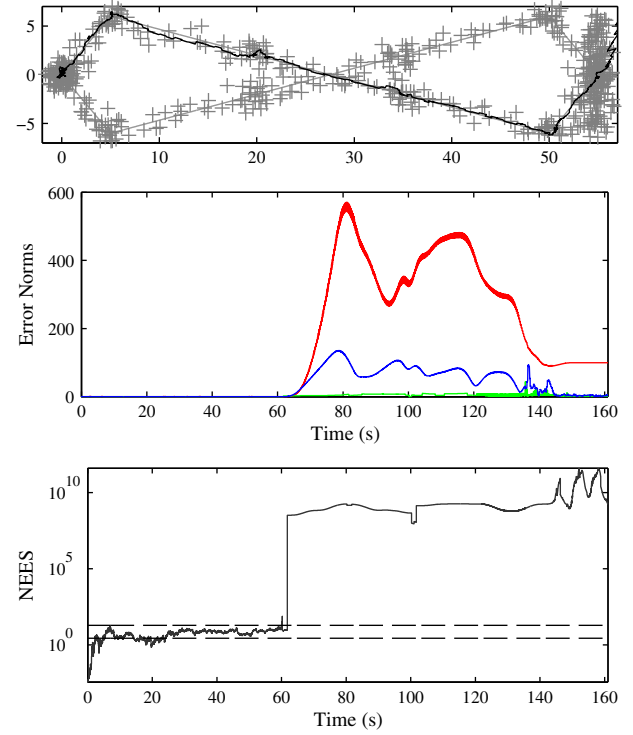
```
// state covariance matrix:
Matrix < double, state::DOF, state::DOF > cov;
cov.setZero ();

// set diagonal of covariance block of pos part:
setDiagonal (cov, & state::pos, 1);

// fill entire orient-pos-covariance block:
subblock (cov, & state::orient, & state::pos).fill
    (0.1);
```

5.3. Generic least squares optimization and UKF implementations

Based on *MTK*, we have developed a generic least squares optimization framework called *SLoM* (Sparse Least Squares on Manifolds) [16] according to Table 1 and *UKFoM* [43], a generic UKF on manifolds implementation according to Table 2. Apart from handling manifolds, *SLoM* automatically infers sparsity, i.e. which measurement depends on which variables, and exploits this in computing the Jacobian in (36), and in representing, and inverting it in (37).

MTK, *SLoM* and *UKFoM* are already available online on http://www.openslam.org/MTK.html under an open source license.

6. Experiments

We now illustrate the advantages of the ⊞-method in terms of both ease of use and algorithmic performance.

6.1. Worked example: INS-GPS integration

In a first experiment, we show how *MTK* and *UKFoM* can be used to implement a minimalistic, but working INS-GPS filter in about 50 lines of C++ code. The focus is on how the framework allows to concisely write down such a filter. It integrates accelerometer and gyroscope readings in the prediction step and fuses a global position measurement (e.g. loosely coupled GPS) in the measurement update. The first version assumes white noise. We then show how the framework allows for an easy extension to colored noise.

6.1.1. IMU process model

Reusing the state manifold definition from Section 5.2, the process model implements $x_t = g(u_t, x_{t-1})$ (cf. Section 4.2.1) where the control u_t in this case comprises acceleration a as measured by a three-axis accelerometer and angular velocity w as measured by a three-axis gyroscope.

```
state process_model (const state & s,
    const vect < 3> & a, const vect < 3 > & w)
{
    state s2;

    // apply rotation
    vect < 3 > scaled_axis = w * dt;
    SO3 rot = SO3::exp (scaled_axis);
    s2.orient = s.orient * rot;

    // accelerate with gravity
    s2.vel = s.vel + (s.orient * a + gravity) * dt;

    //translate
    s2.pos = s.pos + s.vel *dt;

    return s2;
}
```

The body of the process model uses Euler integration of the motion described by a and w over a short time interval dt. Note how individual components of the manifold state are accessed by name and how they can provide non-trivial methods (overloaded operators).

Further, we need to implement a function that returns the process noise term R_t.

```
ukf < state>::cov process_noise_cov ()
{
    ukf < state>::cov cov = ukf < state>::cov::Zero ();

    setDiagonal (cov, & state::pos, 0);
    setDiagonal (cov, & state::orient,gyro_noise*dt);
    setDiagonal (cov, & state::vel, acc_noise*dt);

    return cov;
}
```

Note how the MTK's setDiagonal function automatically fills in the diagonal entries of the covariance matrix such that their order matches the way the ⊟/⊞ operators locally vectorize the state space, i.e. the user does not need to know about these internals. The constants $\sigma_\omega^2 = $ gyro_noise and $\sigma_\nu^2 = $ acc_noise are continuous noise spectral densities for the gyroscope $\left(\frac{rad^2}{s}\right)$ and accelerometer $\left(\frac{m^2}{s^3}\right)$ and multiplied by dt in the process noise covariance matrix

$$R_t = dt \cdot \mathrm{diag}\left(0, 0, 0, \sigma_\omega^2, \sigma_\omega^2, \sigma_\omega^2, \sigma_\nu^2, \sigma_\nu^2, \sigma_\nu^2\right). \quad (73)$$

6.1.2. GPS measurement model

Measurement models implement $z_t = h(x_t)$ (cf. Section 4.2.1), in the case of a position measurement simply returning the position from x_t.

```
vect < 3 > gps_measurement_model (const state & s)
{
    return s.pos;
}
```

We also need to implement a function that returns the measurement noise term Q_t.

```
Matrix3x3 gps_measurement_noise_cov ()
{
    return gpos_noise* Matrix3x3::Identity ();
}
```

Again, $\sigma_p^2 = $ gps_noise is constant. Note that although we show a vector measurement in this example, *UKFoM* also supports manifold measurements.

6.1.3. Executing the filter

Executing the UKF is now straight-forward. We first instantiate the ukf template with our state type and pass a default $(0, 1, 0)$ initial state to the constructor along with an initial covariance matrix. We then assume that sensor data is acquired in some form of loop, and at each iteration execute the prediction and correction (update) steps with the process and measurement models and sensor readings as arguments.
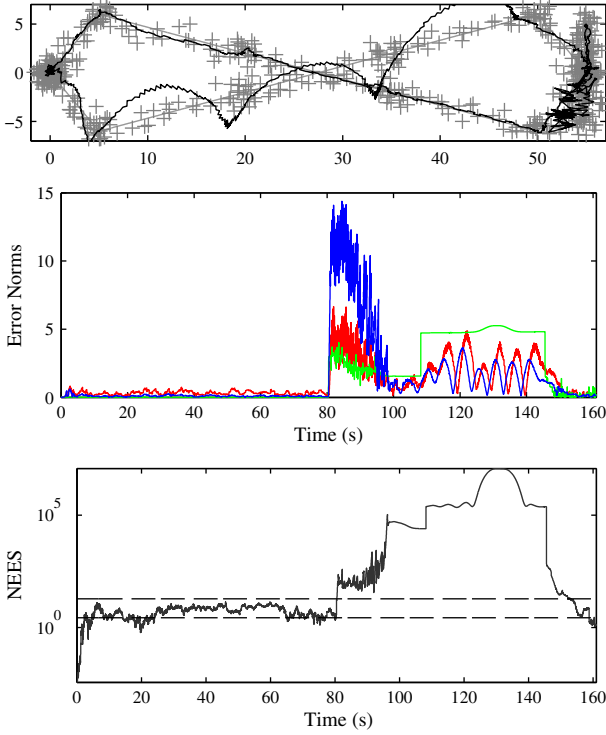
Solely based on accelerometer and gyroscope data the state estimate would drift over time. *GPS* measurements allow the filter to reset errors stemming from accumulated process noise. However, over short time periods accelerometer and gyroscope smooth out the noisy GPS measurements as illustrated in Fig. 5. As suggested by Ref. [4] we verify the filter consistency by computing the *normalized estimation error squared* (NEES) and the *normalized mean estimation error* (NMEE) for each state component (Fig. 6). Figs. 7 and 8 present the same results for a UKF using Euler angles and scaled axis respectively, showing the expected failure once the orientation approaches singularity. Fig. 9 shows the results for the technique proposed in Ref. [33] with a plain quaternion $\mathbb{R}^4$ in the UKF state and a process model that drives the quaternion towards normalization ($\eta = 0.1/s$, cf. Section 2.3). Overall, Euler-angle and scaled axis fail at singularities, the ⊞-method is very slightly better than the plain quaternion. The difference in performance is not very relevant, our claim is rather that the ⊞-method is conceptually more elegant. Computation time was 21/32 μs (⊞), 28/23 μs (Euler), 25/23 μs (scaled axis), 21/33 μs (quaternion) for a predict-step/for a GPS update. All timings were determined on an Intel Xeon CPU E5420 @2.50 GHz running 32 bit Linux.

6.2. Extension to colored noise errors

GPS errors are correlated and hence an INS-GPS filter should model colored not white noise. Therefore, a bias vector must be added to the state:
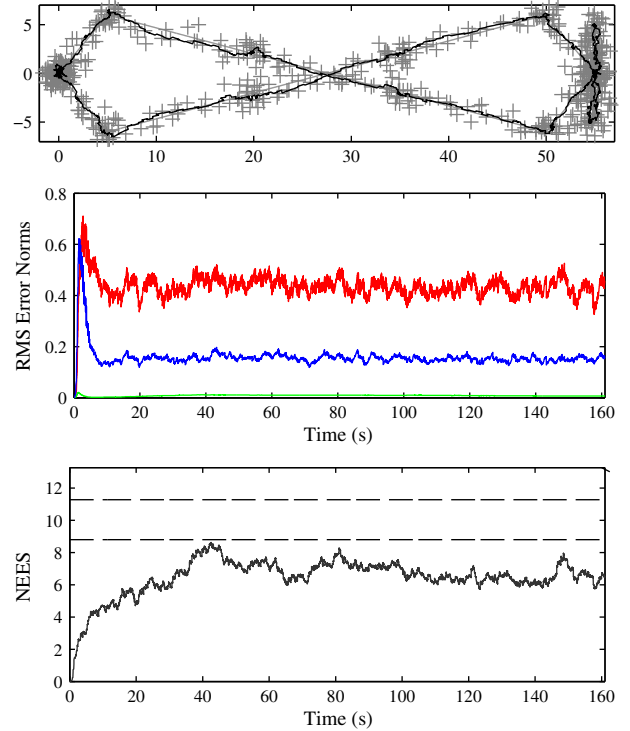
Fig. 8. Performance of a UKF using a scaled axis representation for the orientation (top to bottom): Trajectory estimated, error norms as in Fig. 5, NEES, from a single run. The filter operates normally until $t = 80$ s where the orientation approaches the singularity, the filter becomes inconsistent and the error rapidly increases. Surprisingly, the filter recovers in the end, indicating that scaled axis is a more robust representation than Euler angles.

```
ukf < state>::cov init_cov=
  0.001 * ukf < state>::cov::Identity ();
ukf < state > kf (state (), init_cov);
vect < 3 > acc, gyro, gps;
for (...)
{
  kf.predict (
    boost::bind (process_model, _1, acc, gyro),
    process_noise_cov);
  kf.update < 3>(gps,

      gps_measurement_model,
      gps_measurement_noise_cov);
```

Note how our use of `boost::bind` denotes an anonymous function that maps the state `_1` to `process_model (_1, acc, gyro)`.

Also note that there is no particular order in which `predict ()` and `update ()` need to be called, and that there can be more than one measurement model – typically one per type of sensor data.

6.1.4. Evaluation on a synthetic dataset

To conclude the worked example, we run the filter on a synthetic data set consisting of sensor data generated from a predefined trajectory with added white Gaussian noise. The estimated 3D trajectory is shown in Fig. 4, its projection into the *x-y*-plane compared to ground truth in Fig. 5. Accelerometer and gyroscope readings are available at 100 Hz ($dt = 0.01$) with white noise standard deviations $\sigma_\omega = 0.05°/s^{1/2}$ and $\sigma_v = 2$ mm/$s^{3/2}$ (MEMS class IMU). *GPS* is available at 4 Hz, with white noise of $\sigma_p = 0.75$ m.

Fig. 9. Performance of a UKF using a plain quaternion (top to bottom): Trajectory estimated, RMS error norms as in Fig. 5, NEES, from 50 Monte Carlo runs. The NEES is slightly too low (ca. by 1), probably because by using σ_ω^2 as process noise covariance in all four quaternion components, the filter "thinks" there is process noise on the norm of the quaternion, while in fact there is none. The time-averaged error is 0.434 m, 0.0171 rad, and 0.160 m/s in position, orientation, and velocity, respectively. This is slightly worse than for the ⊞-method, probably caused by the fact, that the quaternion is not fully normalized making the filter use the false DOF created by the quaternion's norm to fit to the measurements.

```
MTK_BUILD_MANIFOLD (state,
  ...
  ((vect<3>, gps_bias))
)
```

The bias b = `gps_bias` follows the process model

$$b_{t+1} = \exp\left(-\frac{dt}{T}\right)b_t + \mathcal{N}(0,\left(1 - \exp\left(-\frac{2dt}{T}\right)\right)\sigma_b^2 I_3)$$

which realizes an autocorrelation with a given variance $\text{Cov}(b_t) = \sigma_b^2 I_3$ and specified exponential decay $\text{cor}(b_t, b_{t+k}) = \exp\left(-\frac{kdt}{T}\right)$. The formula is taken from the textbook by Grewal [13, (8.76) and (8.78)] and implemented in `process_model` by

```
s2.gps_bias = exp (-dt/T_pos)* s.gps_bias;
```

and in `process_noise_cov` by

```
setDiagonal (cov, & state::gps_bias,
  gps_cnoise*(1-exp (-2*dt/T_pos)));
```

The initial covariance is also set to σ_b^2:

```
setDiagonal (init_cov, & state::gps_bias,
  gps_cnoise);
```

Finally, `gps_measurement_model` adds the bias:

```
return s.pos + s.bias;
```

Fig. 10 shows the performance of the modified filter with a simulation that includes colored noise on the GPS measurement ($\sigma_b^2 = 5$ m, $T = 1800$ s).

This example shows that *MTK* and *UKFoM* allow for rapidly trying out different representations and models without being hindered by implementing book-keeping issues. In a similar way, omitted here for lack of space, gyroscope and accelerometer bias can be integrated. Beyond that, further improvement would require operating on GPS pseudo-ranges (tightly coupled setup), with the state augmented by biases compensating for the various error sources (receiver clock error, per-satellite clock errors, ephemeris errors, atmospheric delays, etc. [13, Chapter 5]).

6.3. Pose relation graph optimization

To show the benefit of our ⊞-approach we optimized several 3D pose graphs using our manifold representation and compared it to the singular representations of Euler angles and matrix exponential (see Section 3.6.3), as well as a four dimensional quaternion representation. When using Gauss–Newton optimization the latter would fail, due to the rank-deficiency of the problem, so we added the pseudo measurement $\|q\| = 1$.

First, we show that the ⊞-method works on real-world data sets. Fig. 11 shows a simple 2D landmark SLAM problem (DLR/Spatial Cognition data set [26]). Fig. 12 shows the 3D Stanford multistorey parking garage data set, where the initial estimate is so good, most methods work well.

Second, for a quantitative comparison, we use the simulated dataset from [19, supplement], shown in Fig. 13 and investigate
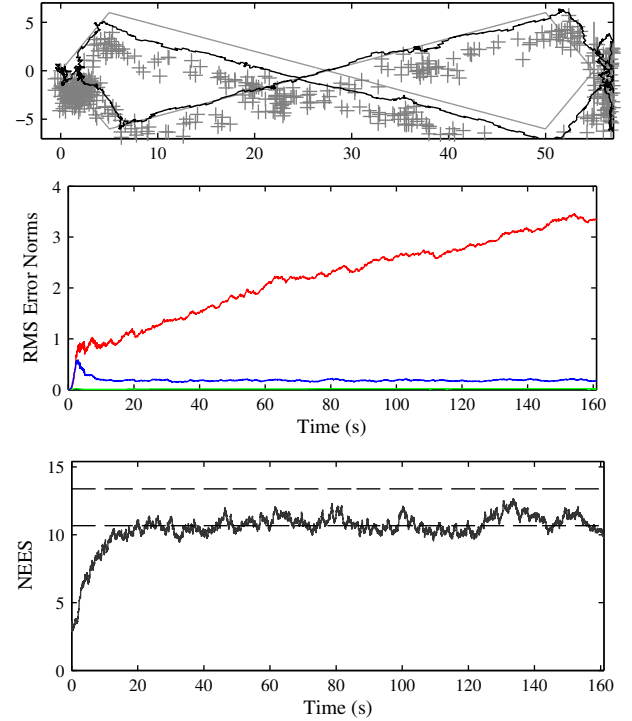

Fig. 10. Performance of the ⊞-method UKF with colored noise (top to bottom): Trajectory estimated, error norms as in Fig. 5, NEES from 50 Monte Carlo runs. The filter is consistent, notably the position error grows over time. This is as expected: The filter knows its initial position and from this it can initially deduce the GPS-bias with about 1 m precision. However, over time the bias drifts and with the inertial system being too imprecise, there is no information on the new bias and hence the position error grows. Velocity and orientation errors remain low, because these are deduced from the relative position of GPS measurements where the bias cancels out.
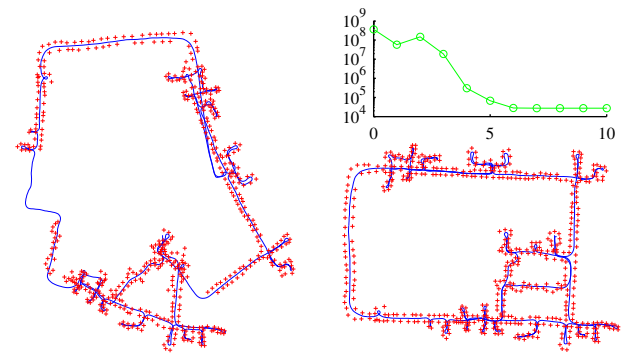

Fig. 11. The DLR data set [26] before (left) and after (right) Gauss–Newton optimization. We also show the residual sum of squares over iteration steps (top, right). No comparison is made, as in 2D the ⊞-operator only encapsulates the handling of angular periodicity.

how the different state representations behave under increasing noise levels. Fig. 14 shows the results.

7. Conclusions

We have presented a principled way of providing a local vector-space view of a manifold $\mathcal{S}$ for use with sensor fusion algorithms. We have achieved this by means of an operator $\boxplus : \mathcal{S} \times \mathbb{R}^n \to \mathcal{S}$ that adds a small vector-valued perturbation to a state in $\mathcal{S}$ and an
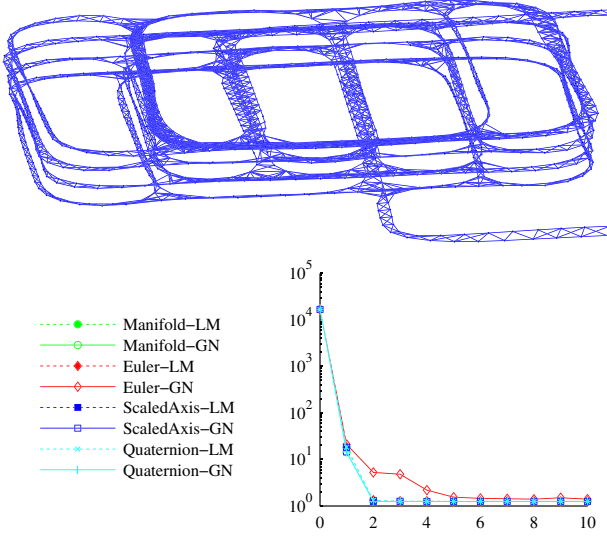
Fig. 12. Residual sum of squares over iteration steps of Gauss–Newton (GN) and Levenberg–Marquardt (LM) optimization on the Stanford parking garage data set as used in Ref. [14]. Gauss–Newton with Euler-angles is clearly inferior to all other representations, however being far away from singularities, it still converges.
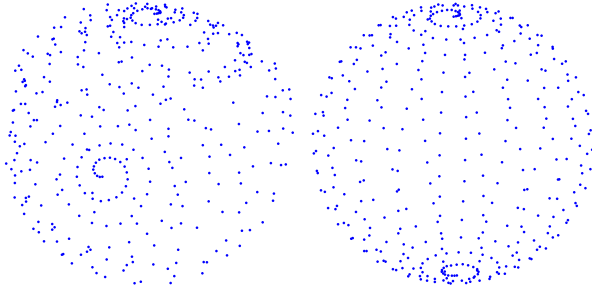

Fig. 13. The sphere400 dataset [19] generated by a virtual robot driving on a 3D sphere. It consists of a set of 400 three-dimensional poses and about 780 noisy constraints between them. The constraints stem from motion between consecutive poses or relations to previously visited poses. Poses are initialized from motion constraints (left) and then optimized with our *SLoM* framework.
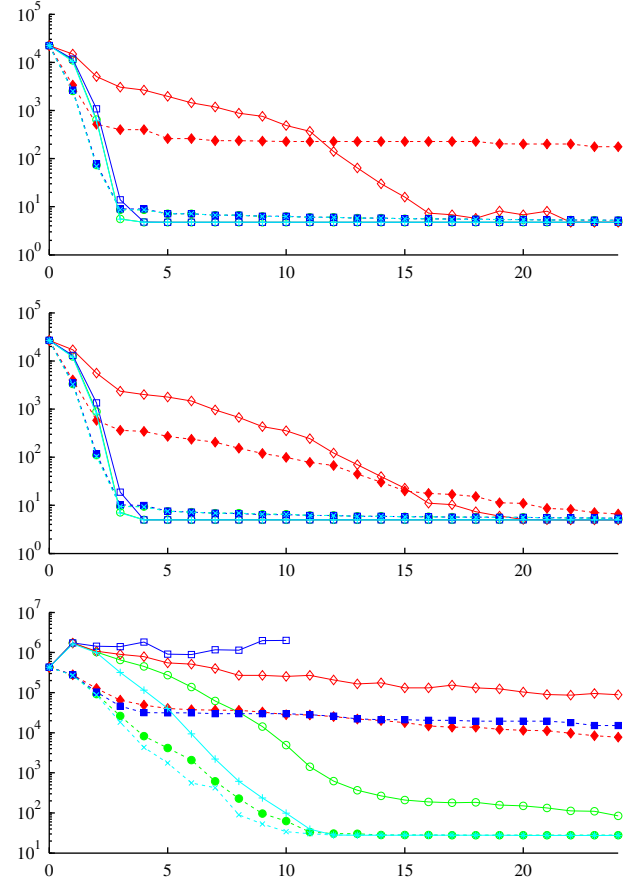
Fig. 14. Residual sum of squares over iteration steps of Gauss–Newton and Levenberg–Marquardt with different state representations on the dataset in Fig. 13 (see Fig. 12 for legend). The same data set was optimized using original noise (top) and with additional noises of 0.01 (middle) and 0.1 (bottom) rad or m, respectively. For the latter two, the median of 31 runs is plotted. Plots ending unfinished indicate that more than half the optimizations could not be finished due to a singularity. The ⊞-method clearly out-performs the singular representations. For the high noise-level the quaternion representation is slightly better. However, when comparing run-times instead of iteration counts, the latter is slower due to the additional measurements and the larger state-space (7 DOF instead of 6 DOF per pose). Computation times were 63 ms for the 4D quaternion and 42 ms for the ⊞-approach per step. This fits well to the nominal factor of $\left(\frac{7}{6}\right)^3 \approx 1.59$ for the $O(n^3)$-matrix decomposition. For Euler angle and matrix exponential, the evaluation took longer with times of 80 ms and 62 ms per step probably due to not hand-tuned code and the high number of trigonometric functions involved.

inverse operator $\boxminus : \mathcal{S} \times \mathcal{S} \to \mathbb{R}^n$ that computes the vector-valued perturbation turning one state into another. A space equipped with such operators is called a ⊞-manifold.

We have axiomatized this approach and lifted the concepts of Gaussian distribution, mean, and covariance to ⊞-manifolds therewith. The ⊞/⊟ operators allow for the integration of manifolds into generic estimation algorithms such as least-squares or the UKF mainly by replacing + and − with ⊞ and ⊟. For the UKF additionally the computation of the mean and the covariance update are modified.

The ⊞-method is not only an abstract mathematical framework but also a software engineering toolkit for implementing estimation algorithms. In the form of our Manifold Toolkit (*MTK*) implementation (and its MATLAB variant MTKM), it automatically derives ⊞/⊟ operators for compound manifolds and mediates between a flat-vector and a structured-components view.

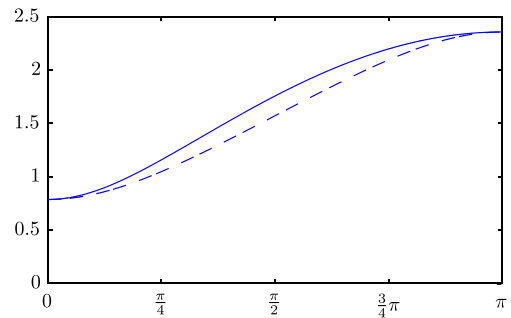**Fig. 15.** Spherical distance $\|(x \boxplus \delta_1) \boxminus (x \boxplus \delta_2)\|$ (dashed) and Euclidean distance in the tangential plane $\|\delta_1 - \delta_2\|$ (solid) plotted over the angle γ between δ_1 and δ_2 for the case $\|\delta_1\| = \alpha = \frac{\pi}{4}$ and $\|\delta_2\| = \beta = \frac{\pi}{2}$. Confer both sides of (145) for the concrete terms plotted.

Acknowledgements

Appendix A. Mathematical analysis of ⊞-manifolds

In Section 3 we have introduced the ⊞-method from a conceptual point of view, including the axiomatization of ⊞-manifolds and the generalization of the probabilistic notions of expected value, covariance, and Gaussian distribution from vector spaces to ⊞-manifolds. We will now underpin this discussion with mathematical proofs.

A.1. ⊞-Manifolds

First, we recall the textbook definition of manifolds (see, e.g. Ref. [28]). We simplify certain aspects not relevant within the context of our method; this does not affect formal correctness. In particular, we view a manifold $\mathcal{S}$ as embedded as a subset $\mathcal{S} \subset \mathbb{R}^s$ into Euclidean space $\mathbb{R}^s$ from the outset; this is without loss of generality by Whitney's embedding theorem, and simplifies the presentation for our purposes.

Definition 2 (*Manifold [28]*). A (C^k-, or smooth) *manifold* is a pair $(\mathcal{S}, \mathcal{F})$ (usually denoted just $\mathcal{S}$) consisting of a connected set $\mathcal{S} \subset \mathbb{R}^s$ and an *atlas* $\mathcal{F} = (U_\alpha, \varphi_\alpha)_{\alpha \in A}$, i.e. a family of *charts* $(U_\alpha, \varphi_\alpha)$ consisting of an open subset U_α of $\mathcal{S}$ and a homeomorphism $\varphi_\alpha : U_\alpha \to V_\alpha$ of U_α to an open subset $V_\alpha \subset \mathbb{R}^n$. Here, U_α being open in $\mathcal{S}$ means that there is an open set $\widetilde{U}_\alpha \subset \mathbb{R}^s$ such that $U_\alpha = \mathcal{S} \cap \widetilde{U}_\alpha$. These data are subject to the following requirements.

1. The charts in $\mathcal{F}$ cover $\mathcal{S}$, i.e. $\mathcal{S} = \bigcup_{\alpha \in A} U_\alpha$.
2. If $U_\alpha \cap U_\beta \neq \emptyset$, the *transition map*

$$\varphi_\alpha \circ \varphi_\beta^{-1} : \varphi_\beta(U_\alpha \cap U_\beta) \to \varphi_\alpha(U_\alpha \cap U_\beta) \tag{74}$$

is a C^k-diffeomorphism.

The number n is called the *dimension* or the number of *degrees of freedom* of $\mathcal{S}$.

We recall the generalization of the definition of *smoothness*, i.e. being k times differentiable, to functions defined on arbitrary (not necessarily open) subsets $\mathcal{S} \subset \mathbb{R}^s$:

Definition 3 (*Smooth Function*). For $\mathcal{S} \subset \mathbb{R}^s$, a function $f : \mathcal{S} \to \mathbb{R}^n$ is called *smooth*, i.e. C^k, in $x \in \mathcal{S}$ if there exists an open neighborhood $U \subset \mathbb{R}^s$ of x and a smooth function $\tilde{f} : U \to \mathbb{R}^n$ that extends $f|_{U \cap \mathcal{S}}$.

Next we show that every ⊞-manifold is indeed a manifold, justifying the name. The reverse is not true in general.

Lemma 1. *Every ⊞-manifold is a manifold, with the atlas* $(U_x, \varphi_x)_{x \in \mathcal{S}}$ *where*

$$U_x = \{ y \mid y \boxminus x \in V \}, \tag{75}$$
$$\varphi_x : U_x \to V, \quad y \mapsto y \boxminus x. \tag{76}$$

(*This result will be sharpened later in Corollary 1.*)

Proof. $\mathcal{S}$ is connected, as $\gamma : \lambda \mapsto x \boxplus (\lambda(y \boxminus x))$ is a path from $\gamma(0) = x$ by (11a) to $\gamma(1) = y$ by (11b). From (11c) we have that $\varphi_x^{-1}(\delta) = x \boxplus \delta$ is injective on V, and therefore bijective onto its image U_x. As both φ_x and φ_x^{-1} are required to be smooth, φ_x is a diffeomorphism, in particular a homeomorphism. The set U_x is open in $\mathcal{S}$, as it is the preimage of V under the continuous function φ_x, and since $x \in U_x$ we have $\mathcal{S} = \bigcup_{x \in \mathcal{S}} U_x$. Finally, the transition map $\varphi_x \circ \varphi_y^{-1}$ is a composite of diffeomorphisms and therefore diffeomorphic. □

A.2. Induced metric

Lemma 2. *The operation* $\boxminus$ *defines a metric d on* $\mathcal{S}$ *by*

$$d_\mathcal{S}(x, y) := \| y \boxminus_\mathcal{S} x \|. \tag{77}$$

Proof. Positive definiteness of d follows from Axiom (11a) and positive definiteness of $\| \cdot \|$.

Symmetry can be shown using (11d):

$$d(x, y) = \| y \boxminus x \| = \| (y \boxplus 0) \boxminus (y \boxplus (x \boxminus y)) \|$$
$$\leqslant \| x \boxminus y \| = d(y, x) \tag{78}$$

and symmetrically, which implies equality. The triangle inequality also follows from (11d):

$$d(x, z) = \| z \boxminus x \| \tag{79}$$
$$= \| (y \boxplus (z \boxminus y)) \boxminus (y \boxplus (x \boxminus y)) \| \tag{80}$$
$$\leqslant \| (z \boxminus y) - (x \boxminus y) \| \tag{81}$$
$$\leqslant \| (x \boxminus y) \| + \| (z \boxminus y) \| \tag{82}$$
$$= d(x, y) + d(y, z). \qquad \square$$

A.3. Smooth functions

Lemma 3. *For a map* $f : \mathcal{S} \to \mathcal{M}$ *between ⊞-manifolds* $\mathcal{S}$ *and* $\mathcal{M}$, *the following are equivalent for every* $x \in \mathcal{S}$:

1. f *is smooth in x (Definition 3)*
2. $f(x \boxplus_\mathcal{S} \delta) \boxminus_\mathcal{M} z$ *is smooth in* δ *at* $\delta = 0$ *whenever* $z \in \mathcal{M}$ *is such that* $f(x) \in U_z$.

Proof. 1 implies 2, as the concatenation of smooth functions is smooth.

For the converse implication, fix z as in 2, let $\mathcal{S} \subset \mathbb{R}^s$ and $\mathcal{M} \subset \mathbb{R}^m$, and let $U \subset \mathbb{R}^s$ be a neighborhood of x such that $\boxminus$ extends smoothly to $U \times \{x\}$. Then we extend f smoothly to $\tilde{f} : U \to \mathbb{R}^m$ by

$$\tilde{f}(y) = z \boxplus_\mathcal{M} (f(x \boxplus_\mathcal{S} (y \boxminus_\mathcal{S} x)) \boxminus_\mathcal{M} z). \qquad \square$$

Replacing $\boxplus / \boxminus$ in $\delta \mapsto f(x \boxplus \delta) \boxminus z$ by the induced charts as in Lemma 1, we see that smoothness corresponds to the classical definition of smooth functions on manifolds [28, p. 32]:

$$f(x \boxplus \delta) \boxminus z = \varphi_z \big(f(\varphi_x^{-1}(\delta)) \big), \tag{83}$$

with the right hand side required to be smooth in δ at $\delta = \varphi_x(x) = 0$.

A direct consequence of this fact is

Corollary 1. *Every ⊞-manifold* $\mathcal{S} \subset \mathbb{R}^s$ *is an embedded submanifold of* $\mathbb{R}^s$.

(Recall that this means that the embedding $\mathcal{S} \hookrightarrow \mathbb{R}^s$ is an immersion, i.e. a smooth map of manifolds that induces an injection of tangent spaces, and moreover that the topology of the manifold $\mathcal{S}$ is the subspace topology in $\mathbb{R}^s$ [28].)

Proof. $\mathcal{S}$ carries the subspace topology by construction. Clearly, the injection $\mathcal{S} \hookrightarrow \mathbb{R}^s$ is smooth according to Definition 3, and hence as a map of manifolds by the above argument. Since tangent spaces are spaces of differential operators on smooth real-valued functions and moreover have a local nature [28], the immersion property amounts to every smooth real-valued function on an open subset of $\mathcal{S}$ extending to a smooth function on an open subset of $\mathbb{R}^s$, which is precisely the content of Definition 3. □

A.4. Isomorphic ⊞-manifolds

For every type of structure, one has a notion of homomorphism, which describes mappings that preserve the relevant structure. The natural notion of morphism $\varphi : \mathcal{S} \to \mathcal{M}$ of ⊞-manifolds is a smooth map $\varphi : \mathcal{S} \to \mathcal{M}$ that is homomorphic w.r.t. the algebraic operations ⊞ and ⊟, i.e.

$$\varphi(x \boxplus_{\mathcal{S}} \delta) = \varphi(x) \boxplus_{\mathcal{M}} \delta, \tag{84}$$
$$\varphi(x \boxminus_{\mathcal{S}} y) = \varphi(x) \boxminus_{\mathcal{M}} \varphi(y). \tag{85}$$

As usual, an isomorphism is a bijective homomorphism whose inverse is again a homomorphism. Compatibility of inverses of homomorphisms with algebraic operations as above is automatic, so that an isomorphism $\varphi : \mathcal{S} \to \mathcal{M}$ of ⊞-manifolds is just a diffeomorphism $\varphi : \mathcal{S} \to \mathcal{M}$ (i.e. an invertible smooth map with smooth inverse) that is homomorphic w.r.t. ⊞ and ⊟. If such a φ exists, $\mathcal{S}$ and $\mathcal{M}$ are *isomorphic*. It is clear that isomorphic ⊞-manifolds are indistinguishable as such, i.e. differ only w.r.t. the representation of their elements. We will give examples of isomorphic ⊞-manifolds in Appendix B; e.g. orthonormal matrices and unit quaternions form isomorphic ⊞-manifolds.

A.5. Defining symmetric ⊞-manifolds

Most ⊞-manifolds arising in practice are manifolds with inherent symmetries. This can be exploited by defining ⊞ at one reference element and pulling this structure back to the other elements along the symmetry.

Formally, we describe the following procedure for turning an n-dimensional manifold $\mathcal{S}$ with sufficient symmetry into a ⊞-manifold. The first step is to define a smooth and surjective function $\psi : \mathbb{R}^n \to \mathcal{S}$ which is required to be locally diffeomorphic, i.e. for a neighborhood V of $0 \in \mathbb{R}^n$ it must have a smooth inverse $\varphi := \psi^{-1}$. As ψ is surjective, φ can be extended globally to a (not necessarily smooth) function $\varphi : \mathcal{S} \to \mathbb{R}^n$ such that $\psi \circ \varphi = id_{\mathcal{S}}$.

The next step is to define, for every $x \in \mathcal{S}$, a diffeomorphic transformation $R_x : \mathcal{S} \to \mathcal{S}$ (visually a "rotation") such that $R_x(\psi(0)) = x$. We can then define

$$x \boxplus \delta := R_x(\psi(\delta)), \quad y \boxminus x := \varphi\left(R_x^{-1}(y)\right). \tag{86}$$

Since $x \boxplus 0 = R_x(\psi(0)) = x$, Axiom (11a) holds under this construction. Axiom (11b) holds as we require $\psi \circ \varphi = id_{\mathcal{S}}$ globally. Finally, as $\varphi \circ \psi = id_V$ Axiom (11c) is fulfilled for $\delta \in V$ as required. Axiom (11d) depends on ψ and R_x and needs to be established on a case-by-case basis.

A.6. Lie-groups as ⊞-manifolds

For connected Lie-groups [28, Chap. 20], i.e. manifolds with a diffeomorphic group structure, the above steps are very simple. On the one hand the mapping $\psi : \mathbb{R}^n \to \mathcal{S}$ can be defined using the exponential map [28, p. 522], which is a locally diffeomorphic map from a 0-neighborhood in the Lie-algebra (a vector space diffeomorphic to $\mathbb{R}^n$) to a neighborhood of the unit element in $\mathcal{S}$. For compact Lie-groups, the exponential map is also surjective, with global inverse log.

The transformation R_x can be simply defined as $R_x(y) := x \cdot y$ (or alternatively $R_x(y) := y \cdot x$) using the group's multiplication:

$$x \boxplus \delta := x \cdot \exp(\delta), \quad y \boxminus x := \log(x^{-1} \cdot y). \tag{87}$$

Again (11a)–(11c) follow from the construction in Appendix A.5. Axiom (11d) reduces to whether

$$\|(x \boxplus \delta_1) \boxminus (x \boxplus \delta_2)\| \tag{88}$$
$$= \| \log \left((x \cdot \exp \delta_2)^{-1} \cdot (x \cdot \exp \delta_1) \right) \| \tag{89}$$
$$= \| \log (\exp(-\delta_2) \cdot \exp \delta_1) \| \tag{90}$$
$$\overset{?}{\leqslant} \|\delta_1 - \delta_2\|. \tag{91}$$

We do not know of a result that would establish this fact in general, and instead prove the inequality individually for each case.

A.7. Cartesian product of manifolds

Lemma 4. *The Cartesian product of two ⊞-manifolds $\mathcal{S}_1$ and $\mathcal{S}_2$ is a ⊞-manifold $\mathcal{S} = \mathcal{S}_1 \times \mathcal{S}_2$, with $V = V_1 \times V_2$ and*

$$(x_1, x_2) \boxplus \begin{bmatrix} \delta_1 \\ \delta_2 \end{bmatrix} = (x_1 \boxplus_{\mathcal{S}_1} \delta_1, x_2 \boxplus_{\mathcal{S}_2} \delta_2), \tag{92}$$

$$(y_1, y_2) \boxminus (x_1, x_2) = \begin{bmatrix} y_1 \boxminus_{\mathcal{S}_1} x_1 \\ y_2 \boxminus_{\mathcal{S}_2} x_2 \end{bmatrix}, \tag{93}$$

for $(x_1, x_2), (y_1, y_2) \in \mathcal{S} := \mathcal{S}_1 \times \mathcal{S}_2$ and $\begin{bmatrix} \delta_1 \\ \delta_2 \end{bmatrix} \in \mathbb{R}^{n_1} \times \mathbb{R}^{n_2}$.

Proof. Smoothness of ⊞ and ⊟ as well as Axioms (11a)–(11c) hold by component. For Axiom (11d), we see that

$$\|(x \boxplus \delta) \boxminus (x \boxplus \varepsilon)\|^2 = \|(x_1 \boxplus \delta_1) \boxminus (x_1 \boxplus \varepsilon_1)\|^2 + \|(x_2 \boxplus \delta_2) \boxminus (x_2 \boxplus \varepsilon_2)\|^2$$
$$\leqslant \|\delta_1 - \varepsilon_1\|^2 + \|\delta_2 - \varepsilon_2\|^2 = \|\delta - \varepsilon\|^2. \qquad \square$$

A.8. Expected value on ⊞-manifolds

Also using Axiom (11d), we prove that the definition of the expected value by a minimization problem in (15) implies the implicit definition (16):

Lemma 5. *For a random variable $X : \Omega \to \mathcal{S}$ and $\mu \in \mathcal{S}$,*

$$\mathrm{E}\|X \boxminus \mu\|^2 = \min_{\mu \in \mathcal{S}}! \Rightarrow \mathrm{E}(X \boxminus \mu) = 0. \tag{94}$$

Proof. Let $\mu := \mathrm{argmin}_\mu \mathrm{E}\|X \boxminus \mu\|^2$. Then

$$\mathrm{E}\|X \boxminus \mu\|^2 \leqslant \mathrm{E}\|X \boxminus (\mu \boxplus \mathrm{E}(X \boxminus \mu))\|^2 \tag{95}$$
$$= \mathrm{E}\|(\mu \boxplus (X \boxminus \mu)) \boxminus (\mu \boxplus \mathrm{E}(X \boxminus \mu))\|^2 \tag{96}$$
$$\leqslant \mathrm{E}\|(X \boxminus \mu) - \mathrm{E}(X \boxminus \mu)\|^2 \tag{97}$$
$$= \mathrm{E}\|X \boxminus \mu\|^2 - \|\mathrm{E}(X \boxminus \mu)\|^2. \tag{98}$$

Hence, $\mathrm{E}(X \boxminus \mu) = 0$. $\qquad \square$

A.9. (Gaussian) Distributions on ⊞-manifolds

The basic idea of (12) in Section 3.4 was to map a distribution $X : \Omega \to \mathbb{R}^n$ to a distribution $Y : \Omega \to \mathcal{S}$ by defining $Y := \mu \boxplus X$ for some $\mu \in \mathcal{S}$. The problem is that in general ⊞ is not injective. Thus (infinitely) many X are mapped to the same Y, which makes even simple things such as computing $p(Y = y)$ for a given $y \in \mathcal{S}$ complicated, not to mention maximizing likelihoods.

A pragmatic approach is to "cut off" the distribution X where ⊞ becomes ambiguous, i.e. define a distribution $\widetilde{X}$ with

$$\mathrm{p}(\widetilde{X} = x) := \mathrm{p}(X = x | X \in V). \tag{99}$$

This can be justified because, if V is large compared to the covariance, $P(X \notin V)$ is small and the cut-off error is negligible. In practice,

the fact that noise usually does not really obey a normal distribution leads to a much bigger error.

Now (12) simplifies to

$$p(\mu \boxplus \widetilde{X} = y) = p(\widetilde{X} = y \boxminus \mu), \tag{100}$$

because $\boxplus$ is bijective for $\widetilde{X} \in V$. We also find that for normal distributed noise, the maximum likelihood solution is the least squares solution.

Lemma 6. *For random variables $X : \Omega \to \mathcal{S}, Z : \Omega \to \mathcal{M}$, a measurement $z \in \mathcal{M}$, and a measurement function $f : \mathcal{S} \to \mathcal{M}$, with $f(X) = z \boxplus \tilde{\varepsilon}$ and $\tilde{\varepsilon} \sim \mathcal{N}(0, \Sigma)$ under the precondition $\tilde{\varepsilon} \in V$, the x with largest likelihood $p(Z = z | X = x, \tilde{\varepsilon} \in V)$ is the one that minimizes $\frac{1}{2} \| f(x) \boxminus z \|_{\Sigma}^2$.*

Proof.

$$p(Z = z | X = x, \tilde{\varepsilon} \in V) = \tag{101}$$
$$p(z \boxplus \tilde{\varepsilon} = f(x) | \tilde{\varepsilon} \in V) = p(\tilde{\varepsilon} = f(x) \boxminus z)$$

$$\propto \exp\left(-\frac{1}{2} \| f(x) \boxminus z \|_{\Sigma}^2\right) = \max! \tag{102}$$

$$\iff -\ln\left(\exp\left(-\frac{1}{2} \| f(x) \boxminus z \|_{\Sigma}^2\right)\right)$$

$$= \frac{1}{2} \| f(x) \boxminus z \|_{\Sigma}^2 = \min! \tag{103}$$

Thus the classical approach of taking the negative log-likelihood shows the equivalence. $\quad\square$

Appendix B. Examples of $\boxplus$-manifolds

In this appendix we will show Axioms (11) for the $\boxplus$-manifolds discussed in Section 3.6 and further important examples. All, except $\mathbb{R}/2\pi\mathbb{Z}$ are based on either rotation matrices $SO(n)$ or unit vectors S^n, so we start with these general ones. Often several representations are possible, i.e. $\mathbb{R}/2\pi\mathbb{Z}, SO(2)$, or S^1 for 2D rotations and $SO(3)$ or $\mathbb{H}$ for 3D rotations. We will show these representations to be isomorphic, so in particular, Axiom (11d) holds for all if it holds for one.

B.1. The rotation group SO(n)

Rotations are length, handedness, and origin preserving transformations of $\mathbb{R}^n$. Formally they are defined as a matrix-group

$$SO(n) = \{Q \in \mathbb{R}^{n \times n} \mid Q^\top Q = I, \det Q = 1\}.$$

Being subgroups of $Gl(n)$, the $SO(n)$ are Lie-groups. Thus we can use the construction in (87):

$$x \boxplus \delta = x \exp \delta \quad y \boxminus x = \log(x^{-1}y). \tag{104}$$

The matrix exponential is defined by the usual power series $\exp \delta = \sum_{i=0}^{\infty} \frac{1}{i!} \delta^i$, where the vector δ is converted to an antisymmetric matrix (we omit the $^\wedge$ commonly indicating this). The logarithm is the inverse of exp. The most relevant $SO(2), SO(3)$ have analytic formulas, Refs. [34,6] give general numerical algorithms.

Eq. (104) fulfills axioms (11a)–(11c) for suitable V by the construction using the Lie-group structure. We conjecture that we can take $V = B_\pi(0)$, and that the remaining axiom (11d) also holds in general. We prove this for $n = 2$ using an isomorphism to $\mathbb{R}/2\pi\mathbb{Z}$ (Appendix B.3) and for $n = 3$ using an isomorphism to $\mathbb{H}$ (Appendix B.6).

B.2. Directions in $\mathbb{R}^{n+1}$ as unit vectors S^n

Another important manifold is the unit-sphere

$$S^n = \{x \in \mathbb{R}^{n+1} \mid \|x\| = 1\}, \tag{105}$$

the set of directions in $\mathbb{R}^{n+1}$. In general S^n is no Lie-group, but we can still exploit symmetry by (86) (Section A.5) and define a mapping R_x that takes the first unit vector e_1 to x. This is achieved by a Householder-reflection [35, Chapter 11.2].

$$R_x = \begin{cases} (I - 2\frac{vv^\top}{v^\top v})X, & \text{for } v = x - e_1 \neq 0, \\ I, & \text{for } x = e_1. \end{cases} \tag{106}$$

Here X is a matrix negating the second vector component. It makes R_x the product of two reflections and hence a rotation. To define $e_1 \boxplus \delta$ we define exp and log for S^n as

$$\exp \delta = \begin{bmatrix} \cos \|\delta\| \\ \text{sinc}\|\delta\|\delta \end{bmatrix}, \tag{107}$$

$$\log[wv] = \begin{cases} \text{atan2}(0, w)e_1 & v = 0 \\ \frac{\text{atan2}(\|v\|, w)}{\|v\|}v & v \neq 0. \end{cases} \tag{108}$$

We call these functions exp and log, because they correspond to the usual power-series on complex numbers (S^1) and quaternions (S^3). In general, however, there is only a rough analogy.

Now, S^n can be made a $\boxplus$-manifold by (86), with $\psi = \exp$ and $\varphi = \psi^{-1} = \log$:

$$x \boxplus \delta = R_x \exp \delta, \quad y \boxminus x = \log(R_x^\top y). \tag{109}$$

The result looks the same as the corresponding definition (87) for Lie-groups, justifying the naming of (107) and (108) as exp and log. We have that exp is left inverse to log, and log is left inverse to exp on $\|\delta\| < \pi$. As proved in Lemma 7 (Appendix C), exp and log are smooth. Hence Axioms (11a)–(11c) hold for $V = B_\pi(0)$ (Section A.5). Axiom (11d) is proved as Lemma 11 in Appendix C.

The induced metric $d(x, y)$ corresponds to the angle between x and y (Lemma 10).

Note that the popular stereographic projection cannot be extended to a $\boxplus$-manifold, because it violates Axiom (11b).

Equipped with $\boxplus$-manifolds for $SO(n)$ and S^n, we now discuss the most important special cases, first for $n = 2$, then $n = 3$.

B.3. 2D orientation as an orthonormal matrix

For planar rotations, exp in (104) takes an antisymmetric 2×2 matrix, i.e. a number and returns the well-known 2D rotation matrix

$$x \boxplus \delta = x \exp \delta, \quad y \boxminus x = \log(x^{-1}y), \tag{110}$$

$$\exp \delta = \begin{bmatrix} \cos \delta & -\sin \delta \\ \sin \delta & \cos \delta \end{bmatrix}, \qquad \log x = \text{atan2}(x_{21}, x_{11}).$$

The function exp is also an isomorphism between $\mathbb{R}/2\pi\mathbb{Z}$ (24) and $SO(2)$ (111), because the 2π-periodicity of exp as a function matches the periodicity of $\mathbb{R}/2\pi\mathbb{Z}$ as a set of equivalence classes and $\exp x \cdot \exp \delta = \exp(x + \delta)$. The latter holds as multiplication in $SO(2)$ commutes. From this argument we see that $\mathbb{R}/2\pi\mathbb{Z}$ (Section 3.6.2) and $SO(2)$ are isomorphic $\boxplus$-manifolds and also that axiom (11d) holds for the latter.

B.4. 2D orientation as a complex number

Using complex multiplication, $SO(2)$ is isomorphic to the complex numbers of unit-length, i.e. $S^2 \subset \mathbb{C}$. For $n = 2$, (107) and (108) simplify to

$$\exp_{S^1} \delta = \begin{bmatrix} \cos \delta \\ \sin \delta \end{bmatrix}, \quad \log_{S^1} \begin{bmatrix} x \\ y \end{bmatrix} = \text{atan2}(y, x), \tag{111}$$

$R_{\begin{bmatrix} x \\ y \end{bmatrix}}$ equals complex multiplication with $\begin{bmatrix} x \\ y \end{bmatrix}$ or, as a matrix,

$\begin{bmatrix} x & -y \\ y & x \end{bmatrix}$. This is because $R_{\begin{bmatrix} x \\ y \end{bmatrix}}$ it is a rotation mapping e_1 to $\begin{bmatrix} x \\ y \end{bmatrix}$

and there is only one such rotation on S^1. With these prerequisites, S^1 is isomorphic to $SO(2)$ by Lemma 14 as is $\mathbb{R}/2\pi\mathbb{Z}$.

B.5. Directions in 3D space as S^2

The unit sphere is the most important example of a manifold that is not a Lie-group. This is a consequence of the "hairy ball theorem" [9], which states that on S^2 every continuous vector-field has a zero – if S^2 was a Lie-group, one could take the derivative of $x \cdot \delta$ for every x and some fixed δ to obtain a vector field on S^2 without zeroes. With the same argument applied to $x \boxplus \delta$, it is also impossible to give S^2 a $\boxplus$-structure that is continuous in x. This is one reason why we did not demand continuity in x for $\boxplus$. It also shows that the discontinuity in (106) cannot be avoided in general (although it can for S^1 and S^3).

However, we can give a simpler analytical formula for R_x in S^2:

$$R_{\begin{bmatrix} x \\ y \\ z \end{bmatrix}} := \begin{bmatrix} x & -r & 0 \\ y & xc & -s \\ z & xs & c \end{bmatrix}, \quad \alpha = \operatorname{atan2}(z,y), \tag{112}$$

$$c = \cos\alpha, \quad s = \sin\alpha, \quad r = \sqrt{x^2 + z^2},$$

$$x \boxplus \delta = R_x \exp\delta, \quad y \boxminus x = \log\left(R_x^\top y\right). \tag{113}$$

The formula is discontinuous for $r = 0$, but any value for α leads to a proper rotation matrix R_x. Therefore, neither $\boxplus$ nor $\boxminus$ are continuous in x, but they are smooth with respect to δ or y.

B.6. 3D orientation as a unit quaternion

The $\boxplus$-manifold for quaternions $\mathbb{H}$ presented in Section 3.6.4, (29)–(31) is a special case of (107)–(109) for the unit sphere S^3. In the construction R_q from (106) is conveniently replaced by $q^{-1}\cdot$, because $\mathbb{H}$ is a Lie-group. Also, exp and log correspond again to the usual functions for $\mathbb{H}$. The Axioms (11a)–(11c) are fulfilled for $V = B_\pi(0)$. Axiom (11d) is proved in Lemma 12.

The metric $d(x, y)$ is the angle between x and y, and also monotonically related to the simple Euclidean metric $\|x - y\|$ (Lemma 8).

Orthonormal matrices $SO(3)$ and unit quaternions $S^3/\{\pm 1\}$ are two different representations of rotations. Topologically this is called a universal covering [3, §12]. Hence, their $\boxplus$-manifolds are isomorphic with the usual conversion operation

$$\varphi\begin{bmatrix} w \\ x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1 - 2(y^2 + z^2) & -2wz + 2xy & -2wy + 2xy \\ 2wz + 2xy & 1 - 2(x^2 + z^2) & -2wx + 2yz \\ -2wy + 2xz & 2wx + 2yz & 1 - 2(x^2 + y^2) \end{bmatrix}. \tag{114}$$

For the proof we use that $\exp_{SO(3)}(\alpha v/\|v\|)$ (26) and $\exp_{S^3}(\frac{1}{2}\alpha v/\|v\|)$ (30) are the well-known expressions for the rotation by an angle α around an axis v in matrix and quaternion representation. So after converting the quaternion to a matrix they are equal:

$$\varphi(\exp_{S^3}(\delta/2)) = \exp_{SO(3)}(\delta), \tag{115}$$

$$\varphi(q\boxplus_{S^3}\delta), \tag{116}$$

$$= \varphi(q\exp_{S^3}(\delta/2)) = \varphi(q)\varphi(\exp_{S^3}(\delta/2)), \tag{117}$$

$$= \varphi(q)\exp_{SO(3)}(\delta) = \varphi(q)\boxplus_{SO(3)}\delta. \tag{118}$$

The isomorphism also shows Axiom (11d) for $SO(3)$.

B.7. The projective space as S^n

Further important manifolds, e.g. in computer vision, are the projective spaces, informally non-Euclidean spaces where parallels intersect at infinity. Formally we define

$$\mathbb{P}^n := (\mathbb{R}^{n+1} \setminus \{0\})/(\mathbb{R} \setminus \{0\}) \tag{119}$$

and write

$$[x_0 : x_1 : \cdots : x_n] = [x] = \{\lambda x \mid \lambda \in \mathbb{R} \setminus \{0\}\} \tag{120}$$

for the equivalence class modulo $\mathbb{R} \setminus \{0\}$ of a vector $x = (x_1, \ldots, x_n) \in \mathbb{R}^{n+1} \setminus \{0\}$. In other words, $\mathbb{P}^n$ is the space of non-zero $(n + 1)$-dimensional vectors modulo identification of scalar multiples.

As every point $[x] \in \mathbb{P}^n$ can uniquely be identified with the set $\left\{\frac{x}{\|x\|}, \frac{-x}{\|x\|}\right\} \subset S^n$ we find that $\mathbb{P}^n \cong S^n/\{\pm 1\}$ and S^n is a cover of $\mathbb{P}^n$. For $n = 3$, we see the not quite intuitive fact that $\mathbb{P}^3 \cong SO(3)$, so we can reuse the same $\boxplus$-manifold there. For $\mathbb{P}^2$ we can basically reuse S^2 but care has to be taken due to the ambiguity of $x \equiv -x$. This can be solved by using $\overline{\log}$ instead of log in $\boxminus$, fulfilling Axioms (11) for $V = B_{\pi/2}$.

We cannot currently say anything about the induced metric on a projective space.

Appendix C. Technical proofs

Lemma 7. *The exponential* $\exp\delta = \begin{bmatrix} \cos\|\delta\| \\ \operatorname{sinc}\|\delta\|\delta \end{bmatrix}$ *from (107) is analytical on* $\mathbb{R}^n$ *and log is analytical on* $S^n \setminus \left\{\begin{bmatrix} -1 \\ 0 \end{bmatrix}\right\}$. *(Therefore, also* C^∞.)

Proof. The functions cos and sinc are both globally analytic, with Taylor series

$$\cos\|\delta\| = \sum_{k=0}^\infty \frac{(-1)^k\|\delta\|^{2k}}{(2k)!} = \sum_{k=0}^\infty \frac{(-\|\delta\|^2)^k}{(2k)!}, \tag{121}$$

$$\operatorname{sinc}\|\delta\| = \sum_{k=0}^\infty \frac{(-1)^k\|\delta\|^{2k}}{(2k+1)!} = \sum_{k=0}^\infty \frac{(-\|\delta\|^2)^k}{(2k+1)!}. \tag{122}$$

Moreover, $\|\delta\|^2 = \sum_{i=1}^n \delta_i^2$ is also analytic.

On the restriction $\exp : B_\pi \to S^n \setminus \left\{\begin{bmatrix} -1 \\ 0 \end{bmatrix}\right\}$ the inverse of exp is log. In order to prove that log is analytic, we have to show that the Jacobian of exp has full rank. Using that the derivative of $\|\delta\|$ is $\delta^\top/\|\delta\|$ and hence the derivative of $\cos\|\delta\|$ is $\operatorname{sinc}\|\delta\|\delta^\top$, we show that for every $v \neq 0$:

$$\frac{\partial}{\partial\delta}(\exp\|\delta\|)v = \begin{bmatrix} \operatorname{sinc}\|\delta\|\delta^\top \\ \operatorname{sinc}\|\delta\|I + \delta(\operatorname{sinc}'\|\delta\|)/\|\delta\|\delta^\top \end{bmatrix} v, \tag{123}$$

where the first component vanishes only for $\delta^\top v = 0$ (as $\operatorname{sinc} x \neq 0$ for $|x| < \pi$). In this case the lower part becomes $\operatorname{sinc}\|\delta\|v$, which never vanishes for $\|\delta\| < \pi$. $\qquad\square$

Lemma 8. *For two unit quaternions* $a, b \in S^3$, *there is a monotonic mapping between their Euclidean distance and the distance induced by* $\boxminus$:

$$\|a - b\|^2 = f(\|a \boxminus b\|), \quad \text{with} f(\alpha) = 2 - 2\cos(\alpha/2) \tag{124}$$

This also holds when antipodes are identified and both metrics are defined as the minimum obtained for any choice of representatives from the equivalence classes $\{a, -a\}, \{b, -b\}$.

Proof. We put $\delta := b \boxminus a = 2\log(b^{-1}a)$. Then

$$\|b - a\|^2 = \left\| \begin{bmatrix} 1 \\ 0 \end{bmatrix} - b^{-1}a \right\|^2 \tag{125}$$

$$= \left\| \begin{bmatrix} 1 \\ 0 \end{bmatrix} - \exp\left(\frac{1}{2}\delta\right) \right\|^2 \tag{126}$$

$$= \left\| \begin{bmatrix} 1 - \cos(\frac{1}{2}\|\delta\|) \\ \operatorname{sinc}(\frac{1}{2}\|\delta\|)\frac{1}{2}\delta \end{bmatrix} \right\|^2 \tag{127}$$

$$= \left(1 - \cos\left(\frac{1}{2}\|\delta\|\right)\right)^2 + \left(\operatorname{sinc}\left(\frac{1}{2}\|\delta\|\right)\left(\frac{1}{2}\|\delta\|\right)\right)^2$$

$$= \left(1 - \cos\left(\frac{1}{2}\|\delta\|\right)\right)^2 + \sin^2\left(\frac{1}{2}\|\delta\|\right) \tag{128}$$

$$= 2 - 2\cos\left(\frac{1}{2}\|\delta\|\right). \tag{129}$$

The inequality holds for every pair of antipodes (a, b), $(-a, b)$, $(a, -b)$, $(-a, -b)$ hence also for their minimum, which is by definition the distance of the equivalence classes. $\square$

Lemma 9. *For two orthonormal matrices $A, B \in SO(3)$, there is a monotonic mapping between their Frobenius distance and the distance induced by $\boxminus$:*

$$\|B - A\|_F^2 = f(\|A \boxminus B\|) \quad \text{with } f(\alpha) = 4 - 4\cos\alpha. \tag{130}$$

Proof. We put $\delta = A \boxminus B$. Then

$$\|B - A\|_F^2 = \|A \boxplus \delta - A\|_F^2 \tag{131}$$

$$= \|A \exp\delta - A\|_F^2 \tag{132}$$

$$= \|\exp\delta - I\|_F^2. \tag{133}$$

Let Q be an orthonormal matrix that rotates δ into x-direction, i.e. $Q\delta = (\|\delta\|, 0, 0)^\top$. As $\exp(Q\delta)Q = Q\exp\delta$, we have

$$= \|Q^\top \exp(Q\delta)Q - I\|_F^2 \tag{134}$$

$$= \|\exp(Q\delta) - I\|_F^2 \tag{135}$$

$$= \|\exp(\|\delta\|, 0, 0)^\top - I\|_F^2 \tag{136}$$

$$= \left\| \begin{bmatrix} 0 & 0 & 0 \\ 0 & \cos\|\delta\| - 1 & -\sin\|\delta\| \\ 0 & \sin\|\delta\| & \cos\|\delta\| - 1 \end{bmatrix} \right\|_F^2 \tag{137}$$

$$= 2(\cos\|\delta\| - 1)^2 + 2\sin^2\|\delta\| \tag{138}$$

$$= 4 - 4\cos\|\delta\|. \qquad \square$$

Lemma 10. *The curve $x \boxplus (\lambda\delta)$, with $\lambda \in [0, 1]$ and $\boxplus$ defined by (109), is a geodetic on S^n with arc-length $\|\delta\|$.*

Proof. We have

$$x \boxplus (\lambda\delta) = R_x \exp(\lambda\delta) = R_x \begin{bmatrix} \cos(\lambda\|\delta\|) \\ \operatorname{sinc}(\lambda\|\delta\|)(\lambda\delta) \end{bmatrix}$$

$$= R_x \begin{bmatrix} \cos(\lambda\|\delta\|) \\ \sin(\lambda\|\delta\|)\delta/\|\delta\| \end{bmatrix} = R_x \begin{bmatrix} 1 & 0 \\ 0 & \delta/\|\delta\| \end{bmatrix} \begin{bmatrix} \cos(\lambda\|\delta\|) \\ \sin(\lambda\|\delta\|) \end{bmatrix}$$

It can be seen that $x \boxplus (\lambda\delta)$ is a circle segment with radius 1 and hence a geodetic of length $\|\delta\|$ on S^n. $\square$

Lemma 11. *For the $\boxplus$ and $\boxminus$ operators on the hypersphere S^n defined in (109), Axiom (11d) holds (Fig. 15).*

Proof. By Lemma 10, the expression $\|(x \boxplus \delta_1) \boxminus (x \boxplus \delta_2)\|$ involves a triangle of three geodetics $(x, x \boxplus \delta_1)$, $(x, x \boxplus \delta_2)$, and $(x \boxplus \delta_1, x \boxplus \delta_2)$. By the same lemma, the first two have length $\alpha = \|\delta_1\|$ and $\beta = \|\delta_2\|$. Hence by the spherical law of cosines, with $\gamma = \angle(\delta_1, \delta_2)$, the third has a length of

$$\|(x \boxplus \delta_1) \boxminus (x \boxplus \delta_2)\| \tag{139}$$

$$= \operatorname{acos}(\cos\alpha\cos\beta + \sin\alpha\sin\beta\cos\gamma) \tag{140}$$

$$\overset{\text{Lemma 13}}{\leqslant} \sqrt{\alpha^2 + \beta^2 - 2\alpha\beta\cos\gamma} = \|\delta_1 - \delta_2\|. \qquad \square$$

Lemma 12. *For the quaternion $\boxplus$ and $\boxminus$ operators defined in (29), Axiom (11d) holds (Fig. 15).*

Proof. We apply the definitions and notice from (31) that $\|\overline{\log}q\| = \operatorname{acos}|\Re q| \leqslant \operatorname{acos}(\Re q)$ exploiting that $\|q\| = 1$. Thus

$$\|(x \boxplus \delta_1) \boxminus (x \boxplus \delta_2)\|$$

$$= \left\| 2\overline{\log}\left(\left(x\exp\frac{\delta_1}{2}\right)^{-1} \cdot \left(x\exp\frac{\delta_2}{2}\right) \right) \right\| \tag{141}$$

$$\leqslant 2\operatorname{acos}\Re\left(\exp\frac{-\delta_1}{2} \cdot \exp\frac{\delta_2}{2} \right) \tag{142}$$

$$= 2\operatorname{acos}\Re\left(\begin{bmatrix} \cos(\|\delta_1\|/2) \\ -\operatorname{sinc}(\|\delta_1\|/2)/2\delta_1 \end{bmatrix} \cdot \begin{bmatrix} \cos(\|\delta_2\|/2) \\ \operatorname{sinc}(\|\delta_2\|/2)/2\delta_2 \end{bmatrix} \right).$$

Now we apply the definition of quaternion multiplication $\begin{bmatrix} r_1 \\ v_1 \end{bmatrix} \cdot \begin{bmatrix} r_2 \\ v_2 \end{bmatrix} = \begin{bmatrix} r_1 r_2 - v_1^\top v_2 \\ \bullet \end{bmatrix}$ and substitute $\alpha = \|\delta_1\|/2$, $\beta = \|\delta_2\|/2$, and $\gamma = \angle(\delta_1, \delta_2)$. The term $v_1^\top v_2$ becomes $-\sin\alpha\sin\beta\cos\gamma$, and hence we can continue the above chain of equalities with

$$= 2\operatorname{acos}(\cos\alpha\cos\beta + \sin\alpha\sin\beta\cos\gamma) \tag{143}$$

$$\overset{\text{Lemma 13}}{\leqslant} 2\sqrt{\alpha^2 + \beta^2 - 2\alpha\beta\cos\gamma} \tag{144}$$

$$= \|\delta_1 - \delta_2\|. \qquad \square$$

Lemma 13. *The distance on a sphere is less or equal to the Euclidean distance in the tangential plane (Fig. 15). Formally, for all $\alpha, \beta \geqslant 0, \gamma \in \mathbb{R}$*

$$\operatorname{acos}(\cos\alpha\cos\beta + \sin\alpha\sin\beta\cos\gamma) \leqslant \sqrt{\alpha^2 + \beta^2 - 2\alpha\beta\cos\gamma} \tag{145}$$

Proof. If the right-hand side exceeds π, the inequality is trivial. Otherwise we substitute $\lambda := \cos\gamma$ and take the cosine:

$$\cos\alpha\cos\beta + \sin\alpha\sin\beta\lambda \geqslant \cos\sqrt{\alpha^2 + \beta^2 - 2\alpha\beta\lambda}. \tag{146}$$

The proof idea is that the left-hand side of (146) is linear in λ, the right-hand side is convex in λ, and both are equal for $\lambda = \pm 1$. Formally, from the cosine addition formula we get

$$\cos\alpha\cos\beta - \sin\alpha\sin\beta = \cos(\alpha+\beta) \tag{147}$$

$$= \cos\sqrt{\alpha^2+\beta^2+2\alpha\beta}, \tag{148}$$

$$\cos\alpha\cos\beta + \sin\alpha\sin\beta = \cos(\alpha-\beta) \tag{149}$$

$$= \cos\sqrt{\alpha^2+\beta^2-2\alpha\beta}. \tag{150}$$

Taking a $\frac{1-\lambda}{2}:\frac{1+\lambda}{2}$ convex combination of (147) and (149) we get the left-hand side of (146):

$$\cos\alpha\cos\beta + \sin\alpha\sin\beta\lambda \tag{151}$$

$$= \cos\alpha\cos\beta + \sin\alpha\sin\beta\left(\frac{1-\lambda}{2}(-1)+\frac{1+\lambda}{2}(+1)\right)$$

$$= \frac{1-\lambda}{2}\cos\sqrt{\alpha^2+\beta^2-2\alpha\beta}$$
$$+ \frac{1+\lambda}{2}\cos\sqrt{\alpha^2+\beta^2+2\alpha\beta}$$

$$\geqslant \cos\sqrt{\alpha^2+\beta^2+2\alpha\beta\left(\frac{1-\lambda}{2}(-1)+\frac{1+\lambda}{2}(+1)\right)}$$

$$= \cos\sqrt{\alpha^2+\beta^2+2\alpha\beta\lambda}. \tag{152}$$

The inequality comes from the convexity of the right-hand side of (146) in λ. We prove this by calculating its derivative

$$-\sin\sqrt{\alpha^2+\beta^2+2\alpha\beta\lambda}\frac{1}{2\sqrt{\alpha^2+\beta^2+2\alpha\beta\lambda}}2\alpha\beta$$

$$= -\mathrm{sinc}\sqrt{\alpha^2+\beta^2+2\alpha\beta\lambda}\alpha\beta \tag{153}$$

and observing that (153) increases monotonically until the square root exceeds π. □

Lemma 14. *The following function φ is an $\boxplus$-isomorphism between S^1 and SO(2):*

$$\varphi: S^1 \to SO(2), \quad \varphi\begin{bmatrix}x\\y\end{bmatrix} = \begin{bmatrix}x & -y\\y & x\end{bmatrix}. \tag{154}$$

Proof. The map is bijective, because all matrices in $SO(2)$ are of the form (111). It also commutes with $\boxplus$, since

$$\varphi\left(\begin{bmatrix}x\\y\end{bmatrix}\boxplus_{S^1}\delta\right) = \varphi\left(R^\top_{\begin{bmatrix}x\\y\end{bmatrix}}\exp\delta\right) \tag{155}$$

$$= \varphi\left(\begin{bmatrix}x & -y\\y & x\end{bmatrix}\begin{bmatrix}\cos\delta\\\sin\delta\end{bmatrix}\right) \tag{156}$$

$$= \varphi\left(\begin{bmatrix}x\cos\delta - y\sin\delta\\y\cos\delta + x\sin\delta\end{bmatrix}\right) \tag{157}$$

$$= \begin{bmatrix}x\cos\delta - y\sin\delta & -y\cos\delta - x\sin\delta\\y\cos\delta + x\sin\delta & x\cos\delta - y\sin\delta\end{bmatrix} \tag{158}$$

$$= \begin{bmatrix}x & -y\\y & x\end{bmatrix}\begin{bmatrix}\cos\delta & -\sin\delta\\\sin\delta & \cos\delta\end{bmatrix} \tag{159}$$

$$= \begin{bmatrix}x & -y\\y & x\end{bmatrix}\exp\delta \tag{160}$$

$$= \varphi\left(\begin{bmatrix}x\\y\end{bmatrix}\right)\boxplus_{SO(2)}\delta. \quad \square$$

Appendix D. Comparison to SPmap

In an SPmap [7] (originally 2D but extended to 3D) every geometric entity is represented by a reference pose ($SE(3)$) in an arbitrary, potentially overparametrized representation, and a perturbation vector parametrizing the entity relative to its reference pose in minimal parametrization. The estimation algorithm operates solely on the perturbation vector. This corresponds to $s\boxplus\delta$, with s being the reference pose and δ the perturbation vector. Actually, this concept and the idea that in most algorithms $\boxplus$ can simply replace + motivated the axiomatization of $\boxplus$-systems we propose. Our contribution is to give this idea, which has been around for a while, a thorough mathematical framework more general than geometric entities.

If a geometric entity is "less than a pose", e.g. a point, SPmap still uses a reference pose but the redundant DOFs, e.g. rotation, are removed from the perturbation vector. In our axiomatization the pose would simply be an overparametrization of a point. Using the notation of [7]

$$s\boxplus\delta = s\oplus(B^\top\delta), \tag{161}$$

where B is the so-called binding matrix that maps entries of δ to the DOF of a pose, and $\oplus$ is the concatenation of poses.

Analogously, the SPmap represents, e.g. a line as the pose's x-axis with x-rotation and translation being redundant DOFs removed from the perturbation vector. Here lies a theoretical difference. Consider two such poses differing by an x-rotation. They represent the same line but differ in the effect of the perturbation vector, as y- and z-axes point into different directions. For us, the $\boxplus$-system $\mathcal{S}$ would be a space of equivalence classes of poses. However, $\boxplus$ maps from $\mathcal{S}\times\mathbb{R}^n$ to $\mathcal{S}$, so $s\boxplus\delta$ must formally be the equivalent for equivalent poses s. The SPmap representation has the advantage that it is continuous both in the pose and the perturbation vector, which is not possible with our axiomatization due to the hairy ball theorem as discussed in Section B.5.

Overall, our contribution is the axiomatized and more general view, not limited to quotients of $SE(3)$ as with the SPmap. We currently investigate axiomatization of SPmap's idea to allow different representatives of the same equivalence class to define different $\boxplus$-coordinate systems. We avoided this here, because it adds another level of conceptual complexity.

References

[1] P. Abbeel, Apprenticeship Learning and Reinforcement Learning with Application to Robotic Control, Ph.D. thesis, Department of Computer Science, Stanford University, 2008.
[2] D. Abrahams, A. Gurtovoy, C++ Template Metaprogramming, Pearson Education, 2005.
[3] S.L. Altmann, Rotations, Quaternions and Double Groups, Oxford University Press, 1986.
[4] Y. Bar-Shalom, X. Li, T. Kirubarajan, Estimation with Applications to Tracking and Navigation, John Wiley & Sons, Inc., 2001.
[5] O. Birbach, Accuracy Analysis of Camera-Inertial Sensor Based Ball-Trajectory Prediction, Master's thesis, Universität Bremen, 2008.
[6] J. Cardosoa, F. Leitec, Exponentials of skew-symmetric matrices and logarithms of orthogonal matrices, Journal of Computational and Applied Mathematics 233 (2010) 2867–2875.
[7] J.A. Castellanos, J. Montiel, J. Neira, J.D. Tardós, The SPmap: a probablistic framework for simultaneous localization and map building, IEEE Transactions on Robotics and Automation 15 (1999) 948–952.
[8] A. Davison, Real-time simultaneous localisation and mapping with a single camera, in: Proceedings of the International Conference on Computer Vision, 2003.
[9] M. Eisenberg, R. Guy, A proof of the hairy ball theorem, The American Mathematical Monthly 86 (1979) 571–574.
[10] R. Fisher, Dispersion on a sphere, Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences 217 (1953) 295–305.
[11] U. Frese, L. Schröder, Theorie der Sensorfusion 2006, Universität Bremen, Lecture Notes #03-05-H-699.55, 2006.

[12] C.F. Gauss, Theoria combinationis observationum erroribus minimis obnoxiae, Commentationes Societatis Regiae Scientiarum Gottingensis Recentiores 5 (1821) 6–93.

[13] M.S. Grewal, L.R. Weill, A.P. Andrews, Global Positioning Systems, Inertial Navigation, and Integration, second ed., John Wiley, 2007.

[14] G. Grisetti, C. Stachniss, W. Burgard, Nonlinear constraint network optimization for efficient map learning, IEEE Transactions on Intelligent Transportation Systems 10 (2009) 428–439.

[15] G. Guennebaud, B. Jacob, et al., Eigen 2.0.15, 2010. <eigen.tuxfamily.org>.

[16] C. Hertzberg, A Framework for Sparse, Non-Linear Least Squares Problems on Manifolds, Master's thesis, Universität Bremen, 2008.

[17] S.J. Julier, J.K. Uhlmann, A General Method for Approximating Nonlinear Transformations of Probability Distributions, Technical Report, University of Oxford, 1996.

[18] S.J. Julier, J.K. Uhlmann, A new extension of the Kalman filter to nonlinear systems, in: International Symposium on Aerospace/Defense Sensing, Simulation and Controls, Orlando, FL, 1997, pp. 182–193.

[19] M. Kaess, A. Ranganathan, F. Dellaert, iSAM: incremental smoothing and mapping, IEEE Transactions on Robotics 24 (2008) 1365–1378. <openslam.org/iSAM>.

[20] R.E. Kalman, A new approach to linear filtering and prediction problems, Transactions of the ASME – Journal of Basic Engineering 82 (1960) 35–45.

[21] J.T. Kent, The Fisher–Bingham distribution on the sphere, Journal of the Royal Statistical Society. Series B, Methodological 44 (1982) 71–80.

[22] A.R. Klumpp, Apollo lunar-descent guidance, Automatica 10 (1974) 133–146.

[23] D. Koks, Explorations in Mathematical Physics, Springer, 2006.

[24] E. Kraft, A quaternion-based unscented Kalman filter for orientation tracking, in: Proceedings of the Sixth International Conference of Information Fusion, vol. 1, 2003, pp. 47–54.

[25] J. Kurlbaum, Verfolgung von Ballflugbahnen mit einem frei beweglichen Kamera–Inertialsensor, Master's thesis, Universität Bremen, 2007.

[26] J. Kurlbaum, U. Frese, A Benchmark Dataset for Data Association, Technical Report 017-02/2009, SFB TR/8, 2009.

[27] J.J. LaViola Jr., A comparison of unscented and extended Kalman filtering for estimating quaternion motion, in: Proceedings of the 2003 American Control Conference, vol. 6, 2003, pp. 2435–2440.

[28] J.M. Lee, Introduction to Smooth Manifolds, Graduate Texts in Mathematics, vol. 218, Springer Verlag, 2003.

[29] A.M. Legendre, Nouvelles méthodes pour la détermination des orbites des comètes, Courcier, Paris, 1805.

[30] J.L. Marins, X. Yun, E.R. Bachmann, R.B. Mcghee, M.J. Zyda, An Extended Kalman Filter for Quaternion-based Orientation Estimation Using MARG Sensors, Engineer's Thesis, Naval Postgraduate School, 2001, pp. 2003–2011.

[31] F.L. Markley, Y. Cheng, J.L. Crassidis, Y. Oshman, Quaternion Averaging, Technical Report, NASA Goddard Space Flight Center, 2007, #20070017872.

[32] R. van der Merwe, E. Wan, Sigma-point Kalman filters for probabilistic inference in dynamic state-space models, in: Proceedings of the Workshop on Advances in Machine Learning, Montreal, Canada, 2003.

[33] R. van der Merwe, E. Wan, S. Julier, Sigma-point Kalman filters for nonlinear estimation and sensor-fusion: applications to integrated navigation, in: AIAA Guidance, Navigation, and Control Conference and Exhibit, Providence, Rhode Island, 2004.

[34] C. Moler, C.F. Van Loan, Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later, SIAM Review 45 (2003) 3–49.

[35] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, Numerical Recipes, second ed., Cambridge University Press, Cambridge, 1992. pp. 681–699.

[36] B. Quine, J. Uhlmann, H. Durrant-Whyte, Implicit Jacobians for linearised state estimation in nonlinear systems, in: Proceedings of the American Control Conference, Seattle, WA, 1995, pp. 1645–1646.

[37] J. Schmidt, H. Niemann, Using quaternions for parametrizing 3D rotations in unconstrained nonlinear optimization, in: Proceedings of the Vision, Modelling and Visualization, 2001.

[38] B.J. Sipos, Application of the manifold-constrained unscented Kalman filter, in: Position, Location and Navigation Symposium, 2008 IEEE/ION, 2008, pp. 30–43.

[39] H. Strasdat, J. Montiel, A. Davison, Scale drift-aware large scale monocular slam, in: Proceedings of the Robotics Science and Systems Conference, 2010.

[40] S. Thrun, W. Burgard, D. Fox, Probabilistic Robotics, MIT Press, Cambridge, MA, 2005.

[41] S. Thrun et al., Stanley: the robot that won the DARPA grand challenge, Journal of Field Robotics 23 (2006).

[42] W. Triggs, P. McLauchlan, R. Hartley, A. Fitzgibbon, Bundle adjustment – a modern synthesis, in: W. Triggs, A. Zisserman, R. Szeliski (Eds.), Vision Algorithms: Theory and Practice, LNCS, Springer Verlag, 2000, pp. 298–375.

[43] R. Wagner, Autonomous Outdoor Navigation with a Robotic R/C Car, Master's thesis, Universität Bremen, 2010.

[44] R. Wagner, O. Birbach, U. Frese, Rapid development of manifold-based graph optimization for multi-sensor calibration and SLAM, in: Proceedings of the International Conference on Intelligent Robots and Systems, in press.

[45] M.D. Wheeler, K. Ikeuchi, Iterative Estimation of Rotation and Translation Using the Quaternion, Technical Report, Carnegie Mellon University, Pittsburgh, PA, 1995.