# Bridging Implementation and Theory: A Clean View on Parametrization of States with 3-D Rotations by Defining a Encapsulation Operator

Jörg Kurlbaum Udo Frese Christoph Hertzberg

Universität Bremen

Especially in 3D mapping but also in most other robotic applications you need to handle 3D poses that include orientations. Orientations are tricky because their representation is in the $SO(3)$ space, which is a manifold. In applications Euler angles are used frequently, but also axis-angle, matrices and quaternions are common for the representation of rotations and orientation in the specific state. The problem arises when you have to represent an orientation as a flat vector with independent variables, as needed for estimation algorithms such as the Kalman Filter family or other least square optimizers, which work by stepwise adding small rotations to the orientation state. All representation of orientation with three independent variables such as the Euler angles suffer from singularities. Indeed it can be proven that there is no representation of $SO(3)$ with only three parameters, that is singularity-free. For the other representations you need to handle their special characteristics to integrate them in the straight-forward algorithms. This implies that each algorithm has to be modified to actually handle orientations.

We propose a clean view onto the 3-manifold of rotations by encapsulating these special operations into new operators. These operators work for different representations (matrix, quaternion, ...), cleanly cover theory, and provide a basis for a concrete implementation.

While your state representation $S$ may contain any number of manifolds the numerical calculations work only in vector space $\mathbb{R}^n$. Changes to rotations are handled as small local changes that are added to the state. The operators we like to introduce are $\boxplus$ and $\boxminus$, which replace the vector based operations $+$ and $-$ on manifolds. We define as follows:

$$\boxplus : S \times \mathbb{R}^n \to S \tag{1}$$

$$\boxminus : S \times S \to \mathbb{R}^n \tag{2}$$

That means you can add a vector of a small change in rotation to the state with $\boxplus$ and conversely get a small rotation as a vector from two states with $\boxminus$. This defines an axiom for the interplay of $\boxplus$ and $\boxminus$:

$$s_1 \boxplus (s_2 \boxminus s_1) = s_2 \text{ with } s_n \in S \tag{3}$$

With this representation of operators, most algorithms can be extended to manifolds by simply replacing $+$ with $\boxplus$ and $-$ with $\boxminus$ as here for the sigma point generation in the UKF (see figure 1). When implementing states with manifolds, the introduction of this operators also leads to a clean view on the programming part. By implementing a state class variable with operations embedded you can write down the same algorithm for different representations of orientation. We used this view on orientation with success already in related research [1]. And we are pushing the idea even further to have a general least square optimizer framework using a manifold abstraction written in C++, which enables the user to define states with arbitrary mixtures of vectors and manifolds and make calculations based on the $\boxplus$ and $\boxminus$ operator. The most usual and most useful manifolds (matrix, quaternion) are readily usable. The use of particular language features (namely templates in C++) gives this approach even high-performance in runtime.

```
GENERATE-SIGMA-POINTS(μ, Σ)
1  sigmapoints[] ← μ
2  U ← CHOLESKY(Σ)
3  for i = 0 to N
4  do sigmapoints[] ← μ ⊞ U_{i-1}
5     sigmapoints[] ← μ ⊞ −U_{i-1}
6  return sigmapoints
```

**Figure 1:** Pseudo code for sigma point generation with manifolds in the UKF

## References

[1] Oliver Birbach. Accuracy analysis of camera-inertial sensor based ball-trajectory prediction. Master's thesis, Universität Bremen, 2008.