Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Erling Hjermstad

Target Tracking From a Moving Platform

Master's thesis in Cybernetics and robotics
Supervisor: Edmund F. Brekke
Co-supervisor: Torleiv H. Bryne
June 2024

NTNU
Norwegian University of
Science and Technology

Erling Hjermstad

Target Tracking From a Moving Platform

Master's thesis in Cybernetics and robotics
Supervisor: Edmund F. Brekke
Co-supervisor: Torleiv H. Bryne
June 2024

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

Abstract

Target tracking traditionally assumes the pose of the tracking sensor to be known. This is not the case when the sensor is mounted on a moving platform. In this thesis, two trackers that account for the uncertainty of the platform state are developed, based on an inertial navigation filter. A Lie-theoretic approach is taken, using a filter on the Lie group $\mathrm{SE}_2(3)$. The performance of the setup is compared to the popular error-state Kalman filter and to two naive trackers assuming the pose of the sensor to be known without uncertainty.

Simulations show that the developed trackers are more accurate and consistent compared to the error-state Kalman filter, and far more consistent than their naive counterparts. Consistency is an important property if data association is required. The tracker parameterizing the targets in the body frame outperforms the tracker using a world parameterization on both the RMSE and the NEES metrics.

The aim of this work is to lay a foundation for using an on-manifold approach to the joint inertial navigation and target tracking problem. To this end, the trackers are derived from an on-manifold approach, but some natural concepts are left for future work. Some reflections on the approach to integrate inertial navigation and target tracking, keeping a joint state vector, are offered.

Sammendrag

Den normale tilnærmingen til sporing av bevegelige objekter antar at posisjonen og orienteringen til sporingssensoren er kjent. Dette er ikke tilfellet når sensoren er montert på en bevegelig plattform. I denne oppgaven utledes to sporingsalgoritmer som tar hensyn til usikkerheten i tilstanden til plattformen. Usikkerheten er basert på estimater fra et navigasjonssystem, som også utledes. Lie-teori og Lie-gruppen $SE_2(3)$ benyttes til dette formålet. Ytelsen av oppsettet sammenlignes med det populære feiltilstands Kalmanfilteret, og to naive sporere som antar at tilstanden til plattformen er kjent uten usikkerhet.

Simuleringer viser at de utledede sporingsalgoritmene er mer presise og konsekvente enn feiltilstands Kalmanfilteret. De er også langt mer konsekvente enn de tilsvarende naive sporerne. Konsekvente estimater er viktig for applikasjoner hvor data-assosiering er nødvendig. Sporeren som parameteriserer objekter i et lokalt koordinatsystem presterer bedre enn den som parameteriserer i et globalt koordinatsystem på både RMSE og NEES metrikkene.

Dette arbeidet sikter på å legge et grunnlag for en Lie-teoretisk fremgangsmåte til det kombinerte navigasjons- og sporingsproblemet. Av denne grunn er sporingsalgoritmene utledet med en Lie-teoretisk tilnærming, men noen naturlige konsepter er overlatt til fremtidig arbeid. Noen refleksjoner om mulige tilnærminger for å integrere navigasjons- og sporingsproblemet, slik at en felles tilstandsvektor estimeres, tilbys.

Table of Contents

List of Figures

List of Tables

1 Introduction

Target tracking is an important topic in autonomous solutions that provides data for collision avoidance algorithms, navigation systems, and situational awareness. Having a correct understanding of the whereabouts of targets in a world frame is necessary for navigation using global data or collaboration. This is especially challenging when targets are moving. Using sensor fusion, uncertain measurements and estimates can be fused into optimal probability distributions over the position and velocity of targets. This probability distribution depends on the state of the moving platform from which the targets are tracked.

In this work, an on-manifold inertial navigation filter is derived, along with two trackers compensating for the uncertainty of the navigation filter. The performance of the trackers is compared to two baseline trackers that ignore the uncertainty of the state of the platform. Capturing the rotational uncertainty of the platform receives much attention. This is not unfounded, as uncertain orientation will influence the entire system to a greater degree than uncertain position and velocity. This is because measurements from the sensors on the platform must be rotated into the world frame before being used. This rotation depends on the orientation of the platform.

Target tracking encompasses the tasks of filtering measurements and data association. Filtering is the focus of this work, as a good understanding of the estimates is paramount for data association to work properly, defending the prioritization of the filtering problem. Traditionally, tracking has been done with sensors whose position and orientation is known.[1], [2] When sensors are mounted on a moving platform, the motion of the platform must be taken into account. This makes the pose of the sensors a stochastic variable. Errors in the orientation, position, and velocity estimates of the platform will affect the accuracy of the targets' position and velocity estimates. The area of moving target tracking has seen a lot of research in recent years. The problem can be formulated as a simultaneous localization and mapping (SLAM) problem with moving landmarks.[3], [4] Others take a probabilistic approach, formulating the problem as a filtering task using random finite set theory.[5] Theoretically, SLAM should be the optimal approach, solving the estimation problem jointly, making navigation and tracking affect each other. Allowing target measurements to affect the navigation system can be catastrophic if the motion of the targets is modeled poorly or the data association fails.[6] To avoid this, a popular approach is to only let the navigation affect the targets, not the other way around. A Schmidt-Kalman filter does this while keeping the correlations between the platform and targets.[7] A less optimal but more efficient approach is to neglect the correlations entirely.

The problem is two-fold. The state of the platform must be estimated, as well as the state of the targets. Estimation of the platform's own state is called navigation. Navigation is typically done with an error-state Kalman filter (ESKF) based on measurements from an inertial measurement unit (IMU).[8] For there to be any hope of good tracker performance, the inertial navigation must be consistent and accurate. Poor inertial navigation performance propagates into poor tracker performance, as the tracker is on the platform. Using the Lie group $\mathrm{SE}_2(3)$ for inertial navigation is an approach recently proposed.[9] The group allows the application of the invariant extended Kalman filter equations to the inertial navigation problem.[10], [11] Using Lie groups not only for inertial navigation, but also for tracking has not been done much. [12] treats the full SLAM problem from a Lie theoretic viewpoint, but without the generalization to moving landmarks.

This work aims to take some steps towards unifying manifold theory and moving object tracking, allowing for integrating inertial navigation concepts, such as pre-integration, into tracking applications. The main contributions of the thesis include a thorough review of the derivation of the extended pose $\mathrm{SE}_2(3)$ filter, where some of the finer details are examined to a greater degree than in [9]. This includes developing the update step of the filter with GNSS updates, combining the structure of the IEKF of [11] with the reset of the ESKF from [8]. A formulation of the ESKF from the same Lie-theoretic viewpoint of the $\mathrm{SE}_2(3)$ filter is offered. Two trackers are developed, both taking into account the distribution of the inertial navigation estimates on the exponential coordinates of the Lie group. This includes developing motion and measurement models that compensate for the movement of the platform. A novel approach to converting body frame quantities to the world frame, using the tangent space of the Lie group, is presented. The implementation of

the full inertial navigation and tracking filter is available on Github.[1]

This thesis follows a traditional structure with a slight modification. The work in this thesis is split between two main topics; the inertial navigation and the target tracking. For clarity, the two topics are introduced separately. A consequence of this is that some sections are split in two. First, inertial navigation is presented, then tracking, since the tracking is based on the estimates of the inertial navigation. This is the case for the theory and results sections.

An overview of the notation follows immediately after the introduction. Then the relevant theory is presented, where an attempt is made to offer some intuition for the mathematics. The theory includes the derivation of an inertial navigation filter, where a fairly thorough introduction to Lie theory is made. Lie theory is a mathematical tool which is used throughout the thesis. The basic building blocks of tracking theory are then presented, before two target trackers are derived. In the derivation of the filter, several choices must be made. The effect of the choices is studied in the following results sections.

The work in this thesis leaves some interesting topics unexplored and instead aims to build a foundation for the on-manifold approach to target tracking from a moving platform. Some remarks on possible extensions of the work is made in the future research section, before the thesis is concluded.

[1] https://github.com/erlingh99/masteroppgave

2 Notation

The notation follows that of [12] closely. Vectors are in bold. Bold and capitalized letters indicate matrices. Calligraphic font will be used for Lie elements, but matrix Lie elements will use the matrix notation.

$\mathcal{N}(x; \mu, \sigma^2)$	x is normal distributed with mean μ and variance σ^2.
$\mathcal{N}_{\mathcal{G}}(\boldsymbol{T}; \hat{\boldsymbol{T}}, \boldsymbol{\Sigma})$	$\boldsymbol{T}$ is Exponentially normally distributed with mean $\hat{\boldsymbol{T}}$ and variance $\boldsymbol{\Sigma}$.
$\mathcal{M}$	Generic Lie manifold
$\mathcal{X}$	Generic Lie group element
$\mathcal{J}$	The Jacobian of the Lie group, comes in both left and right variants.
$\boldsymbol{J}_x^f$	The Jacobian of the function f with respect to the variable x.
$\boldsymbol{T}$	The pose of the platform, a matrix Lie group random variable.
$\boldsymbol{\delta T}$	The error state, an element of the tangent space of $\hat{\boldsymbol{T}}$
$\delta(\cdot)$	Some small quantity, e.g. the length of a time step δt
$(\cdot)_{k\|k-1}$	Quantity at time step k, given measurements up to time step $k-1$.
$(\cdot)_k$	Quantity at time step k, given measurements up to time step k.
$\hat{(\cdot)}$	Prediction of random variable, the mean of the Gaussian distribution of the variable.
$\boldsymbol{z}_k$	Platform measurement received at time step k.
$\boldsymbol{z}_{1:k}$	The set of platform measurements received at time steps 1 through k.
$\boldsymbol{y}_k$	Target measurement received at time step k.
$\boldsymbol{y}_{1:k}$	The set of target measurements received at time steps 1 through k.
$\boldsymbol{x}_k$	State of a target at time step k.
$\boldsymbol{F}^{\text{CV}}$	CV kinematic propagation matrix.
$\boldsymbol{\Gamma}$	CV kinematic propagation covariance.
$\boldsymbol{Q}$	Inertial navigation propagation covariance.
$\boldsymbol{H}$	Measurement matrix or measurement Jacobian
$\boldsymbol{R}$	Rotation matrix, the orientation state of the platform
$\boldsymbol{p}$	The position of the platform or target
$\boldsymbol{v}$	The velocity of the platform or target
$\boldsymbol{\phi}$	A rotation in the tangent space of $\mathrm{SE}_2(3)$
$\boldsymbol{\nu}$	A velocity in the tangent space of $\mathrm{SE}_2(3)$
$\boldsymbol{\rho}$	A position in the tangent space of $\mathrm{SE}_2(3)$
$\boldsymbol{P}$	State uncertainty covariance of the target
$\boldsymbol{\Sigma}$	State uncertainty covariance of the platform

Table 1: An overview of most of the notation used in this work.

Depending on the context, different formulations for the same quantities may be used. The platform is the object tracking targets, however, before target tracking receives focus in this thesis, the inertial navigation of the platform is explored. In this context, the platform will interchangeably be referred to as the body, object, or platform. The state is the matrix $\boldsymbol{T}$ for the platform and the vector $\boldsymbol{x}$ for the targets. The word pose is also used for this. Rotation receives much attention in this work. Orientation would probably be a better word, as it does not mean the act of rotating, but rather the heading of the object. However, since rotation matrices are used to describe the orientation, both words are used interchangeably.

Although both the inertial navigation and tracking will be done with probabilistic filters, the term filter will mostly be used to refer to the inertial navigation filter architecture. The term tracker will be used for the tracking architecture, or the combined inertial navigation filter and tracker. In some cases, the term filter will be used for the tracker, but this should be clear from context.

When working with Lie theory, the concept of tangent space is important. Due to the mapping between the tangent space and the Lie group, the tangent space will also be denoted the Exponential coordinates. A Gaussian distribution on the tangent space is then an Exponential Gaussian. For short, this will sometimes be referred to as an Exponential distribution, but must not be confused with the usual exponential distribution, which is never used in this work. The Exponential

coordinates will also equivalently be called the error coordinates.

In this work, the notation for random variables will be confused with their realizations. For example, $\boldsymbol{x} \sim f(\boldsymbol{x})$ might be written for the random variable X with probability distribution $f(\boldsymbol{x})$. This is for notational simplicity, as the other approach would quickly become too verbose.

The acronyms used in this work are summarized in the following Table 2.

IMU	Inertial Measurement Unit
EKF	Extended Kalman Filter
ESKF	Error State Kalman Filter
IEKF	Invariant Extended Kalman Filter
SLAM	Simultaneous Localization And Mapping
NEES	Normalized Estimation Error Squared
ANEES	Average Normalized Estimation Error Squared
RMSE	Root Mean Square Error
SO(3)	Special Orthogonal group in 3 dimensions
SE(3)	Special Euclidean group in 3 dimensions
CV	Constant Velocity
GNSS	Global Navigation Satellite System

Table 2: Acronyms used in this work

3 Inertial navigation theory

In this section, relevant theory for motion estimation is presented and a inertial navigation filter is derived. The choices made in the design of the system are motivated, and the equations governing the estimator are derived.

The derivations in this chapter use Lie theory. The relevant Lie theory is presented, but before that an introduction to why using the theory is convenient is presented.

3.1 Why rotations are difficult

Velocity, position and other quantities of vector spaces can easily be added, averaged, or inverted.[2] For rotations this is not as straightforward owning to the fact that rotations can wrap-around, meaning an angle is only unique modulo 2π. This must be considered when doing calculations with rotations. Consider Figure 1, rotating from the blue to the green dot can be done in two ways, clockwise or counterclockwise. On the circle describing 2D rotations both are continuous paths, but identifying the angle with a one-to-one mapping onto the half open interval $[0, 2\pi)$ gives only one continuous path. The other path is discontinuous when exiting the interval where it jumps from 0 to 2π. This discontinuity must not be forgotten when doing calculations on the interval, as it does not follow automatically on the real line. An example where this issue is present is averaging the angles on the interval. The average of the blue and green dots gives the orange dot, but on the circle the red dot is clearly an equally valid answer. It might even be the preferred solution as it is closer to both dots. Calculating rotations as they were elements of the real line can give misleading answers as the wrapping around effect is not preserved.

An idea is to use a different type of representation of the rotation, which does not suffer from the discontinuity of the angle interval. In 2D there are two main methods. A 2×2 rotation matrix or a unit complex number. Both are *homeomorphic* to the unit circle, meaning there exists bijective mappings back and forth preserving continuity of paths. That is, wrap-around is not an issue. This was the issue with the map from the unit circle to the interval $[0, 2\pi)$. Both representations are over-parameterized, with respectively 4 and 2 coordinates for an inherently 1 dimensional quantity. The degrees of freedom are restricted through, respectively, 3 orthonormality constraints and a unit constraint. Through matrix multiplication and complex multiplication rotations can be composed and vectors can be rotated.[3] This also differ these representation from the angle; they can act on

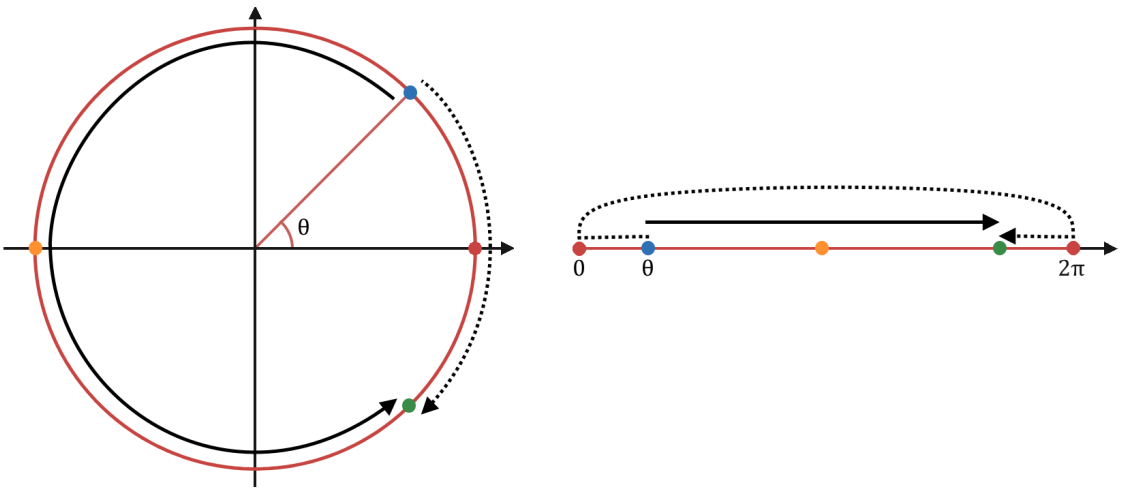


Figure 1: Is the red or orange dot the average of the blue and green dots?

[2] Inverting meaning the corresponding negative element for vector spaces.

[3] In 2D, defining composition for the angle parametrization is actually straight forward with summation of angles modulo 2π, however it does not generalize to higher dimensions.

other vectors. The action is a rotation described by the rotation matrix or unit complex number. The fact that they can be used to both rotate and describe an orientation is practical.

In three dimensions, the issue is perhaps a bit more tricky. A natural thought might be to identify rotations on the unit sphere, like the unit circle in 2D, however this is not possible. Euler's rotation theorem states that every rotation can be defined by a unit axis and an angle.[13] The unit axis lives on the sphere, but in addition an angle is necessary, meaning more than the sphere is needed to describe a rotation. Intuitively, this makes sense. In 2D there is only one way to rotate, but in 3D a rotation can be in any direction, meaning we have gained 2 degrees of freedom. Euler's theorem can be thought of as defining a plane through the origin with the axis as a normal, and then doing a 2D rotation on that plane. In fact, this shows that the angle-axis representation in 3D is the analogy to the angle representation in 2D, meaning the wrap-around issue exists for the angle-axis. Using the length of a vector as the angle, and the direction as the unit axis, axis-angle is a minimal parametrization, with 3 parameters for 3 degrees of freedom. The domain is the inside of a half open ball of radius π. Another minimal parameterization is Euler angles, however it suffers from a singularity known as gimbal lock, causing loss of a degree of freedom under certain motions.[14] This makes it unsuited.

Similarly to 2D, a 3×3 rotation matrix can be used. Analogous to the unit complex numbers are the unit quaternions. Respectively 9 and 4 coordinates are used in these parameterizations, with 6 and 1 constraints leaving 3 degrees of freedom. The gain of using rotation matrices or unit quaternions is the lack of singularities. In addition, composition of rotations is easy through matrix or quaternion multiplication. This follows easily from the fact that they can act on vectors. Axis-angle can not do this, only represent an orientation. Composing two rotations parameterized with axis-angle is not straight-forward, and converting to either rotation matrix or quaternion, composing, and converting back is often the way to go.

In estimation, calculating the uncertainty on the estimates is critical. The uncertainties encode the confidence on the estimates and enable fusing information reliably. When dealing with uncertainties, over-parameterization can be an issue. We might easily end up with some probability mass for objects not fulfilling the constraints on the over-parameterization. Artificially enforcing the constraints can give singular covariance matrices.[14]

To summarize, axis-angle is a concise, minimal way to describe an orientation, but is difficult to do compositions and actions with. It is however vectorial, meaning the usual mathematical tools, like differentiation and probability theory, can be used. Rotation matrices or quaternions are over-parameterized, but are suited for compositions as they are singularity free and can act on vectors. A combination of both will be shown to give a "best-of-both worlds" for mathematical applications. In this work, rotation matrices are used, but all results are valid for quaternions, and they can therefore easily be inserted in place of the rotation matrices in implementations.

3.2 Lie Theory

Lie theory is useful for abstracting vector space concepts to non-vector quantities. A vector space is a set, a collection of unique mathematical objects, whose elements can be added and scaled. An example is the set of all real numbers in three dimensions $\mathbb{R}^3$. Combining the set with the usual addition and scalar multiplication makes it a vector space. Not all constructs used in mathematics have these properties. In robotics, rotation matrices are the first and foremost example of this. In contrast to vector quantities, like velocities and positions, the sum of two rotation matrices is generally not a rotation matrix. Lie theory provides the tools necessary to generalize the usual operations, like addition and differentiation, to Lie groups.

To see that rotation matrices is not a vector space is straight-forward. A square matrix is a rotation matrix if it fulfills the following orthonormality constraint $\boldsymbol{R}\boldsymbol{R}^\top = \boldsymbol{R}^\top\boldsymbol{R} = \boldsymbol{I}$, and the special constraint $\det \boldsymbol{R} = 1$.[14] The sum of any two rotation matrices does not generally fulfill either constraint,

$$(\boldsymbol{R}_1 + \boldsymbol{R}_2)^\top(\boldsymbol{R}_1 + \boldsymbol{R}_2) \neq \boldsymbol{I},$$
$$\det(\boldsymbol{R}_1 + \boldsymbol{R}_2) \neq 1.$$

Choosing for example $\boldsymbol{R}_1 = \boldsymbol{R}_2$ shows this.

Rotation matrices are an example of a Lie group and are known as the special orthogonal group, $SO(n)$. The name refers to the constraints of the group, and the n refers to the number of dimensions. Formally, a group $(\mathcal{G}, \circ)$ is a set $\mathcal{G}$ with a composition operation $(\circ)$ which fulfills the group axioms.[15] The axioms are summarized in Table 3, where $\mathcal{X}, \mathcal{Y}, \mathcal{Z} \in \mathcal{G}$. For rotation matrices the composition operation is normal matrix multiplication. The inverse is the matrix transpose, and the identity element is the identity matrix.

Closure	$\mathcal{X} \circ \mathcal{Y} \in \mathcal{G}$
Identity	$\mathcal{E} \circ \mathcal{X} = \mathcal{X} \circ \mathcal{E} = \mathcal{X}$
Inverse	$\mathcal{X}^{-1} \circ \mathcal{X} = \mathcal{X} \circ \mathcal{X}^{-1} = \mathcal{E}$
Associativity	$(\mathcal{X} \circ \mathcal{Y}) \circ \mathcal{Z} = \mathcal{X} \circ (\mathcal{Y} \circ \mathcal{Z})$

Table 3: The group axioms

The closure axiom states that the possibly nonlinear composition operation is closed on the group, meaning it generates new elements in the group, $(\circ : \mathcal{G} \times \mathcal{G} \to \mathcal{G})$. It is important to note that it is generally not commutative, $\mathcal{X} \circ \mathcal{Y} \neq \mathcal{Y} \circ \mathcal{X}$. The identity axiom requires an identity element in the group, which is the same for both left- and right-composition. The inverse axiom says that for every element in the group there exists an inverse element, also in the group. It is this axiom that encodes the constraints of the group. Inserting the correct operations for rotation matrices, the constraint degenerates to exactly the orthonormality constraint presented earlier. Lastly, the composition operation is associative, meaning the order of operations is unimportant.

A Lie group is a group, but also a smooth manifold.[12] A smooth manifold is a space locally resembling linear space everywhere. Intuitively, this is all topological spaces where a tangent hyperplane is defined at every point. The geometry of the manifold is defined by the inverse axiom in Table 3. An example is the surface of a sphere. Locally the surface resembles linear space (with the unfortunate consequence some people believe the Earth is flat), and tangent planes exist at all points. The shape of the sphere is encoded in the constraint of the group. For example, the rotation group in two dimensions $SO(2)$ live on a subspace of $\mathbb{R}^{2\times 2}$. Inserting the correct operations for rotation matrices, the inverse axiom constraint degenerates to exactly the orthogonality constraint presented earlier. From here, finding the subspace is straight forward.

$$\begin{aligned}\boldsymbol{I} = \boldsymbol{R}^\top \boldsymbol{R} &= \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} a & c \\ b & d \end{bmatrix} \\ &= \begin{bmatrix} a^2 + b^2 & ac + bd \\ ac + bd & c^2 + d^2 \end{bmatrix} \Rightarrow \begin{cases} a^2 + b^2 = 1 \\ c^2 + d^2 = 1 \\ ac + bd = 0 \\ ad - bc = 1 \quad (\det \boldsymbol{R} = 1) \end{cases}\end{aligned}$$

This means (a, b) and (c, d) each lie on a circle with radius 1 in $\mathbb{R}^2$. Using the cartesian to polar transform the coefficients can be written as $a = \cos\theta, b = \sin\theta, c = \cos\phi$, and $d = \sin\phi$, $\theta, \phi \in \mathbb{R}$. Inserting this into the last two equations lets us relate θ and ϕ through some trigonometric identities.

$$\begin{aligned}\cos\theta\cos\phi + \sin\theta\sin\phi &= \cos(\phi - \theta) = 0 \\ \cos\theta\sin\phi - \sin\theta\cos\phi &= \sin(\phi - \theta) = 1 \\ \Rightarrow \phi - \theta &\in \pi/2 + 2\pi\mathbb{Z}\end{aligned}$$

Finally, inserting $\phi = \theta + \pi/2$ gives the well-known 2D rotation matrix formula.

$$\boldsymbol{R} = \begin{bmatrix} a & c \\ b & d \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}.$$

This shows that SO(2) is a closed curve in $\mathbb{R}^{2\times 2}$ parameterized by an angle θ. For this reason, SO(2) is homeomorphic to the circle group S^1 of the unit complex numbers, used for rotation of complex numbers. This proves that 2D rotations can be identified with the unit circle, as done in

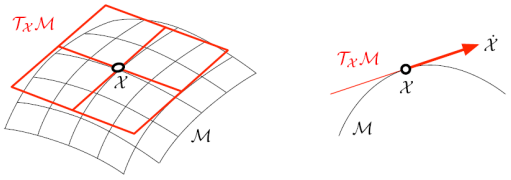


Figure 2: A manifold $\mathcal{M}$ of a Lie group $\mathcal{G}$, and the tangent space $\mathcal{T}_{\mathcal{X}}\mathcal{M}$ at the element $\mathcal{X}$. Here, the tangent plane is homeomorphic to $\mathbb{R}^2$, $\mathcal{T}_{\mathcal{X}}\mathcal{M} \cong \mathbb{R}^2$. Image from [12].

the previous section. This simple example actually hints about the power of Lie theory. As stated previously, rotation matrices are not addable, but we can see that the 2D rotation matrices are defined by a single parameter θ, the angle of rotation, which *is* addable. Lie theory generalizes this concept allowing finding complicated mathematical structures from a minimal vector space parametrization. The function from this "natural" parametrization to the manifold, is called the Exponential map (with capitalized E) for reasons which will be made clear.[12] Going back to the example of the introduction, the Exponential map is the map from the real line onto the circle. This is related to the tangent hyper-planes of the manifold.

Figure 2 illustrates a Lie group together with one of its tangent hyper-planes, viewed from two different angles. Elements $\mathcal{X}$ of a Lie group live on a manifold $\mathcal{M}$. The curvature of the manifold illustrates that the Lie group is not a vector space, as adding two elements would produce an element not on the manifold, and therefore not in the group. In contrast, the tangent hyper-plane $\mathcal{T}_{\mathcal{X}}\mathcal{M}$ is a vector space since it is linear. In linear space all the usual operations of calculus are defined.[16] The combination of the properties of a smooth manifold and a group is what makes Lie theory useful. The smooth manifold allows calculus to be applied locally, whilst the group captures the nonlinear properties in the composition operation.

The manifold has a possibly infinite number of equivalent tangent planes, each tangent to the manifold at an element of the Lie group. Since the identity element of a group is special, one of the tangent hyper-planes is too. The tangent hyper-plane at the identity element is called the Lie algebra $\mathfrak{m}$ of the Lie group. To illustrate the applications of the Lie algebra, some derivations are needed. In similar fashion to finding tangents in calculus, the tangent hyper-planes are derived through differentiation. Since the inverse axiom defines the structure of the manifold the group elements evolve on, the tangent hyper-planes can be found by differentiating it with respect to time. For this reason, the tangent hyper-planes are also known as the velocity space. For groups where the composition operation has multiplicative nature, meaning the product rule of differentiation applies, the constraint on elements tangent to the manifold at $\mathcal{X}$ is

$$\begin{aligned} \frac{d}{dt}\mathcal{X}^{-1} \circ \mathcal{X} &= \frac{d}{dt}\mathcal{E} \\ \Rightarrow \dot{\mathcal{X}^{-1}} \circ \mathcal{X} + \mathcal{X}^{-1} \circ \dot{\mathcal{X}} &= 0. \end{aligned}$$

This is the constraint defining the tangent $\dot{\mathcal{X}}$ at $\mathcal{X}$. This makes $\mathcal{X}^{-1} \circ \dot{\mathcal{X}}$ the tangent at the origin, the Lie algebra. Using $\mathcal{X}^{-1} \circ \dot{\mathcal{X}} = -\dot{\mathcal{X}^{-1}} \circ \mathcal{X} = \boldsymbol{v}$, defining $\boldsymbol{v}$ as an element of the Lie algebra, an interesting differential equation appears.

$$\begin{aligned} \mathcal{X}^{-1} \circ \dot{\mathcal{X}} &= \boldsymbol{v} \\ \Rightarrow \dot{\mathcal{X}} &= \mathcal{X} \circ \boldsymbol{v} \end{aligned} \tag{1}$$

Inserting $\mathcal{X} = \mathcal{E}$ at the identity, we get $\dot{\mathcal{X}} = \boldsymbol{v}$, showing that $\boldsymbol{v}$ is in the Lie algebra. Assuming constant $\boldsymbol{v}$, the solution to the differential Equation 1 is

$$\mathcal{X}(t) = \mathcal{X}(0) \circ \exp(\boldsymbol{v}t). \tag{2}$$

Since both $\mathcal{X}(t)$ and $\mathcal{X}(0)$ are elements of the group, $\exp(\boldsymbol{v}t)$ must be too, by the closure of the group composition operation. Noting that $\boldsymbol{v}$ is an arbitrary element of the Lie algebra, then so is $\boldsymbol{v}t$ because the algebra is a vector space. This shows that the exponential map is an exact transform

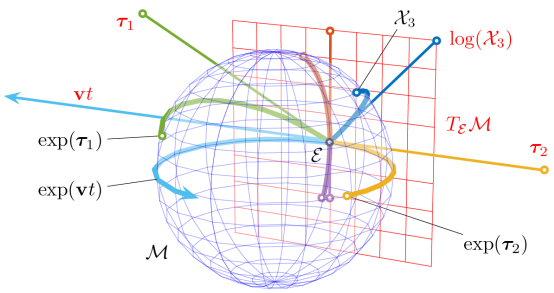


Figure 3: Elements of the Lie algebra are wrapped onto the manifold by the exponential map, and unwrapped by the logarithmic map. Image from [12].

from the Lie algebra to the manifold. Intuitively, the exponential map wraps an element of the algebra onto the manifold. The logarithmic map does the opposite operation. Figure 3 illustrates how the maps go from the Lie algebra to the manifold and back.

$$\exp : \mathfrak{m} \to \mathcal{M}$$
$$\log : \mathcal{M} \to \mathfrak{m}$$

The Lie algebra elements are usually non-trivial structures. However, since they are vector quantities there exists an homeomorphism from the Lie algebra to $\mathbb{R}^n$, where n is the number of degrees of freedom of the manifold. In Lie theory, these maps between the algebra and the reals are known as *hat* and *vee*. They follow from the fact that vector space elements can be expressed as a linear combination of a basis of the vector space, $\boldsymbol{\tau} = \sum_i \tau_i G_i \in \mathfrak{m}$. The $G_i \in \mathfrak{m}$ are known as the generators of the Lie algebra. Simply swapping the generators for the basis of $\mathbb{R}^n$ is the vee map defining the corresponding real vector element; $\boldsymbol{\tau}^\vee = \sum_i \tau_i \boldsymbol{e}_i = [\tau_1, \ldots, \tau_n]^\top \in \mathbb{R}^n$. The hat map does the opposite operation.

$$(\cdot)^\wedge : \mathbb{R}^n \to \mathfrak{m}$$
$$(\cdot)^\vee : \mathfrak{m} \to \mathbb{R}^n$$

Combining the hat and exp, and vee and log functions allow us to define functions directly from the reals to the manifold and back. These variations of the exponential and logarithmic maps will be denoted with a capital first letter. Using these functions make it possible to skip the Lie algebra, and going forward they will be used almost exclusively.

$$\mathrm{Exp} : \mathbb{R}^n \to \mathcal{M};\ \mathrm{Exp}\,\boldsymbol{\tau} := \exp(\boldsymbol{\tau}^\wedge)$$
$$\mathrm{Log} : \mathcal{M} \to \mathbb{R}^n;\ \mathrm{Log}\,\mathcal{X} := \log(\mathcal{X})^\vee$$

The Exponential map is usually defined through the Taylor expansion

$$\mathrm{Exp}\,\boldsymbol{\tau} = \mathcal{E} + \boldsymbol{\tau}^\wedge + \frac{1}{2}\boldsymbol{\tau}^{\wedge 2} + \ldots. \tag{3}$$

and the Logarithmic map as the inverse of the Exponential map. For the groups considered in this work, and most robotics applications, there exists closed form solutions to both.[12] An important thing to note is that the normal rules for exponentials does not apply when the powers are not scalar. Generally, $\exp(\boldsymbol{x}+\boldsymbol{y}) \neq \exp\boldsymbol{x} \circ \exp\boldsymbol{y}$. This is easily seen by noting that this equality would force $\circ$ to be commutative, since addition is. Since $\circ$ is not commutative, the expression does not hold.

For additive groups the inverse constraint is the trivial $\mathcal{X} - \mathcal{X} = 0$, meaning there is no constraint and the manifold is its own tangent plane. This makes the Exponential and Logarithmic maps identity maps.

Applying the outlined derivations above to SO(n) give the Lie algebra $\mathfrak{so}(n)$ and the hat and vee

maps.

$$\begin{aligned}
\boldsymbol{R}^\top \boldsymbol{R} &= \boldsymbol{I} \\
\boldsymbol{R}^\top \dot{\boldsymbol{R}} + \dot{\boldsymbol{R}}^\top \boldsymbol{R} &= \boldsymbol{0} \\
\boldsymbol{R}^\top \dot{\boldsymbol{R}} &= -(\boldsymbol{R}^\top \dot{\boldsymbol{R}})^\top = \boldsymbol{\Omega} \\
\dot{\boldsymbol{R}} &= \boldsymbol{R}\boldsymbol{\Omega} \\
\boldsymbol{R}(t) &= \boldsymbol{R}(0)\exp(\boldsymbol{\Omega} t)
\end{aligned}$$

The key is to notice that the third relation shows that $\boldsymbol{\Omega}$ is a skew symmetric matrix. This means the Lie algebra consists of skew symmetric matrices. In the case of $\mathfrak{so}(2)$ there are two skew symmetric variations to choose from, $\left[\begin{smallmatrix} 0 & \theta \\ -\theta & 0 \end{smallmatrix}\right]$ and $\left[\begin{smallmatrix} 0 & -\theta \\ \theta & 0 \end{smallmatrix}\right]$, corresponding to clockwise and counterclockwise rotations. Keeping with convention, the counterclockwise variant is chosen. Notice $\left[\begin{smallmatrix} 0 & -1 \\ 1 & 0 \end{smallmatrix}\right]$ is a skew symmetric matrix corresponding to a 90 degree counterclockwise rotation. This is the generator of the Lie algebra $\mathfrak{so}(2)$. At this point, defining the hat and vee maps is trivial.

$$\begin{aligned}
\theta^\wedge &= \begin{bmatrix} 0 & -\theta \\ \theta & 0 \end{bmatrix} \\
\begin{bmatrix} 0 & -\theta \\ \theta & 0 \end{bmatrix}^\vee &= \theta
\end{aligned}$$

Finding the closed form of the SO(2) exponential map is done by inserting the hat and vee maps above into Equation 3.

$$\begin{aligned}
\operatorname{Exp}\theta &= \exp\theta^\wedge \\
&= \sum_k \frac{\theta^k}{k!} 1^{\wedge k} = \boldsymbol{I}_2 \underbrace{(1 - \frac{\theta^2}{2!} + \frac{\theta^4}{4!} - \ldots)}_{\cos\theta} + 1^\wedge \underbrace{(\theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} - \ldots)}_{\sin\theta} \\
&= \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \in SO(2)
\end{aligned}$$

where the pattern the different powers of $1^\wedge$ create is used. It turns out the Exponential map of SO(2) is exactly the equation found earlier when analyzing the constraints. Analyzing the constraints quickly becomes more convoluted in more dimensions, but the Lie theory approach is similar. For SO(3) there are many more skew symmetric variations to choose from. Again, the rotation should obey the right hand rule. The correct choice follows from the connection between a specific skew symmetric matrix and the cross product.

$$\boldsymbol{\omega} \times \boldsymbol{x} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \boldsymbol{x} = \boldsymbol{\omega}_\times \boldsymbol{x}$$

where the subscript $(\cdot)_\times$ denotes the skew symmetric matrix equivalent to the right cross product. Defining $\boldsymbol{\Omega} = \boldsymbol{\omega}_\times$ gives the relation $\dot{\boldsymbol{R}} = \boldsymbol{R}\boldsymbol{\omega}_\times$. This is the well-known differential equation describing the time evolution of a rotation matrix when the vector $\boldsymbol{\omega}$ is the angular velocity.[14] The Exponential map is $\boldsymbol{R} = \exp\boldsymbol{\omega}_\times t = \operatorname{Exp}\boldsymbol{\omega} t$. Since $\boldsymbol{\omega}_\times$ is the Lie algebra, it is straight forward to see that the natural choice of the hat map is the vector to cross matrix map $(\cdot)^\wedge = (\cdot)_\times$. There are three generators of algebra. They are

$$\begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} = \omega_x \left[\begin{smallmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{smallmatrix}\right] + \omega_y \left[\begin{smallmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{smallmatrix}\right] + \omega_z \left[\begin{smallmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{smallmatrix}\right]. \tag{4}$$

Finding the closed form solution of the Exponential map is again an exercise in pattern recognition. Define the angle-axis form $\theta\boldsymbol{u} = \boldsymbol{\omega} t$, $||\boldsymbol{u}||_2 = 1$, find the pattern in the powers of $\boldsymbol{u}_\times$, and then identify the sine and cosine expansions. The resulting formula is the Rodrigues' rotation formula for converting angle-axis to rotation matrix.[17]

$$\boldsymbol{R} = \operatorname{Exp}(\theta\boldsymbol{u}) = \boldsymbol{I} + \boldsymbol{u}_\times \sin\theta + \boldsymbol{u}_\times^2 (1 - \cos\theta) \tag{5}$$

The Logarithmic map is found by inverting the Exponential map.

$$\begin{aligned} \theta &= \cos^{-1} \frac{\text{trace}(\boldsymbol{R}) - 1}{2} \\ \boldsymbol{u} &= \frac{(\boldsymbol{R} - \boldsymbol{R}^\top)^\vee}{2 \sin \theta} \end{aligned} \tag{6}$$

The domain of the Exponential map of Equation 5 is all of $\mathbb{R}^3$, it is not restricted to the half open ball of radius π on which axis-angle is properly defined. The sine and cosine in the formula takes care of the wrap-around issue discussed in the introduction of this chapter, making rotation matrices singularity free. The Logarithmic map, however, maps onto the ball. These tools allow us to relate vector calculations with the manifold.

The last property of Lie groups which will be used in this work is the action. A Lie group can act on elements from a set X, $(\cdot : \mathcal{G} \times X \to X)$. For example, $\text{SO}(n)$ can act on vectors of $\mathbb{R}^n$ through matrix multiplication. The resulting vector is a rotation of the original vector as described by the rotation matrix. Action can be defined in both left and right variants. Left is used in this work. A left group action follows two axioms, summarized in Table 4.[18]

Identity	$\mathcal{E} \cdot \boldsymbol{x} = \boldsymbol{x}$
Compatibility	$(\mathcal{X} \circ \mathcal{Y}) \cdot \boldsymbol{x} = \mathcal{X} \cdot (\mathcal{Y} \cdot \boldsymbol{x})$

Table 4: The axioms of a left group action $(\cdot)$

At this point, all we need to define usual vector operations on the manifold is introduced. However, before doing so, some discussion of Equation 2 is warranted. Rewriting it slightly to use the Exponential map and an arbitrary vector $\boldsymbol{\tau} = (\boldsymbol{v}t)^\vee$ we get $\mathcal{X} = \mathcal{X}_0 \circ \text{Exp}\, \boldsymbol{\tau}$. $\mathcal{X}$ equals $\mathcal{X}_0$ perturbed by the tangent vector $\boldsymbol{\tau}$. Since the composition operation is not commutative we known this is generally not equivalent to $\text{Exp}\, \boldsymbol{\tau} \circ \mathcal{X}_0$. The difference is the coordinates $\boldsymbol{\tau}$ is to be understood in. Letting $\mathcal{X}$ act on $\boldsymbol{x}$, we get $\mathcal{X} \cdot \boldsymbol{x} = (\mathcal{X}_0 \circ \text{Exp}\, \boldsymbol{\tau}) \cdot \boldsymbol{x}$. Using the compatibility axiom, this can be written as $\mathcal{X}_0 \cdot (\text{Exp}\, \boldsymbol{\tau} \cdot \boldsymbol{x})$. Here we see that $\boldsymbol{x}$ is acted on firstly by the Exponential of $\boldsymbol{\tau}$, then $\mathcal{X}_0$. For this reason the first action is relative to the second one. That is, $\boldsymbol{\tau}$ is defined in the local coordinates of $\mathcal{X}_0$. This means $\boldsymbol{\tau}^\wedge$ is in the tangent hyper-plane at $\mathcal{X}_0$. Doing the composition the other way around, $\boldsymbol{\tau}^\wedge$ is in the Lie algebra. Figure 4 visualizes this difference between local and global perturbations.

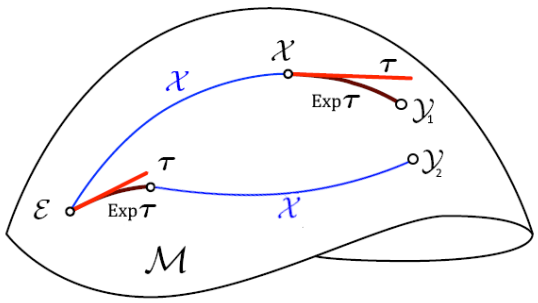


Figure 4: The same tangent vector $\boldsymbol{\tau}$ represents different perturbations when composed with $\mathcal{X}$ from the left or the right. Image adapted from [12].

In robotics, the concept of reference frames is crucial. For example, an airplane will measure the direction of the gravity vector depending on its orientation, but the force from its engines will always act forward. In a global frame, the gravity vector is constant, but the direction of acceleration of the plane changes with the orientation of the plane. This illustrates how some quantities are more conveniently expressed in a global frame, while others are more convenient in the local frame. The special Euclidean group SE(2) expresses all two-dimensional frame transforms. The nature of the group lends itself to visualizations. Figure 5 shows the difference between left and right composition with the same $\boldsymbol{\tau}$ on SE(2).

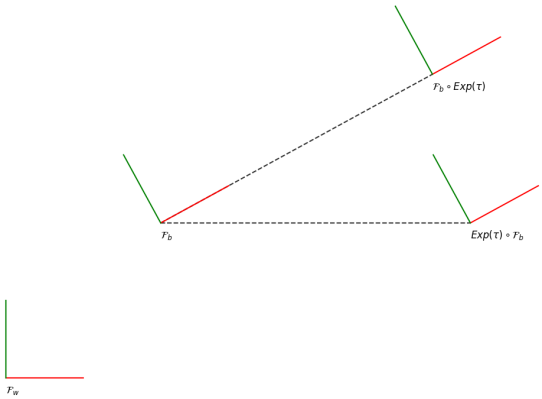


Figure 5: $\boldsymbol{\tau}$ is a perturbation along the x-axis (red). The global frame is in the lower left corner. Clearly, left composing translates along the global x-axis, and right composing along the local x-axis.

Using tangents and the Exponential map, talking about adding increments to an element makes sense. In fact, this is so useful that the $\oplus$ operator is defined just for this.[12] For a vector $\boldsymbol{\tau}^\wedge \in \mathcal{T}_\mathcal{X}\mathcal{M}$ adding the vector to a group element is defined as $\mathcal{X} \oplus \boldsymbol{\tau} := \mathcal{X} \circ \operatorname{Exp} \boldsymbol{\tau}$. The opposite operation $\ominus$ is defined so that it cancels $\oplus$. As with the action, the plus and minus can be defined in left and right variations. Right adds increments expressed in the local tangent, while left is in the Lie algebra. In this work, right will be used exclusively.

$$\mathcal{X} \oplus \boldsymbol{\tau} = \mathcal{X} \circ \operatorname{Exp} \boldsymbol{\tau} \tag{7}$$

$$\mathcal{X} \ominus \mathcal{Y} = \operatorname{Log}(\mathcal{Y}^{-1} \circ \mathcal{X}) \in \mathbb{R}^n \cong \mathcal{T}_\mathcal{X}\mathcal{M} \tag{8}$$

The order of operations follows from the definition of the operators. For example adding and offset to a group element, the subtracting the element can only be evaluated left to right. Trying to evaluate the minus first in Equation 9, means subtracting a group element from a vector, which is not defined.

$$\mathcal{X} \oplus \boldsymbol{\tau} \ominus \mathcal{X} = \operatorname{Log}(\mathcal{X}^{-1} \circ \mathcal{X} \circ \operatorname{Exp} \boldsymbol{\tau}) = \boldsymbol{\tau} \tag{9}$$

Even thought the Exponential map is an exact map from the vector space to the manifold, this does not mean all operations can equivalently be done on the vector space. This is perfectly illustrated by the fact that $\operatorname{Exp}(\boldsymbol{x} + \boldsymbol{y}) \neq \operatorname{Exp} \boldsymbol{x} \circ \operatorname{Exp} \boldsymbol{y}$. The vector space is homeomorphic to the tangent of the manifold, meaning linearization errors are made when calculating on the vector space. The smaller the tangent vectors used, the better the approximation is. In the limit, the calculations are exact. Differentiation on the manifold is defined through vanishingly small perturbations in the tangent space.

The differential of a function $f : \mathbb{R}^m \to \mathbb{R}^n$ is the matrix defined by Equation 10 called the Jacobian. It can be calculated element by element or column by column.

$$\boldsymbol{J}^f_{\boldsymbol{x}} = \frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial f_1(x_1)}{\partial x_1} & \cdots & \frac{\partial f_1(x_m)}{\partial x_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n(x_1)}{\partial x_1} & \cdots & \frac{\partial f_n(x_m)}{\partial x_m} \end{bmatrix} = \begin{bmatrix} \boldsymbol{j}_1, & \ldots, & \boldsymbol{j}_m \end{bmatrix} \tag{10}$$

$$\boldsymbol{j}_i = \frac{\partial f(\boldsymbol{x})}{\partial x_i} = \lim_{h \to 0} \frac{f(\boldsymbol{x} + h\boldsymbol{e}_i) - f(\boldsymbol{x})}{h} \tag{11}$$

The more compact notation alluding the definition of the scalar case is often used.[16]

$$\boldsymbol{J}^f_{\boldsymbol{x}} = \frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}} = \lim_{\boldsymbol{\tau} \to \boldsymbol{0}} \frac{f(\boldsymbol{x} + \boldsymbol{\tau}) - f(\boldsymbol{x})}{\boldsymbol{\tau}} \tag{12}$$

Defining the Jacobian this way is not actually well-defined, as dividing by a vector is not defined. However, in some cases computing the numerator of Equation 12 gives a term linear in $\boldsymbol{\tau}$. This leads to the denominator being cancelled, so the notation is convenient.

Inspired by this, the definition of differentiation of a function $f : \mathcal{M} \to \mathcal{N}$ can be written similarly.[12]

$$\boldsymbol{J}_{\mathcal{X}}^{f} = \frac{Df(\mathcal{X})}{D\mathcal{X}} = \lim_{\boldsymbol{\tau}\to\mathbf{0}} \frac{f(\mathcal{X} \oplus \boldsymbol{\tau}) \ominus f(\mathcal{X})}{\boldsymbol{\tau}} \in \mathbb{R}^{n\times m} \tag{13}$$

Similar to the definition in Equation 12, the derivative measures the rate of change, respecting the nonlinear composition of the group by doing the perturbations in the tangent spaces and mapping onto the manifold. It is important to note that the plus and minus operations and their underlying composition, Exponential, and Logarithmic maps, belong to different manifolds as the f function maps from one manifold to another. Since there exist both left and right addition and subtraction definitions, there exist four variations of the derivative. Again, the right variant is used exclusively in this work, with one exception.

The most important Jacobian is the Jacobian of the Exponential map. It will simply be denoted $\mathcal{J}_r$ and $\mathcal{J}_l$ for the right and left Jacobian respectively. Since the Exponential map maps from the vector tangent to the manifold, the plus of the manifold differentiation definition of Equation 13 degenerates to the normal plus on vector spaces.

$$\begin{aligned}\mathcal{J}_r = \boldsymbol{J}_{\boldsymbol{\tau}}^{\operatorname{Exp}\boldsymbol{\tau}} &= \lim_{\boldsymbol{h}\to\mathbf{0}} \frac{\operatorname{Exp}(\boldsymbol{\tau}+\boldsymbol{h}) \ominus \operatorname{Exp}(\boldsymbol{\tau})}{\boldsymbol{h}} \\ &= \lim_{\boldsymbol{h}\to\mathbf{0}} \frac{\operatorname{Log}(\operatorname{Exp}(\boldsymbol{\tau})^{-1} \circ \operatorname{Exp}(\boldsymbol{\tau}+\boldsymbol{h}))}{\boldsymbol{h}}.\end{aligned} \tag{14}$$

The Jacobian of the Logarithmic map is the inverse of the Jacobian of the Exponential map. The group the Jacobians belong to should be clear from context. There exists closed form solutions to Equation 14 to all groups used in this work.[12] For SO(3) the Jacobians and their inverses are given by

$$\mathcal{J}_r(\theta\boldsymbol{u}) = \boldsymbol{I} - \frac{1-\cos\theta}{\theta}\boldsymbol{u}_\times + \frac{\theta-\sin\theta}{\theta}\boldsymbol{u}_\times^2 \tag{15}$$

$$\mathcal{J}_l(\theta\boldsymbol{u}) = \mathcal{J}_r(-\theta\boldsymbol{u}) = \mathcal{J}_r(\theta\boldsymbol{u})^\top \tag{16}$$

$$\mathcal{J}_r^{-1}(\theta\boldsymbol{u}) = \boldsymbol{I} + \frac{\theta}{2}\boldsymbol{u}_\times + (1 - \theta\frac{1+\cos\theta}{2\sin\theta})\boldsymbol{u}_\times^2 \tag{17}$$

$$\mathcal{J}_l^{-1}(\theta\boldsymbol{u}) = \mathcal{J}_r^{-1}(-\theta\boldsymbol{u}) = \mathcal{J}_r^{-1}(\theta\boldsymbol{u})^\top \tag{18}$$

where $\boldsymbol{u}$ is the unit vector representing the axis of rotation, and θ the angle of rotation.

From Equation 13 an important tool for simplification can be found. For small perturbations $\boldsymbol{\tau}$ the following is accurate

$$f(\mathcal{X} \oplus \boldsymbol{\tau}) \approx f(\mathcal{X}) \oplus \boldsymbol{J}_{\mathcal{X}}^{f}\boldsymbol{\tau}. \tag{19}$$

Using Equation 14 with Equation 19, a useful approximation of the Exponential of a sum is found.

$$\operatorname{Exp}(\boldsymbol{\tau}+\delta\boldsymbol{\tau}) \approx \operatorname{Exp}(\boldsymbol{\tau}) \oplus \mathcal{J}_r(\boldsymbol{\tau})\delta\boldsymbol{\tau} = \operatorname{Exp}(\boldsymbol{\tau}) \circ \operatorname{Exp}(\mathcal{J}_r(\boldsymbol{\tau})\delta\boldsymbol{\tau}) \tag{20}$$

The approximation is better for smaller $\delta\boldsymbol{\tau}$, signified by the δ.

When manipulating equations containing the Exponential map, it will be useful to relate left and right perturbations, $\operatorname{Exp}\boldsymbol{x}^{\mathcal{E}} \circ \mathcal{X} = \mathcal{X} \circ \operatorname{Exp}\boldsymbol{x}^{\mathcal{X}}$. The transform from the local tangent space to the Lie algebra is called the adjoint.

$$\begin{aligned}\operatorname{Exp}{}^{\mathcal{E}}\boldsymbol{x} \circ \mathcal{X} &= \mathcal{X} \circ \operatorname{Exp}{}^{\mathcal{X}}\boldsymbol{x} \\ \exp{}^{\mathcal{E}}\boldsymbol{x}^\wedge \circ \mathcal{X} &= \mathcal{X} \circ \exp{}^{\mathcal{X}}\boldsymbol{x}^\wedge \circ \mathcal{X}^{-1} = \exp(\mathcal{X} \circ {}^{\mathcal{X}}\boldsymbol{x}^\wedge \circ \mathcal{X}^{-1}) \\ {}^{\mathcal{E}}\boldsymbol{x}^\wedge \circ \mathcal{X} &= \mathcal{X} \circ {}^{\mathcal{X}}\boldsymbol{x}^\wedge \circ \mathcal{X}^{-1} = \operatorname{Ad}_{\mathcal{X}}({}^{\mathcal{X}}\boldsymbol{x}^\wedge)\end{aligned}$$

The adjoint is linear;

$$\operatorname{Ad}_{\mathcal{X}}(a\boldsymbol{\tau}^\wedge + b\boldsymbol{\sigma}^\wedge) = \mathcal{X}\circ(a\boldsymbol{\tau}^\wedge + b\boldsymbol{\sigma}^\wedge)\circ\mathcal{X}^{-1} = a\mathcal{X}\circ\boldsymbol{\tau}^\wedge\circ\mathcal{X}^{-1} + b\mathcal{X}\circ\boldsymbol{\sigma}^\wedge\circ\mathcal{X}^{-1} = a\operatorname{Ad}_{\mathcal{X}}(\boldsymbol{\tau}^\wedge) + b\operatorname{Ad}_{\mathcal{X}}(\boldsymbol{\sigma}^\wedge).$$

For any linear function $f : \mathbb{R}^m \to \mathbb{R}^n$ there exists a matrix describing the transform, defined by how the function acts on the basis.

$$f(\boldsymbol{x}) = f(\sum_i x_i\boldsymbol{e}_i) = \sum_i x_i f(\boldsymbol{e}_i) = \sum_i x_i\boldsymbol{b}_i = [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m]\boldsymbol{x} = \boldsymbol{B}\boldsymbol{x}, \quad \boldsymbol{B} \in \mathbb{R}^{n\times m}$$

This means there exists a matrix transformation between the vector tangent spaces describing the adjoint transformation. The matrix $\mathrm{Ad}_{\mathcal{X}}$ can be found by analyzing $(\mathcal{X} \circ \boldsymbol{x}^{\wedge} \circ \mathcal{X}^{-1})^{\vee}$ for something linear in $\boldsymbol{x}$, $(\mathcal{X} \circ \boldsymbol{x}^{\wedge} \circ \mathcal{X}^{-1})^{\vee} = \mathrm{Ad}_{\mathcal{X}}\, \boldsymbol{x}$.

$$\mathrm{Ad}_{\mathcal{X}}(\cdot) : \mathbb{R}^m \to \mathbb{R}^m; {}^{\mathcal{X}}\boldsymbol{x} \to {}^{\mathcal{E}}\boldsymbol{x} = \mathrm{Ad}_{\mathcal{X}}\, {}^{\mathcal{X}}\boldsymbol{x} \tag{21}$$

Applying Equation 21 gives the following property, which will be used later.

$$\mathrm{Exp}\, \boldsymbol{x} \circ \mathcal{X} = \mathcal{X} \circ \mathrm{Exp}(\mathrm{Ad}_{\mathcal{X}}^{-1}\, \boldsymbol{x}) \tag{22}$$

3.3 Inertial navigation

In cases where the dynamics of a body[4] is unknown, convoluted or for other reasons not available, inertial navigation is a viable option. Inertial navigation requires an accelerometer and a gyroscope on the body, collectively known as an IMU sensor. Integrating the measurements of the sensors gives the change of orientation and velocity. Integrating the change of velocity gives change of position. Both the angular velocity and the acceleration are measured in the body frame, as the sensors are on the body. This must be considered when tracking the velocity and position. The equations of motion are

$$\dot{\boldsymbol{R}} = \boldsymbol{R}\boldsymbol{\omega}_{\times}, \tag{23a}$$

$$\dot{\boldsymbol{v}} = \boldsymbol{R}\boldsymbol{a} + \boldsymbol{g}, \tag{23b}$$

$$\dot{\boldsymbol{p}} = \boldsymbol{v}. \tag{23c}$$

The addition of the gravity vector $\boldsymbol{g}$ is necessary as an accelerometer measures specific force, this means gravity is not measured. An accelerometer at rest will measure a specific acceleration of $-\boldsymbol{g}$ in the global frame. Equation 23a is exactly the equation found earlier when analyzing the tangents of SO(n). In Equation 23b the acceleration is rotated to align with the global frame and compensated for gravity. Since the velocity is expressed in the global frame, it is simply position differentiated.

For discrete calculations Equation 23 is not practical. Solving Equation 23 between times t_k and t_{k+1} yields

$$\boldsymbol{R}(t_{k+1}) = \boldsymbol{R}(t_k)\, \mathrm{Exp} \int_{t_k}^{t_{k+1}} \boldsymbol{\omega}(t)dt, \tag{24a}$$

$$\boldsymbol{v}(t_{k+1}) = \boldsymbol{v}(t_k) + \int_{t_k}^{t_{k+1}} \boldsymbol{R}(t)\boldsymbol{a}(t)dt + \boldsymbol{g}\delta t, \tag{24b}$$

$$\boldsymbol{p}(t_{k+1}) = \boldsymbol{p}(t_k) + \boldsymbol{v}(t_k)\delta t + \int_{t_k}^{t_{k+1}} \int_{t_k}^{t} \boldsymbol{R}(s)\boldsymbol{a}(s)dsdt + \frac{1}{2}\boldsymbol{g}\delta t^2. \tag{24c}$$

δt is the time between time step k and $k+1$, $\delta t_k := t_{k+1} - t_k$. This is easily verified by differentiating with respect to t_{k+1}, keeping in mind that δt_k is dependent on t_{k+1}. The Exponential map of SO(n) is inserted removing the need for the cross matrix map. $\boldsymbol{R}(t_k), \boldsymbol{v}(t_k)$ and $\boldsymbol{p}(t_k)$ are the initial conditions of the propagation step. It is convenient to define the increments caused by the movement of the body separately. Let $\Delta\boldsymbol{R}_{t_k}^{t_{k+1}}$ be the rotational increment between time t_k and t_{k+1}. Similar notation is used for the velocity and position increments.

$$\Delta\boldsymbol{R}_{t_k}^{t_{k+1}} = \mathrm{Exp} \int_{t_k}^{t_{k+1}} \boldsymbol{\omega}(t)dt \tag{25a}$$

$$\Delta\boldsymbol{v}_{t_k}^{t_{k+1}} = \int_{t_k}^{t_{k+1}} \Delta\boldsymbol{R}_{t_k}^{t}\boldsymbol{a}(t)dt \tag{25b}$$

$$\Delta\boldsymbol{p}_{t_k}^{t_{k+1}} = \int_{t_k}^{t_{k+1}} \Delta\boldsymbol{v}_{t_k}^{t}dt \tag{25c}$$

[4] With body any object moving in two or three dimensions is meant.

For convenience, the shorthands $\boldsymbol{R}_k = \boldsymbol{R}(t_k)$, $\Delta\boldsymbol{R}_k = \Delta\boldsymbol{R}_{t_k}^{t_{k+1}}$ and similarly for $\boldsymbol{v}$ and $\boldsymbol{p}$ are defined. Using Equation 24 and Equation 25, the discrete propagation equations can be formulated compactly.

$$\boldsymbol{R}_{k+1} = \boldsymbol{R}_k \Delta\boldsymbol{R}_k \tag{26a}$$
$$\boldsymbol{v}_{k+1} = \boldsymbol{v}_k + \boldsymbol{R}_k \Delta\boldsymbol{v}_k + \boldsymbol{g}\delta t_k \tag{26b}$$
$$\boldsymbol{p}_{k+1} = \boldsymbol{p}_k + \boldsymbol{v}_k \delta t_k + \boldsymbol{R}_k \Delta\boldsymbol{p}_k + \frac{1}{2}\boldsymbol{g}\delta t_k^2 \tag{26c}$$

So far the IMU motion equations are discretized without approximations. In reality approximations must be made when calculating the integrals of Equation 25. When calculating the increments, the usual assumption of piece-wise constant measurements will be used. This assumes the acceleration and angular velocity is constant between sampling points, $\boldsymbol{w}(t) = \boldsymbol{w}(t_k) = \boldsymbol{w}_k$ for all $t \in [t_k, t_{k+1})$, and similarly for $\boldsymbol{a}$. Assuming piece-wise constant measurements, the increments for a single IMU measurement can be calculated as

$$\Delta\boldsymbol{R}_k = \mathrm{Exp}\int_{t_k}^{t_{k+1}} \boldsymbol{\omega}_k dt = \mathrm{Exp}(\boldsymbol{\omega}_k \delta t), \tag{27a}$$

$$\begin{aligned}
\Delta\boldsymbol{v}_k &= \int_{t_k}^{t_{k+1}} \mathrm{Exp}(\boldsymbol{\omega}_k(t-t_k))dt\boldsymbol{a}_k = \int_0^{\delta t} \mathrm{Exp}(\boldsymbol{\omega}_k\tau)d\tau\boldsymbol{a}_k \\
&= \int_0^{\delta t} \boldsymbol{I} + \sin(||\boldsymbol{\omega}_k||\tau)\boldsymbol{u}_\times + (1-\cos(||\boldsymbol{\omega}_k||\tau))\boldsymbol{u}_\times^2 d\tau\boldsymbol{a}_k \\
&= (\boldsymbol{I}\delta t + \frac{1-\cos(||\boldsymbol{\omega}_k||\delta t)}{||\boldsymbol{\omega}_k||}\boldsymbol{u}_\times + (\delta t - \frac{\sin(||\boldsymbol{\omega}_k||\delta t)}{||\boldsymbol{\omega}_k||})\boldsymbol{u}_\times^2)\boldsymbol{a}_k \\
&= (\boldsymbol{I} + \frac{1-\cos(||\boldsymbol{\omega}_k||\delta t)}{||\boldsymbol{\omega}_k||\delta t}\boldsymbol{u}_\times + \frac{||\boldsymbol{\omega}_k||\delta t - \sin(||\boldsymbol{\omega}_k||\delta t)}{||\boldsymbol{\omega}_k||\delta t}\boldsymbol{u}_\times^2)\boldsymbol{a}_k\delta t \\
&= \mathcal{J}_l(\boldsymbol{\omega}_k\delta t)\boldsymbol{a}_k\delta t,
\end{aligned} \tag{27b}$$

$$\begin{aligned}
\Delta\boldsymbol{p}_k &= \int_{t_k}^{t_{k+1}} \Delta\boldsymbol{v}_{t_k}^t dt = \int_0^{\delta t} \mathcal{J}_l(\boldsymbol{\omega}_k\tau)\tau d\tau\boldsymbol{a}_k \\
&= (\boldsymbol{I} + 2\frac{||\boldsymbol{\omega}_k||\delta t - \sin(||\boldsymbol{\omega}_k||\delta t)}{(||\boldsymbol{\omega}_k||\delta t)^2}\boldsymbol{u}_\times + \frac{(||\boldsymbol{\omega}_k||\delta t)^2 + 2(\cos(||\boldsymbol{\omega}_k||\delta t) - 1)}{(||\boldsymbol{\omega}_k||\delta t)^2}\boldsymbol{u}_\times^2)\frac{1}{2}\boldsymbol{a}_k\delta t^2 \\
&= \mathcal{K}_l(\boldsymbol{\omega}_k\delta t)\frac{1}{2}\boldsymbol{a}_k\delta t^2.
\end{aligned} \tag{27c}$$

The coordinate transform $\tau = t - t_k$ has been used repeatedly, together with $t_{k-1} - t_k = \delta t$. The entry of the left Jacobian of SO(3) might seem a bit random, but some intuition will be offered in the next section. The name $\mathcal{K}$ signifies nothing, and is simply the letter coming after $\mathcal{J}$ in the alphabet. The expression is not known to the author under any name.

In truth, Equation 27 is an approximation as the measurements are not piece-wise constant. This is a normal assumption as measurements are received discreetly in a digital system. More complicated techniques can be conceived, e.g. interpolating measurements. However, using non-constant measurements comes with some pitfalls, as the reference frames of consecutive measurements are not the same. The frames rotate with the body, meaning frame transformations must be used in calculations of consecutive measurements. The high sample rate of the typical IMU gives satisfactory results with the piece-wise constant assumptions.[14]

Assuming instead the measurements to be piece-wise constant *globally*, the equations simplify drastically. With global constant acceleration the term $\Delta\boldsymbol{R}_{t_k}^t\boldsymbol{a}(t)$ is constant for $t \in [t_k, t_{k+1})$ meaning $\Delta\boldsymbol{R}_{t_k}^t\boldsymbol{a}(t) = \boldsymbol{a}(t_k) = \boldsymbol{a}_k$, since $\Delta\boldsymbol{R}_{t_k}^{t_k} = \boldsymbol{I}$. Using this, calculating the increments is straight forward.

$$\Delta\boldsymbol{R}_k = \mathrm{Exp}(\boldsymbol{\omega}_k\delta t) \tag{28a}$$
$$\Delta\boldsymbol{v}_k = \boldsymbol{a}_k\delta t \tag{28b}$$
$$\Delta\boldsymbol{p}_k = \frac{1}{2}\boldsymbol{a}_k\delta t^2 \tag{28c}$$

The difference in the assumptions of Equation 27 and Equation 28 is whether the acceleration is assumed to rotate with the body during the discretized time step. In the first approach, this is the case. This is reasonable as the source of the acceleration is usually fixed on the body, like motors, thrusters or rotors. In the limit $||\boldsymbol{\omega}||\delta t \to 0$ Equation 27 approaches Equation 28. This makes sense, as this means there either is no rotation, making the constant global and constant local acceleration assumptions identical, or that the sampling period is so short that the body does not rotate enough for the approaches to make a difference.

Inserting Equation 28 into Equation 26 gives the forward Euler scheme for solving Equation 23. IMU measurements are expected to arrive often, so a method like forward Euler is probably sufficient in the face of the typical noise levels expected from budget IMUs.[14] The increments can be pre-integrated so that the full pose update can be run at lower-than-IMU frequency. This is simply done by collecting multiple measurements between time steps, and calculating the increments by combining the measurments.

$$\Delta \boldsymbol{R}_{t_k}^{t_{k+2}} = \Delta \boldsymbol{R}_{t_k}^{t_{k+1}} \Delta \boldsymbol{R}_{t_{k+1}}^{t_{k+2}} = \operatorname{Exp}(\boldsymbol{\omega}_k \delta t_1) \operatorname{Exp}(\boldsymbol{\omega}_{k+1} \delta t_2)$$

where $\delta t_i = t_{k+i} - t_{k+i-1}$. It might be tempting to split the integrals of Equation 25 instead, however, this is incorrect because the following measurements are defined in the tangent spaces of the state at the next time steps, which, generally, is not the same as the current tangent space.

$$\begin{aligned} \operatorname{Exp} \int_{t_k}^{t_{k+2}} \boldsymbol{\omega}(t) dt &= \operatorname{Exp}(\int_{t_k}^{t_{k+1}} \boldsymbol{\omega}_k dt + \int_{t_{k+1}}^{t_{k+2}} \boldsymbol{\omega}_{k+1} dt) \\ &= \operatorname{Exp}(\boldsymbol{\omega}_k \delta t_1 + \boldsymbol{\omega}_{k+1} \delta t_2) \\ &\neq \operatorname{Exp}(\boldsymbol{\omega}_k \delta t_1) \operatorname{Exp}(\boldsymbol{\omega}_{k+1} \delta t_2) \end{aligned}$$

3.4 $\mathrm{SE}_2(3)$

The Lie group $\mathrm{SE}_2(3)$ extends the more usual special euclidean group in three dimensions, SE(3), which encapsulates all poses.[9] That is, orientation and position. The extension in $\mathrm{SE}_2(3)$ is the addition of a velocity state. This makes the group very useful for inertial navigation in robotics, where orientation, velocity and position are the states to be estimated. The composition operation of the group is $\{\boldsymbol{R}_1, \boldsymbol{v}_1, \boldsymbol{p}_1\} \circ \{\boldsymbol{R}_2, \boldsymbol{v}_2, \boldsymbol{p}_2\} = \{\boldsymbol{R}_1\boldsymbol{R}_2, \boldsymbol{v}_1 + \boldsymbol{R}_1\boldsymbol{v_2}, \boldsymbol{p}_1 + \boldsymbol{R}_1\boldsymbol{p_2}\}$. The rotation matrices are composed with $\boldsymbol{R}_2$ being local to $\boldsymbol{R}_1$, and the vectors transformed into the frame of the first pose, then added. The group can be identified with a 5×5 matrix, making the composition operation matrix multiplication. This is useful because manipulating matrices is easy.

$$SE_2(3) := \left\{ \boldsymbol{T} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{v} & \boldsymbol{p} \\ \boldsymbol{0}_{2\times 3} & & \boldsymbol{I}_2 \end{bmatrix} \in \mathbb{R}^{5\times 5} \middle| \begin{matrix} \boldsymbol{R} \in SO(3) \\ \boldsymbol{v}, \boldsymbol{p} \in \mathbb{R}^3 \end{matrix} \right\} \tag{29}$$

It is easily verified that this makes the composition operation matrix multiplication. The $\circ$ is omitted for brevity when it refers to matrix multiplication.

$$\boldsymbol{T}_1\boldsymbol{T}_2 = \begin{bmatrix} \boldsymbol{R}_1\boldsymbol{R}_2 & \boldsymbol{v}_1 + \boldsymbol{R}_1\boldsymbol{v}_2 & \boldsymbol{p}_1 + \boldsymbol{R}_1\boldsymbol{p}_2 \\ \boldsymbol{0}_{2\times 3} & & \boldsymbol{I}_2 \end{bmatrix} \tag{30}$$

The inverse element is given by

$$\boldsymbol{T}^{-1} = \begin{bmatrix} \boldsymbol{R}^\top & -\boldsymbol{R}^\top\boldsymbol{v} & -\boldsymbol{R}^\top\boldsymbol{p} \\ \boldsymbol{0}_{2\times 3} & & \boldsymbol{I}_2 \end{bmatrix}. \tag{31}$$

A common choice for the group action is to use the same action as SE(3). This is a frame transformation from local to global coordinates of a vector $\boldsymbol{x}$ defined in the body frame of $\boldsymbol{T}$, $\boldsymbol{x}^b$.[12] The corresponding world parametrization is denoted $\boldsymbol{x}^w$.

$$\boldsymbol{x}^w = \boldsymbol{T} \cdot \boldsymbol{x}^b := \boldsymbol{R}\boldsymbol{x}^b + \boldsymbol{p} \tag{32}$$

In the target tracking part of this work another group action will prove useful. This action is similar to the one defined above, but transforms a six-dimensional vector consisting of three dimensional

position and velocity state. Let $\boldsymbol{x} = [\boldsymbol{t}, \boldsymbol{u}]^\top$, where $\boldsymbol{t}$ and $\boldsymbol{u}$ are three dimensional position and velocity states respectively. The action is then

$$\boldsymbol{x}^w = \boldsymbol{T} : \boldsymbol{x}^b := \begin{bmatrix} \boldsymbol{R} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{R} \end{bmatrix} \boldsymbol{x}^b + \begin{bmatrix} \boldsymbol{p} \\ \boldsymbol{v} \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}\boldsymbol{t}^b + \boldsymbol{p} \\ \boldsymbol{R}\boldsymbol{u}^b + \boldsymbol{v} \end{bmatrix} \tag{33}$$

This resembles the frame transformation action, and is identical for the positional state, but for the velocity state there is a slight difference. The velocity of the group is added, not the position.

The Lie algebra is the set of 5×5 matrices generating the differential rotations, velocities and translations. There are nine generators of the algebra. The three first correspond to the generators of $\mathfrak{so}(3)$, see Equation 4, the last six are trivial vector generators, 3 for velocity and 3 for position.

$$G_1 = \begin{bmatrix} 0&0&0&0&0 \\ 0&0&-1&0&0 \\ 0&1&0&0&0 \\ 0&0&0&0&0 \\ 0&0&0&0&0 \end{bmatrix} \quad G_2 = \begin{bmatrix} 0&0&1&0&0 \\ 0&0&0&0&0 \\ -1&0&0&0&0 \\ 0&0&0&0&0 \\ 0&0&0&0&0 \end{bmatrix} \quad G_3 = \begin{bmatrix} 0&-1&0&0&0 \\ 1&0&0&0&0 \\ 0&0&0&0&0 \\ 0&0&0&0&0 \\ 0&0&0&0&0 \end{bmatrix}$$

$$G_4 = \begin{bmatrix} 0&0&0&1&0 \\ 0&0&0&0&0 \\ 0&0&0&0&0 \\ 0&0&0&0&0 \\ 0&0&0&0&0 \end{bmatrix} \quad G_5 = \begin{bmatrix} 0&0&0&0&0 \\ 0&0&0&1&0 \\ 0&0&0&0&0 \\ 0&0&0&0&0 \\ 0&0&0&0&0 \end{bmatrix} \quad G_6 = \begin{bmatrix} 0&0&0&0&0 \\ 0&0&0&0&0 \\ 0&0&0&1&0 \\ 0&0&0&0&0 \\ 0&0&0&0&0 \end{bmatrix}$$

$$G_7 = \begin{bmatrix} 0&0&0&0&1 \\ 0&0&0&0&0 \\ 0&0&0&0&0 \\ 0&0&0&0&0 \\ 0&0&0&0&0 \end{bmatrix} \quad G_8 = \begin{bmatrix} 0&0&0&0&0 \\ 0&0&0&0&1 \\ 0&0&0&0&0 \\ 0&0&0&0&0 \\ 0&0&0&0&0 \end{bmatrix} \quad G_9 = \begin{bmatrix} 0&0&0&0&0 \\ 0&0&0&0&0 \\ 0&0&0&0&1 \\ 0&0&0&0&0 \\ 0&0&0&0&0 \end{bmatrix}$$

With this order of the generators, the Lie algebra $\mathfrak{se}_2(3)$ becomes

$$\boldsymbol{\tau}^\wedge = \begin{bmatrix} \boldsymbol{\phi} \\ \boldsymbol{\nu} \\ \boldsymbol{\rho} \end{bmatrix}^\wedge = \begin{bmatrix} \boldsymbol{\phi}_\times & \boldsymbol{\nu} & \boldsymbol{\rho} \\ \boldsymbol{0}_{2\times 3} & \boldsymbol{0}_{2\times 2} \end{bmatrix} \in \mathfrak{se}_2(3). \tag{34}$$

The Exponential map is found by using Equation 3. Notice that $\begin{bmatrix} \boldsymbol{\phi}_\times & \boldsymbol{\nu} & \boldsymbol{\rho} \\ \boldsymbol{0}_{2\times 3} & \boldsymbol{0}_{2\times 1} & \boldsymbol{0}_{2\times 1} \end{bmatrix}^n = \begin{bmatrix} \boldsymbol{\phi}_\times^n & \boldsymbol{\phi}_\times^{n-1}\boldsymbol{\nu} & \boldsymbol{\phi}_\times^{n-1}\boldsymbol{\rho} \\ \boldsymbol{0}_{2\times 3} & \boldsymbol{0}_{2\times 1} & \boldsymbol{0}_{2\times 1} \end{bmatrix}$ for $n > 0$.

$$\begin{aligned} \operatorname{Exp} \boldsymbol{\tau} &= \boldsymbol{I} + \sum_{k=1}^{\infty} \frac{1}{k!} \begin{bmatrix} \boldsymbol{\phi}_\times & \boldsymbol{\nu} & \boldsymbol{\rho} \\ \boldsymbol{0}_{2\times 3} & \boldsymbol{0}_{2\times 1} & \boldsymbol{0}_{2\times 1} \end{bmatrix}^k \\ &= \begin{bmatrix} \sum_{k=0} \frac{\boldsymbol{\phi}_\times^k}{k!} & \sum_{k=0} \frac{\boldsymbol{\phi}_\times^k \boldsymbol{\nu}}{(k+1)!} & \sum_{k=0} \frac{\boldsymbol{\phi}_\times^k \boldsymbol{\rho}}{(k+1)!} \\ \boldsymbol{0}_{2\times 3} & & \boldsymbol{I}_2 \end{bmatrix} \\ &= \begin{bmatrix} \operatorname{Exp} \boldsymbol{\phi} & V(\boldsymbol{\phi}_\times)\boldsymbol{\nu} & V(\boldsymbol{\phi}_\times)\boldsymbol{\rho} \\ \boldsymbol{0}_{2\times 3} & & \boldsymbol{I}_2 \end{bmatrix} \end{aligned}$$

The usual trick is used for solving V. Split $\boldsymbol{\phi}$ into a length θ and a unit vector $\boldsymbol{u}$, then use that $\boldsymbol{u}_\times^3 = -\boldsymbol{u}_\times$ making the series repeat the first two powers of $\boldsymbol{u}_\times$. Following [19];

$$\begin{aligned} V(\boldsymbol{\phi}_\times) &= \sum_{k=0} \frac{\boldsymbol{\phi}_\times^k}{(k+1)!} = \boldsymbol{I} + \frac{1}{2!}\boldsymbol{\phi}_\times + \frac{1}{3!}\boldsymbol{\phi}_\times^2 + \dots \\ &= \boldsymbol{I} + \sum_{k=1} \frac{(-1)^{k-1}\theta^{2k-1}}{(2k)!}\boldsymbol{u}_\times + \sum_{k=1} \frac{(-1)^{k-1}\theta^{2k}}{(2k+1)!}\boldsymbol{u}_\times^2 \\ &= \boldsymbol{I} + \frac{1-\cos\theta}{\theta}\boldsymbol{u}_\times + \frac{\theta - \sin\theta}{\theta}\boldsymbol{u}_\times^2 = \mathcal{J}_l(\boldsymbol{\phi}). \end{aligned}$$

The left Jacobian of SO(3) enters again! The full Exponential map of $\mathrm{SE}_2(3)$ is

$$\operatorname{Exp} \boldsymbol{\tau} = \begin{bmatrix} \operatorname{Exp} \boldsymbol{\phi} & \mathcal{J}_l(\boldsymbol{\phi})\boldsymbol{\nu} & \mathcal{J}_l(\boldsymbol{\phi})\boldsymbol{\rho} \\ \boldsymbol{0}_{2\times 3} & & \boldsymbol{I}_2 \end{bmatrix}, \tag{35}$$

where the Exponential of SO(3) is used. The effect of the Jacobian here is the same as in Equation 27, the velocity and position tangents are rotated with the body. The Jacobian wrap the tangent vectors onto the manifold. Recall that the tangent space is also known as the velocity space. Using this intuition, viewing the angular velocity and the acceleration measurements as members of the tangent space of $\mathrm{SE}_2(3)$, both the rotational and velocity increments are equivalently defined as in Equation 27. Since there is no velocity measurement, the dynamics of $\Delta\boldsymbol{p}$ are

not exactly modeled by the Exponential map of $SE_2(3)$. In fact, $SE_2(3)$ treat velocity and position independently of each other, so the dependence in Equation 27c is not captured.

The Logarithmic map follows from the definition of the Exponential map, since the Logarithmic map is its inverse.

$$\operatorname{Log} \boldsymbol{T} = \begin{bmatrix} \operatorname{Log} \boldsymbol{R} \\ \mathcal{J}_l^{-1}(\boldsymbol{\phi})\boldsymbol{v} \\ \mathcal{J}_l^{-1}(\boldsymbol{\phi})\boldsymbol{p} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\phi} \\ \boldsymbol{\nu} \\ \boldsymbol{\rho} \end{bmatrix} \tag{36}$$

The Jacobians of $SE_2(3)$ are non-trivial. However, since velocity and position are treated independently, it is found by simply extending the Jacobians for SE(3), found in e.g.[20].

$$\mathcal{J}_l(\boldsymbol{\tau}) = \begin{bmatrix} \mathcal{J}_l(\boldsymbol{\phi}) & \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} \\ Q(\boldsymbol{\nu},\boldsymbol{\phi}) & \mathcal{J}_l(\boldsymbol{\phi}) & \boldsymbol{0}_{3\times3} \\ Q(\boldsymbol{\rho},\boldsymbol{\phi}) & \boldsymbol{0}_{3\times3} & \mathcal{J}_l(\boldsymbol{\phi}) \end{bmatrix} \tag{37}$$

$$\begin{aligned} Q(\boldsymbol{\alpha},\boldsymbol{\phi}) = {} & \frac{1}{2}\boldsymbol{\alpha}_\times + \frac{\phi - \sin\phi}{\phi^3}(\boldsymbol{\phi}_\times\boldsymbol{\alpha}_\times + \boldsymbol{\alpha}_\times\boldsymbol{\phi}_\times + \boldsymbol{\phi}_\times\boldsymbol{\alpha}_\times\boldsymbol{\phi}_\times) \\ & - \frac{1 - \phi^2/2 - \cos\phi}{\phi^4}(\boldsymbol{\phi}_\times^2\boldsymbol{\alpha}_\times + \boldsymbol{\alpha}_\times\boldsymbol{\phi}_\times^2 - 3\boldsymbol{\phi}_\times\boldsymbol{\alpha}_\times\boldsymbol{\phi}_\times) \\ & - \frac{1}{2}(\frac{1 - \phi^2/2 - \cos\phi}{\phi^4} - 3\frac{\phi - \sin\phi - \phi^3/6}{\phi^5})(\boldsymbol{\phi}_\times\boldsymbol{\alpha}_\times\boldsymbol{\phi}_\times^2 + \boldsymbol{\phi}_\times^2\boldsymbol{\alpha}_\times\boldsymbol{\phi}_\times) \end{aligned} \tag{38}$$

where $\boldsymbol{\alpha}$ is one of $\boldsymbol{\rho}$ and $\boldsymbol{\nu}$, and $\phi = ||\boldsymbol{\phi}||$. The right Jacobian can be found as $\mathcal{J}_r(\boldsymbol{\tau}) = \mathcal{J}_l(-\boldsymbol{\tau})$.

The adjoint matrix at $\boldsymbol{T}$ can also be found by extending the adjoint of SE (3), found, e.g., in [12], [19].

$$\operatorname{Ad}_{\boldsymbol{T}} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} \\ \boldsymbol{v}_\times\boldsymbol{R} & \boldsymbol{R} & \boldsymbol{0}_{3\times3} \\ \boldsymbol{p}_\times\boldsymbol{R} & \boldsymbol{0}_{3\times3} & \boldsymbol{R} \end{bmatrix} \tag{39}$$

3.5 Modeling on $SE_2(3)$

The equations of Equation 26 can be made to fit the $SE_2(3)$ scheme. Following [9], lets define the function $f_{\delta t}$ which captures the change of position between time steps solely caused by the velocity at the beginning of the time step.

$$f_{\delta t}(\boldsymbol{T}) = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{v} & \boldsymbol{p} + \boldsymbol{v}\delta t \\ \boldsymbol{0}_{2\times3} & & \boldsymbol{I}_2 \end{bmatrix} \tag{40}$$

This function has a few useful properties. It is easily verified that $f_{\delta t}(\boldsymbol{AB}) = f_{\delta t}(\boldsymbol{A})f_{\delta t}(\boldsymbol{B})$, and $f_{\delta t}(\boldsymbol{A})^{-1} = f_{\delta t}(\boldsymbol{A}^{-1})$ for any $\boldsymbol{A}, \boldsymbol{B} \in SE_2(3)$. This leads to

$$\begin{aligned} f_{\delta t}(\boldsymbol{T} \oplus \boldsymbol{\tau}) &= f_{\delta t}(\boldsymbol{T}) f_{\delta t}(\begin{bmatrix} \operatorname{Exp}\boldsymbol{\phi} & \mathcal{J}_l(\boldsymbol{\phi})\boldsymbol{\nu} & \mathcal{J}_l(\boldsymbol{\phi})\boldsymbol{\rho} \\ \boldsymbol{0}_{2\times3} & & \boldsymbol{I}_2 \end{bmatrix}) \\ &= f_{\delta t}(\boldsymbol{T}) \begin{bmatrix} \operatorname{Exp}\boldsymbol{\phi} & \mathcal{J}_l(\boldsymbol{\phi})\boldsymbol{\nu} & \mathcal{J}_l(\boldsymbol{\phi})(\boldsymbol{\rho} + \boldsymbol{\nu}\delta t) \\ \boldsymbol{0}_{2\times3} & & \boldsymbol{I}_2 \end{bmatrix} \\ &= f_{\delta t}(\boldsymbol{T}) \operatorname{Exp}(\boldsymbol{F}_{\delta t}\boldsymbol{\tau}), \end{aligned} \tag{41}$$

where $\boldsymbol{F}_{\delta t} = \begin{bmatrix} \boldsymbol{I}_3 & \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{I}_3 & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & \delta t\boldsymbol{I}_3 & \boldsymbol{I}_3 \end{bmatrix}$.

Next, notice that the increments are defined locally as they are pre-multiplied by the rotation matrix. Gravity is defined globally, as this is much simpler with globally constant vectors. Remember that local increments can be included with post-multiplication, and global with pre-multiplication. This gives

$$\boldsymbol{T}_{k+1} = \boldsymbol{G}_{\delta t} f_{\delta t}(\boldsymbol{T}_k)\boldsymbol{\Delta}_k, \tag{42}$$

where $\boldsymbol{G}_{\delta t}$ are the gravity effects, which are only dependent on the duration of the time step.

$$\boldsymbol{G}_{\delta t} = \begin{bmatrix} \boldsymbol{I}_3 & \boldsymbol{g}\delta t & \boldsymbol{g}\delta t^2/2 \\ \boldsymbol{0}_{2\times 3} & & \boldsymbol{I}_2 \end{bmatrix} \tag{43}$$

$\boldsymbol{\Delta}_k$ is the increment matrix calculated from the IMU measurements at time step k using, for example, Equation 27 or Equation 28.

$$\boldsymbol{\Delta}_k = \begin{bmatrix} \Delta\boldsymbol{R}_k & \Delta\boldsymbol{v}_k & \Delta\boldsymbol{p}_k \\ \boldsymbol{0}_{2\times 3} & & \boldsymbol{I}_2 \end{bmatrix} \tag{44}$$

Multiplying out Equation 42 yields Equation 26 exactly. This is the process model of inertial navigation on the $\mathrm{SE}_2(3)$ group.

Despite the lack of proper modeling of the dependence between velocity and position, using the tangent of $\mathrm{SE}_2(3)$ for measurements is practical. This calls for some exploring in how to define the positional tangent in the tangent space. Is there an expression yielding Equation 27c? We want to factor out the left Jacobian to the left, i.e. $\mathcal{K}_l = \mathcal{J}_l\boldsymbol{C}$ where $\boldsymbol{C}$ is some compensation. This gives $\boldsymbol{C} = \mathcal{J}_l^{-1}\mathcal{K}_l$. Now all that stands in our way is some careful algebra. Remember $\boldsymbol{u}_\times^3 = -\boldsymbol{u}_\times$, since $\boldsymbol{u}$ is a unit vector.

$$\begin{aligned} \mathcal{J}_l^{-1}\mathcal{K}_l &= (\boldsymbol{I} - \frac{\theta}{2}\boldsymbol{u}_\times + (1 - \frac{\theta + \theta\cos\theta}{2\sin\theta})\boldsymbol{u}_\times^2)(\boldsymbol{I} + 2\frac{\theta - \sin\theta}{\theta^2}\boldsymbol{u}_\times + \frac{\theta^2 + 2(\cos\theta - 1)}{\theta^2}\boldsymbol{u}_\times^2) \\ &= \boldsymbol{I} + (\frac{\cos\theta - 1}{\theta} + \frac{\cos\theta + 1}{\sin\theta} - \frac{\cos\theta + 1}{\theta})\boldsymbol{u}_\times + (\frac{\sin\theta}{\theta} + \frac{\cos^2\theta - 1}{\theta\sin\theta})\boldsymbol{u}_\times^2 \\ &= \boldsymbol{I} + (\frac{\cos\theta + 1}{\sin\theta} - \frac{2}{\theta})\boldsymbol{u}_\times = \boldsymbol{C}(\theta\boldsymbol{u}) \end{aligned} \tag{45}$$

Interestingly, all second order terms cancel. Using Equation 45, Equation 27 can be expressed exactly using the Exponential map.

$$\begin{aligned} & f_{\delta t}(\boldsymbol{T}_k) \oplus [\boldsymbol{\omega}_k\delta t, \boldsymbol{a}_k\delta t, \boldsymbol{C}(\boldsymbol{\omega}_k\delta t)\frac{1}{2}\boldsymbol{a}_k\delta t^2]^\top \\ &\quad = f_{\delta t}(\boldsymbol{T}_k)\underbrace{\mathrm{Exp}[\boldsymbol{\omega}_k\delta t, \boldsymbol{a}_k\delta t, \boldsymbol{C}(\boldsymbol{\omega}_k\delta t)\frac{1}{2}\boldsymbol{a}_k\delta t^2]^\top}_{=\boldsymbol{\Delta}_k} \\ \Rightarrow \boldsymbol{\Delta}_k &= \begin{bmatrix} \mathrm{Exp}(\boldsymbol{\omega}_k\delta t) & \mathcal{J}_l(\boldsymbol{\omega}_k\delta t)\boldsymbol{a}_k\delta t & \mathcal{J}_l(\boldsymbol{\omega}_k\delta t)\boldsymbol{C}(\boldsymbol{\omega}_k\delta t)\frac{1}{2}\boldsymbol{a}_k\delta t^2 \\ \boldsymbol{0}_{2\times 3} & & \boldsymbol{I}_2 \end{bmatrix} \\ &= \begin{bmatrix} \mathrm{Exp}(\boldsymbol{\omega}_k\delta t) & \mathcal{J}_l(\boldsymbol{\omega}_k\delta t)\boldsymbol{a}_k\delta t & \mathcal{K}_l(\boldsymbol{\omega}_k\delta t)\frac{1}{2}\boldsymbol{a}_k\delta t^2 \\ \boldsymbol{0}_{2\times 3} & & \boldsymbol{I}_2 \end{bmatrix} \end{aligned} \tag{46}$$

Comparing Equation 46 to Equation 27, the components match exactly! Equation 45 shows that defining the position tangent as the integral of the velocity tangent $\boldsymbol{a}_k\delta t$, i.e. $\frac{1}{2}\boldsymbol{a}_k\delta t^2$, is an acceptable approximation for small δt. Actually, for smaller δt the positional increments caused by the acceleration can be neglected entirely since $\delta t^2 \to 0$ quickly, if the motion dynamics of the body are not very quick. This means $f_{\delta t}$ has the most important influence on change of position.

It can be tempting to formulate the entire process model as an increment in the tangent space of the current pose. Defining the input vector

$$\boldsymbol{\xi}_k := \begin{bmatrix} \boldsymbol{\omega}_k\delta_t \\ (\boldsymbol{a}_k + \boldsymbol{R}_k^\top\boldsymbol{g})\delta t \\ \boldsymbol{R}_k^\top\boldsymbol{v}_k\delta t + \frac{1}{2}(\boldsymbol{a}_k + \boldsymbol{R}_k^\top\boldsymbol{g})\delta t^2 \end{bmatrix} \tag{47}$$

with the process model $\boldsymbol{T}_{k+1} = \boldsymbol{T}_k \oplus \boldsymbol{\xi}_k$ gives such a model. However, this is not entirely equivalent to Equation 26. Here, the local gravity vector is rotated with the body during a time step. This is not right as gravity is a constant vector globally. When gravity is rotated, it erroneously accelerates the body in other directions than down. Figure 6 shows how this affects the modeling of a body in free fall. The same happens with the initial velocity, which arguably can be OK for some applications, e.g. a car, since cars usually keep their velocity oriented with their heading. This should be captured by the accelerometer, but in between time steps there can be angular velocities and accelerations which are not captured. Due to the high rate of IMU measurements, this is unlikely to cause an issue. In addition, in this work there are no assumptions placed on the motion of the body, so a general approach with orientation and velocity detached is taken.

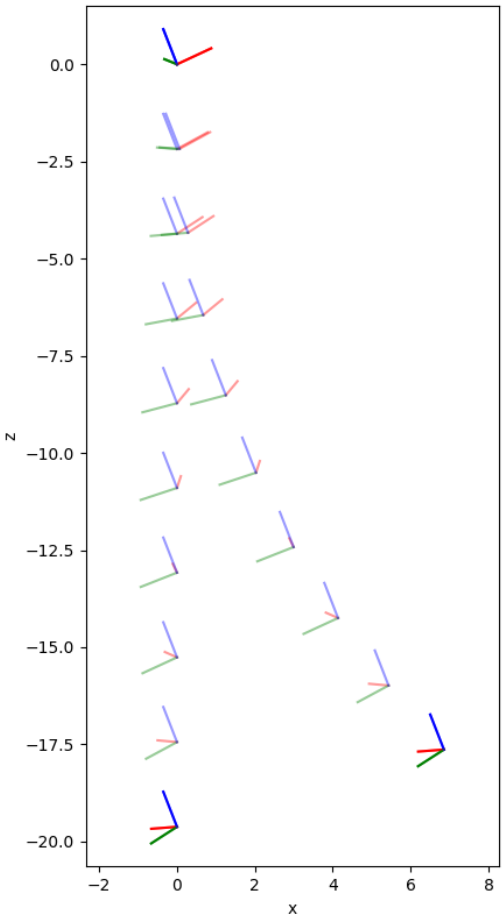


Figure 6: A body in free fall with zero initial velocity falls straight down. This is not the case when using the model of Equation 47, when the body has angular velocity. The effect diminishes with smaller time steps.

3.6 Uncertainty on Lie groups

Due to the over-parameterization of the groups, the uncertainty is defined in the Exponential coordinates. Gaussian distributions in the tangent space model the true uncertainty well.[12][19][20] In this work, zero-mean Gaussians in the local coordinates of the pose estimate will be used, as this is the frame which the IMU measurements are expressed. This is equivalent to right addition with a zero-mean Gaussian variable. As with addition, a global parameterization can also be used, but is less practical for this use-case. Following [9] the following definitions are used

$$\begin{aligned} \boldsymbol{T} &= \hat{\boldsymbol{T}} \operatorname{Exp}(\boldsymbol{\delta T}) = \hat{\boldsymbol{T}} \oplus \boldsymbol{\delta T}, \\ \boldsymbol{\delta T} &= \operatorname{Log}(\hat{\boldsymbol{T}}^{-1}\boldsymbol{T}) = \boldsymbol{T} \ominus \hat{\boldsymbol{T}} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}). \end{aligned} \tag{48}$$

$\boldsymbol{\delta T}$ is the random variable in the tangent space representing the error between the true, unknown state $\boldsymbol{T}$, and the estimate $\hat{\boldsymbol{T}}$. For this reason, it will be called the error coordinates or the Exponential coordinates. Since the normal distribution has probability mass over the entire $\mathbb{R}^n$, there is an issue with the wrap-around for rotational quantities. All of the probability mass for the rotational part of the vector should be within the ball of radius π as points outside of correspond to points inside. This means the probability of a rotation must be summed over all infinite points in $\mathbb{R}^n$ mapping to that rotation. Assuming the distribution to have sufficiently low variance, this can be ignored as the probability mass outside of the ball is vanishingly small. These distributions are known as concentrated Gaussians.[21] In practice, this is the case for all our purposes, as the uncertainty is rarely large enough for the error to be π radians off with any notable probability.

The distribution on the group can be defined from Equation 48. This is done similarly to [20], but

with the error defined locally instead of globally. We can define the probability distribution on the group using Equation 48 and the Exponential map. Since $\boldsymbol{T} = \hat{\boldsymbol{T}}\operatorname{Exp}(\boldsymbol{\delta T})$, a coordinate transform on the normal distribution of $\boldsymbol{\delta T}$ seems straight forward. Due to wrap-around, this is not the case, but let's ignore this for now. Applying Equation 19 to Equation 48 gives $d\boldsymbol{T} = |\mathcal{J}_r(\boldsymbol{\delta T})|d\boldsymbol{\delta T}$, where $|\mathcal{J}_r(\boldsymbol{\delta T})| = |\mathcal{J}_r(\boldsymbol{T} \ominus \hat{\boldsymbol{T}})|$ is the determinant of the right Jacobian at $\boldsymbol{\delta T}$. The determinant follows from the standard change of variables formula.

$$
\begin{aligned}
1 &= \int_{\mathbb{R}^n} \mathcal{N}(\boldsymbol{\delta T}; \mathbf{0}, \boldsymbol{\Sigma}) d\boldsymbol{\delta T} \\
&= \int_{\mathbb{R}^n} \frac{1}{\sqrt{(2\pi)^n |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2} \boldsymbol{\delta T}^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{\delta T} \right) d\boldsymbol{\delta T} \\
&= \int_{\mathcal{G}} \frac{1}{\sqrt{(2\pi)^n |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2} (\boldsymbol{T} \ominus \hat{\boldsymbol{T}})^\top \boldsymbol{\Sigma}^{-1} (\boldsymbol{T} \ominus \hat{\boldsymbol{T}}) \right) \frac{1}{|\mathcal{J}_r(\boldsymbol{T} \ominus \hat{\boldsymbol{T}})|} d\boldsymbol{T} \\
&= \int_{\mathcal{G}} \frac{1}{\sqrt{(2\pi)^n |\mathcal{J}_r(\boldsymbol{T} \ominus \hat{\boldsymbol{T}}) \boldsymbol{\Sigma} \mathcal{J}_r(\boldsymbol{T} \ominus \hat{\boldsymbol{T}})^\top|}} \exp\left(-\frac{1}{2} (\boldsymbol{T} \ominus \hat{\boldsymbol{T}})^\top \boldsymbol{\Sigma}^{-1} (\boldsymbol{T} \ominus \hat{\boldsymbol{T}}) \right) d\boldsymbol{T} \\
&= \int_{\mathcal{G}} \mathcal{N}_{\mathcal{G}}(\boldsymbol{T}; \hat{\boldsymbol{T}}, \boldsymbol{\Sigma}) d\boldsymbol{T}
\end{aligned} \tag{49}
$$

The probability distribution $\mathcal{N}_{\mathcal{G}}(\boldsymbol{T})$ is not Gaussian, since $\mathcal{J}_r$ is dependent on $\boldsymbol{T}$. n is the number of degrees of freedom of the group. For $\mathcal{G} = SE_2(3), n = 9$. Equation 49 shows that the probability distribution on the group is not the regular Gaussian, which has often been used.[22] $\mathcal{N}_{\mathcal{G}}(\boldsymbol{T})$ models the group distribution better, since the nonlinear properties of the group are respected in the distribution by the Jacobian. Actually evaluating $\mathcal{N}_{\mathcal{G}}(\boldsymbol{T})$ should not be done, and a coordinate transform to the error coordinates should be used instead. In our case, this is due to the wrap-around issue discussed earlier. Due to wrap-around, the rotational Logarithmic map is not the inverse of the Exponential map for all $\boldsymbol{\delta T}$. The Exponential map for rotations is many-to-one, and therefore has no inverse. The Logarithmic map for 3D rotations maps into the ball of radius π, meaning the Logarithmic map is only the inverse of the Exponential map for these values. This makes the coordinate transform used above invalid for $\boldsymbol{\delta T}$ outside of the ball. The concentrated Gaussian assumption address this issue, by allowing us to ignore the wrap-around. This makes the probability density of a group element only related to the corresponding unique error coordinate within the image of the Logarithm, not the other error coordinates mapping to the same element. Actually doing calculations with Equation 49 is not easy. Consider calculating the expected value of $\mathcal{N}_{\mathcal{G}}(\boldsymbol{T}; \hat{\boldsymbol{T}}, \boldsymbol{\Sigma})$;

$$
\begin{aligned}
E[\boldsymbol{T}] &= \int_{\mathcal{G}} \boldsymbol{T} \mathcal{N}_{\mathcal{G}}(\boldsymbol{T}; \hat{\boldsymbol{T}}, \boldsymbol{\Sigma}) d\boldsymbol{T} = \hat{\boldsymbol{T}} \int_{\mathbb{R}^n} \operatorname{Exp}(\boldsymbol{\delta T}) \mathcal{N}(\boldsymbol{\delta T}; \mathbf{0}, \boldsymbol{\Sigma}) d\boldsymbol{\delta T} \\
&\approx \hat{\boldsymbol{T}} \int_{\mathbb{R}^n} (\boldsymbol{I} + \boldsymbol{\delta T}^\wedge) \mathcal{N}(\boldsymbol{\delta T}; \mathbf{0}, \boldsymbol{\Sigma}) d\boldsymbol{\delta T} \\
&= \hat{\boldsymbol{T}}.
\end{aligned}
$$

To be able to evaluate the integral, the Exponential must be approximated. The first order approximation is valid due to the concentrated Gaussian assumption, meaning only small $\boldsymbol{\delta T}$ have meaningful probability density. If this is not case, there is no closed form solution to the integral. Defining the expectation as a value $\boldsymbol{T}^*$ fulfilling $E[\boldsymbol{T} \ominus \boldsymbol{T}^*] = \mathbf{0}$ for $\boldsymbol{T} \sim \mathcal{N}_{\mathcal{G}}(\hat{\boldsymbol{T}}, \boldsymbol{\Sigma})$, the only solution is $\boldsymbol{T}^* = \hat{\boldsymbol{T}}$, but this is only the case because the Logarithm maps onto vectors corresponding to unique group elements.

$$
\begin{aligned}
E[\boldsymbol{T} \ominus \hat{\boldsymbol{T}}] &= \int_{\mathcal{G}} (\boldsymbol{T} \ominus \hat{\boldsymbol{T}}) \mathcal{N}_{\mathcal{G}}(\boldsymbol{T}; \hat{\boldsymbol{T}}, \boldsymbol{\Sigma}) d\boldsymbol{T} \\
&= \int_{\mathbb{R}^n} \boldsymbol{\delta T} \mathcal{N}(\boldsymbol{\delta T}; \mathbf{0}, \boldsymbol{\Sigma}) d\boldsymbol{\delta T} \\
&= \mathbf{0}
\end{aligned}
$$

This approach to calculating the expectation can also be used to compute averages. Going back to the example in the introduction, considering the dots in the local coordinates of the blue dot the angle of the blue dot is zero. The other dots will be in the interval $[-\pi, \pi)$. Now the shortest

path is continuous on the interval, which means the average between the blue and green dots will be halfway between them along the shortest path. This is illustrated in Figure 7. Care must still be taken in the transforms back and forth from global to local coordinates, but this is elegantly taken care of by the Exponential and Logarithmic maps mapping between the circle and the real line.

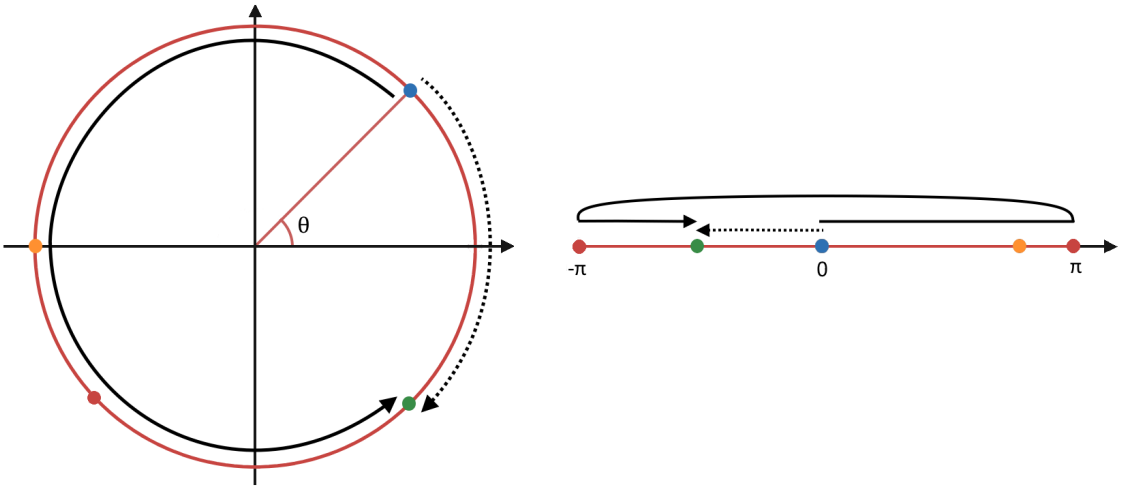


Figure 7: Using local coordinates around the blue dot, the shortest path from it to the other dots is continuous.

Estimating the expectation of a set of sampled states will be practical when evaluating the performance of the filter algorithm developed in the next subsections. For this reason, an algorithm for how to find the mean and covariance of the set is presented. For some sets, the mean is not unique, and the result will depend on the initial guess. The averaging example above is one such set. The approach is similar. The Logarithmic map maps deviations onto the tangent space at the initial guess. Computing the mean deviation is done as usual in the tangent space, and the Exponential map maps the mean back onto the group, where it is composed with the initial guess. Since both the Exponential and Logarithmic maps are nonlinear, the mean deviation of the tangent space does not correspond to the actual mean deviation of the set. However, by the nature of tangents, it should be a reasonable approximation. Therefore, iterating the process outlined above the updated guess should converge. Convergence is monitored by the mean deviation of the tangent space being sufficiently small. When a mean element is found, the covariance can be computed from the final deviations in the tangent space. The covariance is defined in the tangent space, and no iterations are therefore needed. There might exist multiple minima, meaning the solution is not necessarily unique. Usually the algorithm converges in few iterations (< 5).[19][23]

Algorithm 1 Estimating mean and covariance of a set of group elements

procedure FindMeanAndCovariance($\mathcal{X}_1, \ldots, \mathcal{X}_m, \bar{\mathcal{X}}, \epsilon$)
  repeat
    $\bar{\boldsymbol{\tau}} \leftarrow \frac{1}{m}\sum_i \mathcal{X}_i \ominus \bar{\mathcal{X}}$ ▷ Mean deviation on the tangent space at $\bar{\mathcal{X}}$
    $\bar{\mathcal{X}} \leftarrow \bar{\mathcal{X}} \oplus \bar{\boldsymbol{\tau}}$ ▷ Update guess
  until $||\bar{\boldsymbol{\tau}}|| < \epsilon$ ▷ Convergence when the mean deviation is sufficiently small
  $\boldsymbol{\tau}_i \leftarrow \mathcal{X}_i \ominus \bar{\mathcal{X}}$
  $\bar{\boldsymbol{\Sigma}} \leftarrow \frac{1}{m-1}\sum_i \boldsymbol{\tau}_i \boldsymbol{\tau}_i^\top$ ▷ Unbiased estimate of covariance
  return $\bar{\mathcal{X}}, \bar{\boldsymbol{\Sigma}}$
end procedure

3.7 Uncertainty on $\mathrm{SE}_2(3)$

IMU noise is modeled as zero-mean additive Gaussian noise with covariance $\tilde{\boldsymbol{Q}}$. The measured values, $\hat{\boldsymbol{\omega}}$ and $\hat{\boldsymbol{a}}$, are the true values, $\boldsymbol{\omega}$ and $\boldsymbol{a}$, corrupted by some noise $\boldsymbol{\eta}^{\omega}$ and $\boldsymbol{\eta}^{a}$.

$$\begin{aligned} \hat{\boldsymbol{\omega}}_k &= \boldsymbol{\omega}_k + \boldsymbol{\eta}^{\omega} \\ \hat{\boldsymbol{a}}_k &= \boldsymbol{a}_k + \boldsymbol{\eta}^{a}, \\ \boldsymbol{\eta} = [\boldsymbol{\eta}^{\omega}, \boldsymbol{\eta}^{a}]^{\top} &\sim \mathcal{N}(\boldsymbol{0}, \tilde{\boldsymbol{Q}}) \end{aligned} \tag{50}$$

Since the measurements are treated in the tangent space of $\mathrm{SE}_2(3)$, this motivates the choice for modeling uncertainty the way of Equation 48. Combining Equation 50 and Equation 46, the distribution of the increment matrix can be calculated.

$$\begin{aligned} \boldsymbol{\Delta}_k &= \operatorname{Exp}[\boldsymbol{\omega}_k \delta t, \boldsymbol{a}_k \delta t, \boldsymbol{C}(\boldsymbol{\omega}\delta t)\frac{1}{2}\boldsymbol{a}\delta t^2]^{\top} \\ &= \operatorname{Exp}[\hat{\boldsymbol{\omega}}_k \delta t - \boldsymbol{\eta}^{\omega}\delta t, \hat{\boldsymbol{a}}_k \delta t - \boldsymbol{\eta}^{a}\delta t, \boldsymbol{C}(\hat{\boldsymbol{\omega}}_k \delta t - \boldsymbol{\eta}^{\omega}\delta t)\frac{1}{2}(\hat{\boldsymbol{a}}_k \delta t^2 - \boldsymbol{\eta}^{a}\delta t^2)]^{\top} \\ &\approx \underbrace{\operatorname{Exp}([\hat{\boldsymbol{\omega}}_k \delta t, \hat{\boldsymbol{a}}_k \delta t, \boldsymbol{C}(\hat{\boldsymbol{\omega}}_k \delta t)\frac{1}{2}\hat{\boldsymbol{a}}_k \delta t^2]^{\top})}_{\operatorname{Exp}\boldsymbol{\upsilon}_k = \hat{\boldsymbol{\Delta}}_k} \times \\ &\qquad \operatorname{Exp}(\mathcal{J}_r(\boldsymbol{\upsilon}_k)[-\boldsymbol{\eta}^{\omega}\delta t, -\boldsymbol{\eta}^{a}\delta t, -\boldsymbol{C}(\hat{\boldsymbol{\omega}}_k \delta t)\frac{1}{2}\boldsymbol{\eta}^{a}\delta t^2]^{\top}) \\ &= \hat{\boldsymbol{\Delta}}_k \operatorname{Exp}(-\mathcal{J}_r(\boldsymbol{\upsilon}_k) \begin{bmatrix} \delta t \boldsymbol{I}_3 & \boldsymbol{0}_{3\times 3} \\ \boldsymbol{0}_{3\times 3} & \delta t \boldsymbol{I}_3 \\ \boldsymbol{0}_{3\times 3} & \frac{1}{2}\delta t^2 \boldsymbol{C}(\hat{\boldsymbol{\omega}}_k \delta t) \end{bmatrix} \begin{bmatrix} \boldsymbol{\eta}^{\omega} \\ \boldsymbol{\eta}^{a} \end{bmatrix}) \\ &= \hat{\boldsymbol{\Delta}}_k \operatorname{Exp}(-\begin{bmatrix} \mathcal{J}_r(\hat{\boldsymbol{\omega}}_k \delta t) & \boldsymbol{0}_{3\times 3} & \boldsymbol{0}_{3\times 3} \\ Q(-\hat{\boldsymbol{a}}_k \delta t, -\hat{\boldsymbol{\omega}}_k \delta t) & \mathcal{J}_r(\hat{\boldsymbol{\omega}}_k \delta t) & \boldsymbol{0}_{3\times 3} \\ Q(-\boldsymbol{C}(\hat{\boldsymbol{\omega}}_k \delta t)\frac{1}{2}\hat{\boldsymbol{a}}_k \delta t^2, -\hat{\boldsymbol{\omega}}_k \delta t) & \boldsymbol{0}_{3\times 3} & \mathcal{J}_r(\hat{\boldsymbol{\omega}}_k \delta t) \end{bmatrix} \times \\ &\qquad \begin{bmatrix} \delta_t \boldsymbol{I}_3 & \boldsymbol{0}_{3\times 3} \\ \boldsymbol{0}_{3\times 3} & \delta t \boldsymbol{I}_3 \\ \boldsymbol{0}_{3\times 3} & \frac{1}{2}\delta t^2 \boldsymbol{C}(\hat{\boldsymbol{\omega}}_k \delta t) \end{bmatrix} \begin{bmatrix} \boldsymbol{\eta}^{\omega} \\ \boldsymbol{\eta}^{a} \end{bmatrix}) \\ &= \hat{\boldsymbol{\Delta}}_k \operatorname{Exp}(-\begin{bmatrix} \mathcal{J}_r(\hat{\boldsymbol{\omega}}_k \delta t)\delta t & \boldsymbol{0}_{3\times 3} \\ Q(-\hat{\boldsymbol{a}}_k \delta t, -\hat{\boldsymbol{\omega}}_k \delta t)\delta t & \mathcal{J}_r(\hat{\boldsymbol{\omega}}_k \delta t)\delta t \\ Q(-\boldsymbol{C}(\hat{\boldsymbol{\omega}}_k \delta t)\frac{1}{2}\hat{\boldsymbol{a}}_k \delta t^2, -\hat{\boldsymbol{\omega}}_k \delta t)\delta t & \frac{1}{2}\delta t^2 \mathcal{J}_r(\hat{\boldsymbol{\omega}}_k \delta t)\boldsymbol{C}(\hat{\boldsymbol{\omega}}_k \delta t) \end{bmatrix} \begin{bmatrix} \boldsymbol{\eta}^{\omega} \\ \boldsymbol{\eta}^{a} \end{bmatrix}) \end{aligned} \tag{51}$$

The approximation $\boldsymbol{C}(\hat{\boldsymbol{\omega}}_k \delta t - \boldsymbol{\eta}^{\omega}\delta t) \approx \boldsymbol{C}(\hat{\boldsymbol{\omega}}_k \delta t)$ and Equation 20 has been used to achieve the desired form. The correction term $\boldsymbol{C}$ makes the equations intricate. A good approximation of Equation 51 is

$$\boldsymbol{\Delta}_k = \hat{\boldsymbol{\Delta}}_k \operatorname{Exp}(-\begin{bmatrix} \mathcal{J}_r(\hat{\boldsymbol{\omega}}_k \delta t)\delta t & \boldsymbol{0}_{3\times 3} \\ -\frac{1}{2}\hat{\boldsymbol{a}}_{k,\times}\delta t^2 & \mathcal{J}_r(\hat{\boldsymbol{\omega}}_k \delta t)\delta t \\ -\frac{1}{4}\hat{\boldsymbol{a}}_{k,\times}\delta t^3 & \frac{1}{2}\mathcal{J}_r(\hat{\boldsymbol{\omega}}_k \delta t)\delta t^2 \end{bmatrix} \begin{bmatrix} \boldsymbol{\eta}^{\omega} \\ \boldsymbol{\eta}^{a} \end{bmatrix}). \tag{52}$$

Similar calculations can be done for the forward Euler scheme for computing the increments given by Equation 28.

$$\begin{aligned} \boldsymbol{\Delta}_k &= \begin{bmatrix} \operatorname{Exp}(\boldsymbol{\omega}_k \delta t) & \boldsymbol{a}_k \delta t & \boldsymbol{a}_k \delta t^2/2 \\ \boldsymbol{0}_{2\times 3} & & \boldsymbol{I}_2 \end{bmatrix} \\ &\approx \begin{bmatrix} \operatorname{Exp}(\hat{\boldsymbol{\omega}}_k \delta t)\operatorname{Exp}(-\mathcal{J}_r(\hat{\boldsymbol{\omega}}_k \delta t)\boldsymbol{\eta}^{\omega}\delta t) & (\hat{\boldsymbol{a}}_k - \boldsymbol{\eta}^{a})\delta t & (\hat{\boldsymbol{a}}_k - \boldsymbol{\eta}^{a})\delta t^2/2 \\ \boldsymbol{0}_{2\times 3} & & \boldsymbol{I}_2 \end{bmatrix} \\ &= \begin{bmatrix} \operatorname{Exp}(\hat{\boldsymbol{\omega}}_k \delta t) & \hat{\boldsymbol{a}}_k \delta t & \hat{\boldsymbol{a}}_k \delta t^2/2 \\ \boldsymbol{0}_{2\times 3} & & \boldsymbol{I}_2 \end{bmatrix} \times \\ &\qquad \begin{bmatrix} \operatorname{Exp}(-\mathcal{J}_r(\hat{\boldsymbol{\omega}}_k \delta t)\boldsymbol{\eta}^{\omega}\delta t) & -\operatorname{Exp}(-\hat{\boldsymbol{\omega}}_k \delta t)\boldsymbol{\eta}^{a}\delta t & -\operatorname{Exp}(-\hat{\boldsymbol{\omega}}_k \delta t)\boldsymbol{\eta}^{a}\delta t^2/2 \\ \boldsymbol{0}_{2\times 3} & & \boldsymbol{I}_2 \end{bmatrix} \\ &= \hat{\boldsymbol{\Delta}}_k \operatorname{Exp}(-\begin{bmatrix} \mathcal{J}_r(\hat{\boldsymbol{\omega}}_k \delta t)\delta t \boldsymbol{\eta}^{\omega} \\ \mathcal{J}_l^{-1}(\mathcal{J}_r(\hat{\boldsymbol{\omega}}_k \delta t)\delta t \boldsymbol{\eta}^{\omega})\operatorname{Exp}(-\hat{\boldsymbol{\omega}}_k \delta t)\boldsymbol{\eta}^{a}\delta t \\ \mathcal{J}_l^{-1}(\mathcal{J}_r(\hat{\boldsymbol{\omega}}_k \delta t)\delta t \boldsymbol{\eta}^{\omega})\operatorname{Exp}(-\hat{\boldsymbol{\omega}}_k \delta t)\boldsymbol{\eta}^{a}\delta t^2/2 \end{bmatrix}) \\ &\approx \hat{\boldsymbol{\Delta}}_k \operatorname{Exp}(-\begin{bmatrix} \mathcal{J}_r(\hat{\boldsymbol{\omega}}_k \delta t)\delta t & \boldsymbol{0}_{3\times 3} \\ \boldsymbol{0}_{3\times 3} & \operatorname{Exp}(-\hat{\boldsymbol{\omega}}_k \delta t)\delta t \\ \boldsymbol{0}_{3\times 3} & \frac{1}{2}\operatorname{Exp}(-\hat{\boldsymbol{\omega}}_k \delta t)\delta t^2 \end{bmatrix} \begin{bmatrix} \boldsymbol{\eta}^{\omega} \\ \boldsymbol{\eta}^{a} \end{bmatrix}) \end{aligned} \tag{53}$$

A first order expansion of the rotation has been used in the first approximation. The last approximation follows again from assuming all second order terms of δt to be sufficiently small, so the inverse left Jacobian is close to identity. It is interesting to see that the approach to the simplification substitutes the Jacobian for the Exponential map.

Define the new noise variable

$$\boldsymbol{\vartheta}_k = \boldsymbol{D}_k \boldsymbol{\eta}, \tag{54}$$

where $\boldsymbol{D}_k$ is one of the matrices from Equation 51, Equation 52, or Equation 53. Then $\boldsymbol{\vartheta}_k$ is normally distributed according to $\mathcal{N}(\mathbf{0}, \boldsymbol{Q}_k)$, where

$$\boldsymbol{Q}_k = E[\boldsymbol{\vartheta}_k \boldsymbol{\vartheta}_k^\top] = E[\boldsymbol{D}_k \boldsymbol{\eta} (\boldsymbol{D}_k \boldsymbol{\eta})^\top] = E[\boldsymbol{D}_k \boldsymbol{\eta}\boldsymbol{\eta}^\top \boldsymbol{D}_k^\top] = \boldsymbol{D}_k E[\boldsymbol{\eta}\boldsymbol{\eta}^\top] \boldsymbol{D}_k^\top = \boldsymbol{D}_k \tilde{\boldsymbol{Q}} \boldsymbol{D}_k^\top. \tag{55}$$

The covariance $\tilde{\boldsymbol{Q}}$ is a property of the IMU. $\boldsymbol{\vartheta}$ is zero-mean since $\boldsymbol{\eta}$ is assumed zero-mean.

$$E[\boldsymbol{\vartheta}_k] = E[\boldsymbol{D}_k \boldsymbol{\eta}] = \boldsymbol{D}_k E[\boldsymbol{\eta}] = \mathbf{0}$$

Using the above derivations and Equation 49, the distribution of $\boldsymbol{\Delta}$ at time step k is given by $\mathcal{N}_{\mathcal{G}}(\boldsymbol{\Delta}_k; \hat{\boldsymbol{\Delta}}_k, \boldsymbol{Q}_k)$. This can also be written using random variables.

$$\boldsymbol{\Delta}_k = \hat{\boldsymbol{\Delta}}_k \oplus \boldsymbol{\vartheta}_k \tag{56}$$

In addition to the IMU, a GNSS sensor will be used. The sensor measures the position of the body with some zero-mean additive noise.

$$\boldsymbol{p}_m = h(\boldsymbol{T}) = \boldsymbol{p} + \boldsymbol{n}, \quad \boldsymbol{n} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{N}) \tag{57}$$

In contrast to the IMU sensor, the GNSS sensor measures in the global frame.

3.8 Filter propagation

At this point, all the tools are presented to create a filter that estimates the state and uncertainty of a body on SE_2 (3). In this and the next subsection we derive a Kalman filter on $\mathrm{SE}_2(3)$. The goal of the filter is to optimally estimate the state of the body and the uncertainty of that state. As the equations of inertial navigation are non-linear, an Extended Kalman filter might be a natural thought, but on $\mathrm{SE}_2(3)$, we can do better.[9]

The notation and vocabulary used for the filter are inspired by the usual Kalman filter. There are two types of inputs used. In the propagation step, the IMU measurements are used with the inertial navigation equations in place of the typical motion model. The state at time step k is $\boldsymbol{T}_k = \hat{\boldsymbol{T}}_k \oplus \boldsymbol{\delta T}_k$. The filter then propagates the state to $\boldsymbol{T}_{k+1|k} = \hat{\boldsymbol{T}}_{k+1|k} \oplus \boldsymbol{\delta T}_{k+1|k}$. The update step follows the propagation step giving $\boldsymbol{T}_{k+1} = \hat{\boldsymbol{T}}_{k+1} \oplus \boldsymbol{\delta T}_{k+1}$. In practice, the propagation step will be repeated many times per update step, as IMU measurements are much more frequent than GNSS measurements.

Propagating the mean takes the expected form of Equation 42, in spite of the noise.

$$\begin{aligned}
\boldsymbol{T}_{k+1|k} &\overset{(42)}{=} \boldsymbol{G}_{\delta t} f_{\delta t}(\boldsymbol{T}_k) \boldsymbol{\Delta}_k \\
&\overset{(48,56)}{=} \boldsymbol{G}_{\delta t} f_{\delta t}(\hat{\boldsymbol{T}}_k \oplus \boldsymbol{\delta T}_k) \hat{\boldsymbol{\Delta}}_k \oplus \boldsymbol{\vartheta}_k \\
&\overset{(41)}{=} \boldsymbol{G}_{\delta t} f_{\delta t}(\hat{\boldsymbol{T}}_k) \operatorname{Exp}(\boldsymbol{F}_{\delta t} \boldsymbol{\delta T}_k) \hat{\boldsymbol{\Delta}}_k \oplus \boldsymbol{\vartheta}_k \\
&\overset{(22)}{=} \underbrace{\boldsymbol{G}_{\delta t} f_{\delta t}(\hat{\boldsymbol{T}}_k) \hat{\boldsymbol{\Delta}}_k}_{\hat{\boldsymbol{T}}_{k+1|k}} \operatorname{Exp}(\operatorname{Ad}_{\hat{\boldsymbol{\Delta}}_k^{-1}} \boldsymbol{F}_{\delta t} \boldsymbol{\delta T}_k) \operatorname{Exp} \boldsymbol{\vartheta}_k
\end{aligned} \tag{58}$$

Propagating the uncertainty is less straight forward. $\operatorname{Exp}(\operatorname{Ad}_{\hat{\boldsymbol{\Delta}}_k^{-1}} \boldsymbol{F}_{\delta t} \boldsymbol{\delta T}_k) \operatorname{Exp} \boldsymbol{\vartheta}_k$ needs to be approximated as a single Exponential for Equation 58 to be on the form of $\boldsymbol{T}_{k+1|k} = \hat{\boldsymbol{T}}_{k+1|k} \oplus \boldsymbol{\delta T}_{k+1|k}$. Since the normal product of exponentials rule does not apply in higher dimensions, the

exponents cannot simply be added together. In the case of noise free IMU measurements, $\boldsymbol{\vartheta} = \mathbf{0}$, Equation 58 shows that the propagation step of the filter preserves the Gaussian on the error coordinates with the transform $\boldsymbol{\delta T}_{k+1|k} = \mathrm{Ad}_{\hat{\boldsymbol{\Delta}}_k^{-1}} \boldsymbol{F}_{\delta t} \boldsymbol{\delta T}_k$. This is remarkable as the propagation step is non-linear.

In [9], the uncertainty of Equation 58 is shown to be zero-mean Gaussian in the error coordinates to the third order. This makes the approximation nearly exact. The proof uses the non-trivial Baker–Campbell–Hausdorff formula, which is the solution to the composition of Exponentials, i.e. $\operatorname{Exp} \boldsymbol{x} \circ \operatorname{Exp} \boldsymbol{y} = \operatorname{Exp} \boldsymbol{z}$ solved for $\boldsymbol{z}$.[24] The proof is not repeated here. To the third order in the expectation, and the fourth order in the covariance, the uncertainty is Gaussian with moments

$$\begin{aligned} \mathrm{E}[\boldsymbol{\delta T}_{k+1|k}] &\approx \mathbf{0}, \\ \mathrm{E}[\boldsymbol{\delta T}_{k+1|k} \boldsymbol{\delta T}_{k+1|k}^\top] = \boldsymbol{\Sigma}_{k+1|k} &\approx \mathrm{Ad}_{\boldsymbol{\Delta}_k^{-1}} \boldsymbol{F}_{\delta t} \boldsymbol{\Sigma}_k (\mathrm{Ad}_{\boldsymbol{\Delta}_k^{-1}} \boldsymbol{F}_{\delta t})^\top + \boldsymbol{Q}_k + \boldsymbol{S}_{4th}. \end{aligned}$$

The fourth order contribution $\boldsymbol{S}_{4th}$ is important to capture the uncertainty dynamics with reasonable accuracy.[9] Ignoring the fourth order term is equivalent to ignoring that the Exponential does not distribute over addition, and calculating $\operatorname{Exp}(\mathrm{Ad}_{\hat{\boldsymbol{\Delta}}_k^{-1}} \boldsymbol{F}_{\delta t} \boldsymbol{\delta T}_k) \operatorname{Exp} \boldsymbol{\vartheta}_k \approx \operatorname{Exp}(\mathrm{Ad}_{\hat{\boldsymbol{\Delta}}_k^{-1}} \boldsymbol{F}_{\delta t} \boldsymbol{\delta T}_k + \boldsymbol{\vartheta}_k)$. The definition of the fourth order term can be found in Appendix A. The propagation step of the filter takes the form

$$\begin{aligned} \hat{\boldsymbol{T}}_{k+1|k} &= \boldsymbol{G}_{\delta t} f_{\delta t}(\hat{\boldsymbol{T}}_k) \hat{\boldsymbol{\Delta}}_k, \\ \boldsymbol{\Sigma}_{k+1|k} &= \mathrm{Ad}_{\hat{\boldsymbol{\Delta}}_k^{-1}} \boldsymbol{F}_{\delta t} \boldsymbol{\Sigma}_k (\mathrm{Ad}_{\hat{\boldsymbol{\Delta}}_k^{-1}} \boldsymbol{F}_{\delta t})^\top + \boldsymbol{Q}_k + \boldsymbol{S}_{4th}. \end{aligned} \tag{59}$$

The entire propagation step is summarized in Algorithm 2.

Algorithm 2 The propagation step

procedure PROPAGATE($\hat{\boldsymbol{T}}_{\boldsymbol{k}}$, $\boldsymbol{\Sigma}_k$, $\hat{\boldsymbol{\omega}}_k$, $\hat{\boldsymbol{a}}_k$, δt_k, $\tilde{\boldsymbol{Q}}$)
- $\hat{\boldsymbol{\Delta}}_k \leftarrow \boldsymbol{\Delta}(\hat{\boldsymbol{\omega}}_k, \hat{\boldsymbol{a}}_k, \delta t_k)$ ▷ Equation 46
- $\boldsymbol{G}_{\delta t_k} \leftarrow \boldsymbol{G}(\delta t_k)$ ▷ Equation 43
- $f_{\delta t_k}(\hat{\boldsymbol{T}}_k) \leftarrow f(\hat{\boldsymbol{T}}_k, \delta t_k)$ ▷ Equation 40
- $\hat{\boldsymbol{T}}_{k+1|k} \leftarrow \boldsymbol{G}_{\delta t_k} f_{\delta t_k}(\hat{\boldsymbol{T}}_k) \hat{\boldsymbol{\Delta}}_k$ ▷ Equation 58
- $\boldsymbol{D}_k \leftarrow \boldsymbol{D}(\hat{\boldsymbol{\omega}}_k, \hat{\boldsymbol{a}}_k, \delta t)$ ▷ Equation 51, Equation 52, or Equation 53
- $\boldsymbol{Q}_k \leftarrow \boldsymbol{D}_k \tilde{\boldsymbol{Q}} \boldsymbol{D}_k^\top$ ▷ Equation 55
- $\boldsymbol{F}_{\delta t} \leftarrow \boldsymbol{F}(\delta t)$ ▷ Equation 41
- $\mathrm{Ad}_{\hat{\boldsymbol{\Delta}}_k^{-1}} \leftarrow \mathrm{Ad}(\hat{\boldsymbol{\Delta}}_k^{-1})$ ▷ Equation 39
- $\boldsymbol{\Sigma} \leftarrow \mathrm{Ad}_{\hat{\boldsymbol{\Delta}}_k^{-1}} \boldsymbol{F}_{\delta t} \boldsymbol{\Sigma}_k (\mathrm{Ad}_{\hat{\boldsymbol{\Delta}}_k^{-1}} \boldsymbol{F}_{\delta t})^\top$ ▷ Equation 59
- $\boldsymbol{S}_{4\mathrm{th}} \leftarrow \boldsymbol{S}_{4\mathrm{th}}(\boldsymbol{\Sigma}, \boldsymbol{Q}_k)$ ▷ Equation 99
- $\boldsymbol{\Sigma}_{k+1|k} \leftarrow \boldsymbol{\Sigma} + \boldsymbol{Q}_k + \boldsymbol{S}_{4\mathrm{th}}$ ▷ Equation 59
- **return** $\hat{\boldsymbol{T}}_{k+1|k}, \boldsymbol{\Sigma}_{k+1|k}$

end procedure

The difference between using the propagation step of an Extended Kalman filter (EKF) on the error coordinates and the filter derived here is the resulting covariance. To apply the EKF propagation, the error dynamics would have to be linearized. This is not done here. The group properties allow for an almost closed form solution preserving Gaussianity on the error coordinates. The error-state Kalman filter of Sola, takes the linearization approach.[8]

The filter is a version of the Left Invariant Extended Kalman filter (LIEKF).[10][11] The filter gets its name from the invariance to any left multiplications, as they cancel in the error calculations. From Equation 58, the error is easily identified. The error dynamics is approximately linear. It is also independent of the state, which is a property of the regular Kalman filter the Extended

Kalman filter does not share.

$$\begin{aligned}
\hat{\boldsymbol{T}}_{k+1|k}^{-1}\boldsymbol{T}_{k+1} &= \hat{\boldsymbol{\Delta}}_k^{-1} f_{\delta t}(\hat{\boldsymbol{T}}_k)^{-1}\boldsymbol{G}_{\delta t}^{-1}\boldsymbol{G}_{\delta t} f_{\delta t}(\boldsymbol{T}_k)\boldsymbol{\Delta}_k \\
&= \hat{\boldsymbol{\Delta}}_k^{-1} f_{\delta t}(\hat{\boldsymbol{T}}_k^{-1}\boldsymbol{T}_k)\boldsymbol{\Delta}_k \\
\Rightarrow \boldsymbol{\delta T}_{k+1|k} &= \operatorname{Log}\left(\hat{\boldsymbol{\Delta}}_k^{-1} f_{\delta t}(\operatorname{Exp}\boldsymbol{\delta T}_k)\boldsymbol{\Delta}_k\right) = \operatorname{Log}\left(\hat{\boldsymbol{\Delta}}_k^{-1}\operatorname{Exp}(\boldsymbol{F}_{\delta t}\boldsymbol{\delta T}_k)\boldsymbol{\Delta}_k\right) \\
&= \operatorname{Log}\left(\operatorname{Exp}(\operatorname{Ad}_{\hat{\boldsymbol{\Delta}}_k^{-1}}\boldsymbol{F}_{\delta t}\boldsymbol{\delta T}_k)\operatorname{Exp}\boldsymbol{\vartheta}_k\right) \\
&\approx \operatorname{Ad}_{\hat{\boldsymbol{\Delta}}_k^{-1}}\boldsymbol{F}_{\delta t}\boldsymbol{\delta T}_k + \boldsymbol{\vartheta}_k.
\end{aligned} \tag{60}$$

3.9 Filter update

With the propagation step of the filter preserving Gaussianity on the error coordinates, half the filter is done. The next step is the filter update. An update is done when additional information is perceived about the state of the system. This could for example be a position estimate from a GNSS sensor or a bearing from a compass. For completeness, the filter is derived for an arbitrary measurement function.

A measurement function is a function from the state manifold to the measurement manifold. It describes what the measurement should be, given the state of the system. Let $\mathcal{X}$ denote the the state manifold and $\mathcal{Z}$ denote the measurement manifold, then $\boldsymbol{h} : \mathcal{X} \to \mathcal{Z}$. For clarity, the Exponential and Logarithmic maps are subscripted with the manifold they belong to. Using additive noise and the abstraction provided by the Lie theory notation, the model takes a form similar to the measurement model of an Extended Kalman filter.

$$\boldsymbol{z} = \boldsymbol{h}(\boldsymbol{T}) \oplus \boldsymbol{n} = \boldsymbol{h}(\boldsymbol{T}) \circ \operatorname{Exp}_{\mathcal{Z}} \boldsymbol{n}, \qquad \boldsymbol{n} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{N}) \tag{61}$$

Since the probability distribution of $\boldsymbol{z}$ depends on an arbitrary function $\boldsymbol{h}$, it is generally not Gaussian. To able to use a Kalman filter update, the function must be linearized. Using Equation 19 the measurement function can be approximated to the first order as

$$\begin{aligned}
\boldsymbol{z} &= \boldsymbol{h}(\boldsymbol{T}) \oplus \boldsymbol{n} = \boldsymbol{h}(\hat{\boldsymbol{T}} \oplus \boldsymbol{\delta T}) \oplus \boldsymbol{n} \\
&\approx \boldsymbol{h}(\hat{\boldsymbol{T}}) \oplus \boldsymbol{H}\boldsymbol{\delta T} \oplus \boldsymbol{n},
\end{aligned} \tag{62}$$

where $\boldsymbol{H}$ is the Jacobian of the measurement function

$$\boldsymbol{H} = \frac{\partial}{\partial \boldsymbol{T}}\boldsymbol{h}(\boldsymbol{T})\bigg|_{\boldsymbol{T}=\hat{\boldsymbol{T}}} = \boldsymbol{J}_{\boldsymbol{T}}^{\boldsymbol{h}(\boldsymbol{T})}(\hat{\boldsymbol{T}}). \tag{63}$$

Note that the $\oplus$ within the $\boldsymbol{h}$ function and the one outside are not the same operations. They work on different manifolds, making their underlying composition $\circ$ and Exponential operations different. Some caution must be taken when evaluating Equation 62 as the rightmost plus does not act on $\boldsymbol{H}\boldsymbol{\delta T}$, but $\oplus\boldsymbol{H}\boldsymbol{\delta T}$.

From Equation 62 it can be seen that the expectation of $\boldsymbol{z}$ is $\hat{\boldsymbol{z}} = \boldsymbol{h}(\hat{\boldsymbol{T}})$, since both $\boldsymbol{\delta T}$ and $\boldsymbol{n}$ are zero mean. Define the innovation term $\boldsymbol{\lambda}$ as the difference between the expected and actual value of the measurement.

$$\begin{aligned}
\boldsymbol{\lambda} &= \boldsymbol{z} \ominus \hat{\boldsymbol{z}} = \boldsymbol{z} \ominus \boldsymbol{h}(\hat{\boldsymbol{T}}) = \operatorname{Log}_{\mathcal{Z}}(\boldsymbol{h}(\hat{\boldsymbol{T}})^{-1} \circ \boldsymbol{z}) \\
&= \operatorname{Log}_{\mathcal{Z}}(\boldsymbol{h}(\hat{\boldsymbol{T}})^{-1} \circ \boldsymbol{h}(\boldsymbol{T}) \circ \operatorname{Exp}_{\mathcal{Z}} \boldsymbol{n}) \\
&= \operatorname{Log}_{\mathcal{Z}}(\boldsymbol{h}(\hat{\boldsymbol{T}})^{-1} \circ \boldsymbol{h}(\hat{\boldsymbol{T}} \circ \operatorname{Exp}_{\mathcal{X}} \boldsymbol{\delta T}) \circ \operatorname{Exp}_{\mathcal{Z}} \boldsymbol{n}) \\
&\approx \operatorname{Log}_{\mathcal{Z}}(\boldsymbol{h}(\hat{\boldsymbol{T}})^{-1} \circ \boldsymbol{h}(\hat{\boldsymbol{T}}) \oplus \boldsymbol{H}\boldsymbol{\delta T} \oplus \boldsymbol{n}) \\
&= \operatorname{Log}_{\mathcal{Z}}(\operatorname{Exp}_{\mathcal{Z}} \boldsymbol{H}\boldsymbol{\delta T} \circ \operatorname{Exp}_{\mathcal{Z}} \boldsymbol{n}) \\
&\approx \boldsymbol{H}\boldsymbol{\delta T} + \boldsymbol{n}
\end{aligned} \tag{64}$$

The last approximation follows from both $\boldsymbol{\delta T}$ and $\boldsymbol{n}$ being assumed small. It can be seen from using Equation 20 in reverse, which is accurate when $\boldsymbol{n}$ is small, and noting that the inverse Jacobian

is close to identity since $\boldsymbol{\delta T}$ is small. Finally, the Logarithmic and Exponential maps cancel. At this point the process is similar to a regular Kalman filter update step, since the problem has been transferred to the vector space of the error state. Using Equation 64, the covariance of the innovation is given by $\boldsymbol{\Lambda} = \boldsymbol{H\Sigma H}^\top + \boldsymbol{N}$.

When using a GNSS sensor, the measurement function extracts the position from the state matrix. The manifold of the position is the trivial manifold $(\mathbb{R}^3, +)$.

$$\begin{aligned} &\boldsymbol{h} : SE_2(3) \to (\mathbb{R}^3, +), \\ &\boldsymbol{h}(\boldsymbol{T}) = \boldsymbol{p} + \boldsymbol{n} \end{aligned} \tag{65}$$

On $(\mathbb{R}^3, +)$ the inverse is the negative element, and $\oplus, \ominus$ is the regular $+, -$ as the composition operation $\circ$ is addition. The Exponential and Logarithmic maps are identity maps, since the manifold itself is linear. The mesurement Jacobian is given by

$$\begin{aligned} \boldsymbol{H}(\boldsymbol{T}) = \boldsymbol{J}_{\boldsymbol{T}}^{\boldsymbol{h}(\boldsymbol{T})} &:= \lim_{\boldsymbol{\tau}\to\boldsymbol{0}} \frac{\boldsymbol{h}(\boldsymbol{T} \oplus \boldsymbol{\tau}) \ominus \boldsymbol{h}(\boldsymbol{T})}{\boldsymbol{\tau}} \\ &= \lim_{\boldsymbol{\tau}\to\boldsymbol{0}} \frac{\boldsymbol{h}(\boldsymbol{T} \circ \operatorname{Exp} \boldsymbol{\tau}) - \boldsymbol{p}}{\boldsymbol{\tau}} \\ &= \lim_{\boldsymbol{\tau}\to\boldsymbol{0}} \frac{\boldsymbol{R}\mathcal{J}_l(\boldsymbol{\theta})\boldsymbol{\rho} + \boldsymbol{p} - \boldsymbol{p}}{\boldsymbol{\tau}} \\ &= \lim_{\boldsymbol{\tau}\to\boldsymbol{0}} \frac{\boldsymbol{R}\mathcal{J}_l(\boldsymbol{\theta})\boldsymbol{\rho}}{\boldsymbol{\tau}} \\ &= \boldsymbol{R}\begin{bmatrix}\boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} & \boldsymbol{I}_{3\times3}\end{bmatrix} \end{aligned} \tag{66}$$

The final equality follows from the Jacobian going to identity as $\boldsymbol{\tau}$ tends to zero. The result makes the Jacobian a function of the state $\boldsymbol{T}$ the linearization is performed at, because the rotation matrix is a part of this state. The rotation matrix transforms from local to global frame. We can remove it by redefining the measurement function to measure in the local frame, i.e. $\boldsymbol{h}(\boldsymbol{T}) = \boldsymbol{R}^\top(\boldsymbol{p} + \boldsymbol{n})$, which makes the Jacobian constant. However, the measurement noise is then not constant as the GNSS noise is typically related to global directions. One thing to note about this example is that the final approximation of Equation 64 is actually exact in this case, it follows directly from inserting the appropriate functions and operations from $(\mathbb{R}^3, +)$.

The goal of the update step is to find the minimum variance unbiased linear estimator for the error, $\hat{\boldsymbol{\delta T}} = \boldsymbol{K\lambda}$. Using the LIEKF methodology of [11], the group action is defined to match the measurement function $\boldsymbol{z}_k = \boldsymbol{T}_k \cdot \boldsymbol{b}_k + \boldsymbol{n}$, where $\boldsymbol{b}_k$ is some known vector. For GNSS measurements, this is the standard group action on $\mathrm{SE}_2(3)$ on the zero vector, $\boldsymbol{z}_k = \boldsymbol{T}_k \cdot \boldsymbol{0} + \boldsymbol{n} = \boldsymbol{R0} + \boldsymbol{p} + \boldsymbol{n} = \boldsymbol{p} + \boldsymbol{n}$. This makes the innovation term $\boldsymbol{\lambda}_k = \hat{\boldsymbol{T}}_k^{-1} \cdot \boldsymbol{z}_k - \boldsymbol{b}_k$, where it is assumed the action maps to a vector space. With the estimator $\hat{\boldsymbol{\delta T}} = \boldsymbol{K\lambda}$, the error after the update step is

$$\begin{aligned} \boldsymbol{e}_{k+1} = \hat{\boldsymbol{T}}_{k+1}^{-1}\boldsymbol{T}_{k+1} &= \operatorname{Exp}(-\hat{\boldsymbol{\delta T}}_{k+1})\hat{\boldsymbol{T}}_{k+1|k}^{-1}\boldsymbol{T}_{k+1} \\ &= \operatorname{Exp}\Big(- \boldsymbol{K}(\hat{\boldsymbol{T}}_{k+1|k}^{-1} \cdot \boldsymbol{z}_k - \boldsymbol{b}_k)\Big)\boldsymbol{e}_{k+1|k} \\ &= \operatorname{Exp}\Big(- \boldsymbol{K}(\hat{\boldsymbol{T}}_{k+1|k}^{-1}(\boldsymbol{T}_k \cdot \boldsymbol{b}_k + \boldsymbol{n}_k) - \boldsymbol{b}_k)\Big)\boldsymbol{e}_{k+1|k} \\ &= \operatorname{Exp}\Big(- \boldsymbol{K}(\boldsymbol{e}_{k+1|k} \cdot \boldsymbol{b}_k - \boldsymbol{b}_k + \hat{\boldsymbol{n}}_k)\Big)\boldsymbol{e}_{k+1|k}. \end{aligned} \tag{67}$$

where the transformation of the measurement noise $\hat{\boldsymbol{n}} = \hat{\boldsymbol{R}}^\top \boldsymbol{n}$ has been used. In the tangent space the error is

$$\begin{aligned} \boldsymbol{\delta T}_{k+1} &= \operatorname{Log}\left[\operatorname{Exp}\Big(- \boldsymbol{K}(\operatorname{Exp}\boldsymbol{\delta T}_{k+1|k} \cdot \boldsymbol{b}_k - \boldsymbol{b}_k + \hat{\boldsymbol{n}}_k)\Big)\operatorname{Exp}\boldsymbol{\delta T}_{k+1|k}\right] \\ &\approx \boldsymbol{\delta T}_{k+1|k} - \boldsymbol{K}(\boldsymbol{\delta T}_{k+1|k}^{\wedge} \cdot \boldsymbol{b}_k + \hat{\boldsymbol{n}}_k) \\ &= \boldsymbol{\delta T}_{k+1|k} - \boldsymbol{K}(\boldsymbol{H\delta T}_{k+1|k} + \hat{\boldsymbol{n}}_k) \\ &= (\boldsymbol{I} - \boldsymbol{KH})\boldsymbol{\delta T}_{k+1|k} - \boldsymbol{K}\hat{\boldsymbol{n}}_k, \end{aligned} \tag{68}$$

where $\boldsymbol{H} = [-\boldsymbol{b}_\times, \boldsymbol{0}_3, \boldsymbol{I}_3]$. In the GNSS case this is $[\boldsymbol{0}_3, \boldsymbol{0}_3, \boldsymbol{I}_3]$. Here the GNSS noise is in local coordinates, which gives the amended measurement matrix discussed earlier. The two measurement and noise definitions are entirely equivalent, as the $\boldsymbol{K}$ matrix will compensate for the difference.

With this linear system, standard Kalman theory applies to calculate the gain matrix $\boldsymbol{K}$. As the measurement function and error dynamics are only approximated as linear on the error coordinates, the estimate is not necessarily optimal. The on-manifold linearization followed by a regular Kalman filter update is an on-manifold Extended Kalman filter update. The Extended Kalman filter update can be expected to work reasonably well when the posterior density can be approximated as Gaussian, even though nonlinear transforms generally do not preserve Gaussianity. In the case of a GNSS update on a Gaussian prior, this is a reasonable assumption, as shown by Equation 68. The Kalman filter can be derived in multiple ways. [25] shows how the filter is the minimum-variance unbiased linear estimator, from two different approaches. Here, a perspective on the EKF as the solution of a single step of the Gauss-Newton method on a specific optimization problem is offered.

From a Bayesian perspective the goal of the filter is to maximize the posterior distribution

$$p(\boldsymbol{\delta T}_{k+1}|\boldsymbol{z}_{1:k+1}) \propto p(\boldsymbol{z}_{k+1}|\boldsymbol{\delta T}_{k+1})p(\boldsymbol{\delta T}_{k+1}|\boldsymbol{z}_{1:k}). \tag{69}$$

where Bayes rule and some Markov assumptions are used. For a thorough walk-through of the standard Kalman filter from a Bayesian viewpoint, see [26]. The first probability distribution is the measurement model, $p(\boldsymbol{z}_{k+1}|\boldsymbol{\delta T}_{k+1}) = \mathcal{N}(\boldsymbol{z}_{k+1}; \boldsymbol{h}(\hat{\boldsymbol{T}}_{k+1|k} \oplus \boldsymbol{\delta T}_{k+1}), \boldsymbol{N}) = \mathcal{N}(\boldsymbol{z}_{k+1}; \boldsymbol{h}(\boldsymbol{T}_{k+1}), \boldsymbol{N})$. This is a distribution on the measurement manifold, which is now assumed to be a vector space for simplicity. The last distribution is the prior, the output of the propagation step; $p(\boldsymbol{\delta T}_{k+1}|\boldsymbol{z}_{1:k}) = \mathcal{N}(\boldsymbol{\delta T}_{k+1}; \boldsymbol{0}, \boldsymbol{\Sigma}_{k+1|k})$. Maximizing the posterior in Equation 69 is the same as maximizing the logarithm of the posterior, as the distributions are greater than zero everywhere and the logarithm is strictly increasing. For notational brevity, the Mahalanobis norm $||\cdot||_{\boldsymbol{\Sigma}}$ is used to express the quadratic forms of the Gaussians.

$$\begin{aligned}
\hat{\boldsymbol{\delta T}}_{k+1} &= \max_{\boldsymbol{\delta T}_{k+1}} \log p(\boldsymbol{\delta T}_{k+1}|\boldsymbol{z}_{1:k+1}) \\
&= \max_{\boldsymbol{\delta T}_{k+1}} \log p(\boldsymbol{z}_{k+1}|\boldsymbol{\delta T}_{k+1}) + \log p(\boldsymbol{\delta T}_{k+1}|\boldsymbol{z}_{1:k}) \\
&= \max_{\boldsymbol{\delta T}_{k+1}} \text{const} - \frac{1}{2}||\boldsymbol{z}_{k+1} - \boldsymbol{h}(\boldsymbol{T}_{k+1})||^2_{\boldsymbol{N}} - \frac{1}{2}||\boldsymbol{\delta T}_{k+1}||^2_{\boldsymbol{\Sigma}_{k+1|k}} \\
&= \min_{\boldsymbol{\delta T}_{k+1}} \frac{1}{2}||\boldsymbol{z}_{k+1} - \boldsymbol{h}(\boldsymbol{T}_{k+1})||^2_{\boldsymbol{N}} + \frac{1}{2}||\boldsymbol{\delta T}_{k+1}||^2_{\boldsymbol{\Sigma}_{k+1|k}} \\
&= \min_{\boldsymbol{\delta T}_{k+1}} \frac{1}{2} \begin{bmatrix} \boldsymbol{\delta T}_{k+1} \\ \boldsymbol{h}(\boldsymbol{T}_{k+1}) - \boldsymbol{z}_{k+1} \end{bmatrix}^\top \begin{bmatrix} \boldsymbol{\Sigma}_{k+1|k} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{N} \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{\delta T}_{k+1} \\ \boldsymbol{h}(\boldsymbol{T}_{k+1}) - \boldsymbol{z}_{k+1} \end{bmatrix} = c(\boldsymbol{\delta T}_{k+1})
\end{aligned} \tag{70}$$

Due to the non-linear measurement function, the optimal estimate is found iteratively with the Gauss-Newton method. Gauss-Newton uses the an approximation of the Hessian to approach a point where the gradient of the cost function $c(\boldsymbol{\delta T})$ vanishes. The gradient is related to transpose of the Jacobian. The Jacobian of the cost function is

$$\boldsymbol{J}^c_{\boldsymbol{\delta T}_{k+1}} = \begin{bmatrix} \boldsymbol{\delta T}_{k+1} \\ \boldsymbol{h}(\boldsymbol{T}_{k+1}) - \boldsymbol{z}_{k+1} \end{bmatrix}^\top \begin{bmatrix} \boldsymbol{\Sigma}_{k+1|k} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{N} \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{I}_9 \\ \boldsymbol{H} \end{bmatrix}. \tag{71}$$

The optimization scheme tries to find the $\boldsymbol{\delta T}$ such that the gradient difference $\boldsymbol{J}^{c,\top}_{\boldsymbol{\delta T}_{k+1},i+1} - \boldsymbol{J}^{c,\top}_{\boldsymbol{\delta T}_{k+1},i}$ vanishes, where i denotes the iteration number. A linear approximation of the Jacobian is $\boldsymbol{J}^{c,\top}_{\boldsymbol{\delta T}_{k+1},i+1} - \boldsymbol{J}^{c,\top}_{\boldsymbol{\delta T}_{k+1},i} \approx \mathcal{H}^c_{\boldsymbol{\delta T}_{k+1},i}\Delta\boldsymbol{\delta T}_{k+1}$, where $\mathcal{H}^c_{\boldsymbol{\delta T}_{k+1},i}$ is the Hessian of the cost function at the ith estimate of $\boldsymbol{\delta T}_{k+1}$. The optimal step $\Delta\boldsymbol{\delta T}_{k+1}$ is found by solving $\boldsymbol{J}^{c,\top}_{\boldsymbol{\delta T}_{k+1},i} = -\mathcal{H}^c_{\boldsymbol{\delta T}_{k+1},i}\Delta\boldsymbol{\delta T}_{k+1}$ for $\Delta\boldsymbol{\delta T}_{k+1}$, since the goal is for the gradient to vanish. In [27] the Hessian is shown to be

$$\begin{aligned}
\mathcal{H}^c_{\boldsymbol{\delta T}_{k+1}} = &\begin{bmatrix} \boldsymbol{I}_9 \\ \boldsymbol{H} \end{bmatrix}^\top \begin{bmatrix} \boldsymbol{\Sigma}_{k+1|k} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{N} \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{I}_9 \\ \boldsymbol{H} \end{bmatrix} \\
&+ \sum_j \Big(\begin{bmatrix} \boldsymbol{\Sigma}_{k+1|k} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{N} \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{\delta T}_{k+1} \\ \boldsymbol{h}(\boldsymbol{T}_{k+1}) - \boldsymbol{z}_{k+1} \end{bmatrix} \Big)_j \boldsymbol{H} \Big(\begin{bmatrix} \boldsymbol{\delta T}_{k+1} \\ \boldsymbol{h}(\boldsymbol{T}_{k+1}) - \boldsymbol{z}_{k+1} \end{bmatrix}_j \Big).
\end{aligned} \tag{72}$$

The j subscript denotes the jth element. The Gauss-Newton approximation is to neglect the second part of the Hessian. The approximation is exact if the measurement function is linear. Solving for the step size gives the scheme

$$\begin{aligned}
\Delta\boldsymbol{\delta T}_{k+1} &= -\boldsymbol{H}^{c,-1}_{\boldsymbol{\delta T}_{k+1},i}\boldsymbol{J}^{c,\top}_{\boldsymbol{\delta T}_{k+1},i} \\
&\approx (\begin{bmatrix}\boldsymbol{I}_9 \\ \boldsymbol{H}\end{bmatrix}^\top \begin{bmatrix}\boldsymbol{\Sigma}_{k+1|k} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{N}\end{bmatrix}^{-1} \begin{bmatrix}\boldsymbol{I}_9 \\ \boldsymbol{H}\end{bmatrix})^{-1} \\
&\qquad \begin{bmatrix}\boldsymbol{I}_9 \\ \boldsymbol{H}\end{bmatrix}^\top \begin{bmatrix}\boldsymbol{\Sigma}_{k+1|k} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{N}\end{bmatrix}^{-1} \begin{bmatrix}-\hat{\boldsymbol{\delta T}}_{k+1,i} \\ \boldsymbol{z}_{k+1} - \boldsymbol{h}(\hat{\boldsymbol{T}}_{k+1|k} \oplus \hat{\boldsymbol{\delta T}}_{k+1,i})\end{bmatrix} \\
&= (\boldsymbol{H}^\top\boldsymbol{N}^{-1}\boldsymbol{H} + \boldsymbol{\Sigma}^{-1}_{k+1|k})^{-1}\boldsymbol{H}^\top\boldsymbol{N}^{-1}(\boldsymbol{z}_{k+1} - \boldsymbol{h}(\hat{\boldsymbol{T}}_{k+1|k} \oplus \hat{\boldsymbol{\delta T}}_{k+1,i})) \\
&\qquad - (\boldsymbol{H}^\top\boldsymbol{N}^{-1}\boldsymbol{H} + \boldsymbol{\Sigma}^{-1}_{k+1|k})^{-1}\boldsymbol{\Sigma}^{-1}_{k+1|k}\hat{\boldsymbol{\delta T}}_{k+1,i} \\
&= \boldsymbol{K}(\boldsymbol{z}_{k+1} - \boldsymbol{h}(\hat{\boldsymbol{T}}_{k+1|k} \oplus \hat{\boldsymbol{\delta T}}_{k+1,i}) - \boldsymbol{H}\hat{\boldsymbol{\delta T}}_{k+1,i}).
\end{aligned}$$

where $\boldsymbol{K} = \boldsymbol{\Sigma}_{k+1|k}\boldsymbol{H}^\top\boldsymbol{\Lambda}^{-1}$, and $\boldsymbol{\Lambda} = \boldsymbol{H}\boldsymbol{\Sigma}_{k+1|k}\boldsymbol{H}^\top + \boldsymbol{N}$. These are the usual Kalman gain matrix and innovation covariance of the Kalman filter. As the measurement matrix is dependent on the linearization point, which changes with each iteration, these matrices vary with the iterations. Subscripts to mark this should be included. Adding the current estimate gives an expression similar to the EKF update step.

$$\hat{\boldsymbol{\delta T}}_{k+1,i+1} = \hat{\boldsymbol{\delta T}}_{k+1,i} + \boldsymbol{K}_{k,i}(\boldsymbol{z}_{k+1} - \boldsymbol{h}(\hat{\boldsymbol{T}}_{k+1|k} \oplus \hat{\boldsymbol{\delta T}}_{k+1,i}) - \boldsymbol{H}_{k,i}\hat{\boldsymbol{\delta T}}_{k+1,i}). \tag{73}$$

The difference is the iterative nature of this formulation. Inserting the initial estimate $\hat{\boldsymbol{\delta T}}_{k+1,i} = \hat{\boldsymbol{\delta T}}_{k+1|k} = \boldsymbol{0}$ removes the last term revealing the usual EKF update equation. The EKF is therefore a single iteration of a Gauss-Newton optimization scheme.

Using either of the two GNSS measurement functions is possible. Using the one with the constant Jacobian seems like a smart choice, but then the noise changes with the linearization point, making the Kalman gain still dependent on the linearization point. If the GNSS noise is on the form $\sigma_n^2\boldsymbol{I}$, it is rotation symmetric which makes it independent of the linearization point. This indicates that the update step should converge in few iterations for a GNSS measurement, and a single EKF step is probably sufficient.

The update cycle of the filter estimates the error $\boldsymbol{\delta T}$, meaning after a run of the update step our distribution of the error is probably not zero-mean. Since there is no point in propagating this error, the error estimate should be injected into the state estimate. After the update step the distribution has the form

$$\boldsymbol{T}_{k+1} = \hat{\boldsymbol{T}}_{k+1|k}\operatorname{Exp}(\boldsymbol{\delta T}^-_{k+1}), \qquad \boldsymbol{\delta T}^-_{k+1} \sim \mathcal{N}(\hat{\boldsymbol{\delta T}}_{k+1}, \boldsymbol{\Sigma}^-_{k+1}). \tag{74}$$

The term $\hat{\boldsymbol{\delta T}}_{k+1}$ should be injected into $\hat{\boldsymbol{T}}_{k+1|k}$. To achieve this, another first order expansion is the solution. Rewriting $\boldsymbol{\delta T}^-_{k+1} = \hat{\boldsymbol{\delta T}}_{k+1} + \boldsymbol{\zeta}_{k+1}$, where $\boldsymbol{\zeta}_{k+1}$ is the zero-mean random variable with covariance $\boldsymbol{\Sigma}^-_{k+1}$, using Equation 20 gives

$$\begin{aligned}
\boldsymbol{T}_{k+1} &= \hat{\boldsymbol{T}}_{k+1|k}\operatorname{Exp}(\boldsymbol{\delta T}^-_{k+1}) \\
&\approx \underbrace{\hat{\boldsymbol{T}}_{k+1|k}\operatorname{Exp}(\hat{\boldsymbol{\delta T}}_{k+1})}_{\hat{\boldsymbol{T}}_{k+1}}\operatorname{Exp}(\underbrace{\mathcal{J}_r(\hat{\boldsymbol{\delta T}}_{k+1})\boldsymbol{\zeta}_{k+1}}_{\boldsymbol{\delta T}_{k+1}}), \qquad \boldsymbol{\zeta}_{k+1} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}^-_{k+1}) \\
&= \hat{\boldsymbol{T}}_{k+1}\operatorname{Exp}(\boldsymbol{\delta T}_{k+1}), \qquad \boldsymbol{\delta T}_{k+1} \sim \mathcal{N}(\boldsymbol{0}, \underbrace{\mathcal{J}_r(\hat{\boldsymbol{\delta T}}_{k+1})\boldsymbol{\Sigma}^-_{k+1}\mathcal{J}_r(\hat{\boldsymbol{\delta T}}_{k+1})^\top}_{\boldsymbol{\Sigma}_{k+1}})
\end{aligned} \tag{75}$$

Equation 75 has the expected form of Equation 48. There are two important steps in the derivation above. There is the injection of the error estimate, and the reset of the error covariance. The reset is a consequence of the error coordinates changing from being in the tangent at $\hat{\boldsymbol{T}}_{k+1|k}$ to the tangent at $\hat{\boldsymbol{T}}_{k+1}$.[8] The entire update step is summarized in Algorithm 3.

There are several linearizations and approximations made in the derivation of the filter. [11] proves the inertial filter to be an asymptotically stable observer for the noisy inertial navigation

Algorithm 3 Update step

procedure UPDATE($\hat{\boldsymbol{T}}_{k+1|k}, \boldsymbol{\Sigma}_{k+1|k}, \boldsymbol{z}_{k+1}, \boldsymbol{N}, \boldsymbol{h}$)
  $\boldsymbol{H}_{k+1} \leftarrow \frac{\partial}{\partial \boldsymbol{T}} \boldsymbol{h}(\boldsymbol{T}) \Big|_{\boldsymbol{T}=\hat{\boldsymbol{T}}_{k+1|k}}$ ▷ The measurement Jacobian
  $\boldsymbol{\lambda}_{k+1} \leftarrow \boldsymbol{z}_{k+1} \ominus \boldsymbol{h}(\hat{\boldsymbol{T}}_{k+1|k})$ ▷ The innovation
  $\boldsymbol{\Lambda}_{k+1} \leftarrow \boldsymbol{H}_{k+1} \boldsymbol{\Sigma}_{k+1|k} \boldsymbol{H}_{k+1}^\top + \boldsymbol{N}$ ▷ The innovation covariance
  $\boldsymbol{K}_{k+1} \leftarrow \boldsymbol{\Sigma}_{k+1|k} \boldsymbol{H}_{k+1}^\top \boldsymbol{\Lambda}_{k+1}^{-1}$ ▷ The Kalman gain
  $\hat{\boldsymbol{\delta T}}_{k+1} \leftarrow \boldsymbol{K}_{k+1} \boldsymbol{\lambda}_{k+1}$ ▷ The error estimate
  $\boldsymbol{\Sigma}_{k+1}^- \leftarrow (\boldsymbol{I} - \boldsymbol{K}_{k+1} \boldsymbol{H}_{k+1}) \boldsymbol{\Sigma}_{k+1|k}$ ▷ The error covariance
  $\hat{\boldsymbol{T}}_{k+1} \leftarrow \hat{\boldsymbol{T}}_{k+1|k} \oplus \hat{\boldsymbol{\delta T}}_{k+1}$ ▷ The injection
  $\boldsymbol{\Sigma}_{k+1} \leftarrow \mathcal{J}_r(\hat{\boldsymbol{\delta T}}_{k+1}) \boldsymbol{\Sigma}_{k+1}^- \mathcal{J}_r(\hat{\boldsymbol{\delta T}}_{k+1})^\top$ ▷ The covariance reset
  return $\hat{\boldsymbol{T}}_{k+1}$, $\boldsymbol{\Sigma}_{k+1}$
end procedure

system if three or more non-colinear known points are observed. Using GNSS, only a single point is observed per update. This means non-linear movement between three GNSS updates should render the observer stable.

3.10 $\mathrm{SE}_2(3)$ and $\mathrm{SO}(3)\times\mathbb{R}^3 \times \mathbb{R}^3$

Kalman filters parameterizing the attitude error in fewer dimensions than the attitude state is not a new concept, dating back to at least 1969.[28] The approach is known as the Multiplicative Extended Kalman Filter (MEKF) or the Error State Kalman Filter (ESKF).[8] In [8] Sola summarizes the ESKF. Sola uses a quaternion representation of the orientation. This makes no difference as the Exponential coordinates of quaternions and rotation matrices are the same axis-angle representation. The filter has much in common with the filter derived here. In fact, from a Lie viewpoint the ESKF can viewed as a filter on the group $\mathrm{SO}(3)\times\mathbb{R}^3 \times \mathbb{R}^3$. The group is similar to $\mathrm{SE}_2(3)$, but the subtle differences are important. $\mathrm{SO}(3)\times\mathbb{R}^3 \times \mathbb{R}^3$ treats rotation, velocity, and position as separate quantities. Figure 8 illustrates the difference. Interpolating the same state on $\mathrm{SO}(2)\times\mathbb{R}^2$ and $\mathrm{SE}(2)$, gives two very different paths. The groups are the two-dimensional equivalents of the ones discussed here without the velocity state, which is convenient for illustrations. The curved path belongs to $\mathrm{SE}(2)$. Here, the change of position is constant locally, so it rotates with the rotation of the frame. This is not the case for $\mathrm{SO}(2)\times\mathbb{R}^2$. Since rotation and position are treated separately, the change of position is constant globally.

$\mathrm{SO}(3)\times\mathbb{R}^3 \times \mathbb{R}^3$ shares the same composition operation as $\mathrm{SE}_2(3)$. The Exponential and Logarithmic maps are different. The Exponential map does not include the $\mathrm{SO}(3)$ Jacobian, making the Exponential coordinates the same as the state for the vectorial velocity and position. There is no connection between the orientation and vectorial states in the Exponential map, which makes all the difference to $\mathrm{SE}_2(3)$. With no connection between the states, the Jacobian is only dependent on the rotational state, meaning marginalizing the probability distribution over the rotational state yields Gaussian distributions over position and velocity. This is not the case with the filter in this work, as the Jacobian is dependent on all the states.

The ESKF uses linearizations of the error state to propagate the error covariance. This is not the case for the filter in this work. The tools provided by Lie theory allow for an almost closed form solution to the propagation step, with close to linear error dynamics. This preserves Gaussianity on the error coordinates without linearizations and close to no approximations. Excluding the fourth-order contributions of Equation 59 is a linearization equivalent to the one done in the ESKF.

Since the $\mathrm{SE}_2(3)$ filter in this work uses linearizations in the update step, this step is similar to the ESKF. As the reset and injection is dependent on the Jacobian, they differ slightly. The right

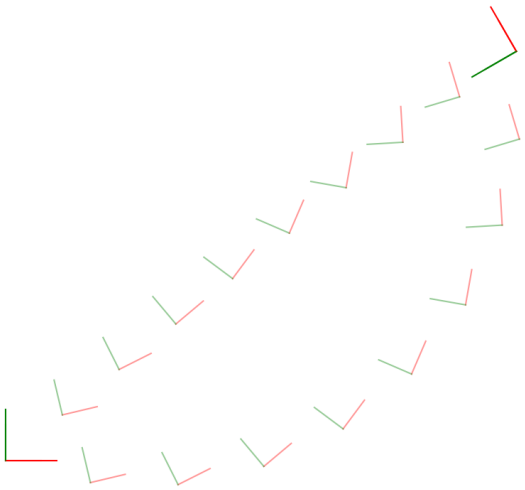


Figure 8: Interpolating the same transform as described by $SO(2) \times \mathbb{R}^2$ and $SE(2)$.

Jacobian of SO(3)$\times\mathbb{R}^3 \times \mathbb{R}^3$ is given by

$$\mathcal{J}_r(\boldsymbol{\tau}) = \begin{bmatrix} \mathcal{J}_r(\boldsymbol{\phi}) & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \boldsymbol{R}^\top & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \boldsymbol{R}^\top \end{bmatrix} \approx \begin{bmatrix} \boldsymbol{I}_3 - \frac{1}{2}\boldsymbol{\phi}_\times & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \boldsymbol{R}^\top & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \boldsymbol{R}^\top \end{bmatrix}, \tag{76}$$

where the right Jacobian of SO(3) is approximated to the first order, and $\boldsymbol{R} = \operatorname{Exp}\boldsymbol{\phi}$. $\mathcal{J}_r(\boldsymbol{\phi}) = \boldsymbol{I}_3 - \frac{1-\cos\phi}{\phi^2}\boldsymbol{\phi}_\times + \frac{\phi-\sin\phi}{\phi^3}\boldsymbol{\phi}_\times^2 \approx \boldsymbol{I}_3 - \frac{1}{2}\boldsymbol{\phi}_\times$ where the second order term has been dropped and the limit $\lim_{\phi\to0}\frac{1-\cos\phi}{\phi^2} = \lim_{\phi\to0}\frac{\sin\phi}{2\phi} = \frac{1}{2}$ is used. This reset matrix matches the matrix in Sola's paper closely, (see eq. 288). The difference is the entry of the rotational matrix, the different ordering of the error state and no bias and gravity estimation. The rotational matrix enters as the vectorial states are still in local coordinates. In the ESKF the vectorial uncertainty is in the global frame. Compared to the Jacobian of $\mathrm{SE}_2(3)$, the disconnect between the rotational state and the vectorial states is obvious.

The SO(3)$\times\mathbb{R}^3 \times \mathbb{R}^3$ can equivalently be defined as the triple $\{\boldsymbol{R}, \boldsymbol{v}, \boldsymbol{p}\}$ with the composition operation $\{\boldsymbol{R}_1, \boldsymbol{v}_1, \boldsymbol{p}_1\} \circ \{\boldsymbol{R}_2, \boldsymbol{v}_2, \boldsymbol{p}_2\} = \{\boldsymbol{R}_1\boldsymbol{R}_2, \boldsymbol{v}_1 + \boldsymbol{v}_2, \boldsymbol{p}_1 + \boldsymbol{p}_2\}$. This defines the error coordinates of the vectorial states in the global frame, as done with the ESKF. This replaces the rotation matrices with identity matrices in the Jacobian above, yielding the reset matrix of Sola's paper exactly. However, this does not fit with the matrix Lie group approach. For code compatibility reasons, the other approach is taken.

4 Tracking theory

With the inertial navigation filter derived, the focus is moved towards target tracking. The body tracking other objects will in this context be called a platform. The inertial navigation provides a probability distribution over the state of the platform. The tracker should give a probability distribution over the state of the targets. To do this, a motion model and a measurement model of the targets are needed.

Initialization and data association are problems of target tracking that are not covered in this thesis. Instead, the targets are initialized with a known state distribution, and it is assumed that it is known what target a measurement is of. In practice, this can be achieved by some kind of recognition algorithm on, for example, a camera feed. [26] covers the topic of data association and initialization on a tracking algorithm. The techniques can be readily included in the methods developed here, but the focus is on filter consistency in this work.

4.1 CV model

The targets are assumed to move according to a Continuous Velocity (CV) model. The state to be estimated is then position and velocity $\boldsymbol{x} = [\boldsymbol{p}, \boldsymbol{v}]^\top \in \mathbb{R}^6$. There are two natural choices for how the parameterize the state; it can either be in the world frame, or in the body frame of the platform. Both will be explored in this section. A superscript will be used to denote which parameterization is used, $(\cdot)^w$ for world and $(\cdot)^b$ for body. The superscripts also serve as a method to disambiguate the target position and velocity from the platform position and velocity, as the same letters are used for both. Since the orientation of targets is not of interest, the trivial Lie Group $(\mathbb{R}^6, +)$ describes the manifold on which they evolve.

The motion model at the core of the CV assumption takes different forms depending on the world or body parameterization. A CV model assumes the targets to move with a reasonably constant velocity. The change in velocity is modelled as a white noise random process. In the world frame this gives the motion model $\boldsymbol{x}^w_{k+1} = \boldsymbol{F}^{\text{CV}}\boldsymbol{x}^w_k + \boldsymbol{\gamma}, \boldsymbol{\gamma} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Gamma})$, where $\boldsymbol{F}^{\text{CV}} = \left[\begin{smallmatrix} \boldsymbol{I}_3 & \delta t \boldsymbol{I}_3 \\ \boldsymbol{0}_3 & \boldsymbol{I}_3 \end{smallmatrix}\right]$. This is equivalently described by the probability distribution

$$p(\boldsymbol{x}^w_{k+1}|\boldsymbol{x}^w_k) = \mathcal{N}(\boldsymbol{x}^w_{k+1}; \boldsymbol{F}^{\text{CV}}\boldsymbol{x}^w_k, \boldsymbol{\Gamma}). \tag{77}$$

As the CV model is inherently a continuous time model with $\dot{\boldsymbol{x}} = \left[\begin{smallmatrix} \boldsymbol{0} & \boldsymbol{I}_3 \\ \boldsymbol{0} & \boldsymbol{0} \end{smallmatrix}\right]\boldsymbol{x} + \left[\begin{smallmatrix} \boldsymbol{0} \\ \boldsymbol{I}_3 \end{smallmatrix}\right]\boldsymbol{\kappa}$, the discrete state transition and covariance matrices should be calculated from the solution to the continuous time system. Assuming the zero-mean random variable $\boldsymbol{\kappa}$ to have a covariance matrix on the form $\boldsymbol{I}_3\sigma_a^2$, the discrete covariance is $\boldsymbol{\Gamma} = \left[\begin{smallmatrix} \delta t^3/3\boldsymbol{I}_3 & \delta t^2/2\boldsymbol{I}_3 \\ \delta t^2/2\boldsymbol{I}_3 & \delta t \boldsymbol{I}_3 \end{smallmatrix}\right]\delta_a^2$. The state transition matrix is $\boldsymbol{F}^{\text{CV}}$.[14]

When the target is parameterized in the body frame, the motion model must be compensated for platform movement. To apply the CV model, the body position and velocity must be converted to world coordinates, and then converted back after. It is convenient to express the frame conversion using the action of the platform state, and the action of Equation 33 is used for this. Using this notation the CV motion model in the body frame can be written as

$$\boldsymbol{x}^b_{k+1} = \boldsymbol{T}^{-1}_{k+1} : (\overbrace{\boldsymbol{F}^{\text{CV}}(\underbrace{\boldsymbol{T}_k : \boldsymbol{x}^b_k}_{\boldsymbol{x}^w_k}) + \boldsymbol{\gamma}}^{\boldsymbol{x}^w_{k+1}}), \qquad \boldsymbol{\gamma} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Gamma}) \tag{78}$$

Finding an exact Gaussian expression for this motion model is not possible, due to the non-linear nature of the action, but an approximation can be found. Applying Equation 19 to Equation 78 gives a linear propagation model. Since the action is affine in $\boldsymbol{x}$ when $\boldsymbol{T}$ is given, there is no need to linearize this variable. It is important to note that the Jacobians are linearized around a constant linearization point $\hat{\boldsymbol{x}}$ also for this variable so that the Jacobians are independent of the actual $\boldsymbol{x}$. Independence is necessary for the linear combination of Gaussians to be Gaussian.

$$\boldsymbol{x}^b_{k+1} \approx \hat{\boldsymbol{T}}^{-1}_{k+1} : (\boldsymbol{F}^{\text{CV}}(\hat{\boldsymbol{T}}_k : \boldsymbol{x}^b_k)) + \boldsymbol{J}^{\boldsymbol{x}^b_{k+1}}_{\boldsymbol{T}^{-1}_{k+1}} \boldsymbol{J}^{\boldsymbol{T}^{-1}_{k+1}}_{\boldsymbol{T}_{k+1}} \boldsymbol{\delta T}_{k+1} + \boldsymbol{J}^{\boldsymbol{x}^b_{k+1}}_{\boldsymbol{x}^w_{k+1}} \boldsymbol{\gamma} + \boldsymbol{J}^{\boldsymbol{x}^b_{k+1}}_{\boldsymbol{x}^w_{k+1}} \boldsymbol{F}^{\text{CV}} \boldsymbol{J}^{\boldsymbol{x}^w_k}_{\boldsymbol{T}_k} \boldsymbol{\delta T}_k \tag{79}$$

Before Equation 79 can be made Gaussian, the dependency between $\boldsymbol{\delta T}_k$ and $\boldsymbol{\delta T}_{k+1}$ must be addressed. Inserting the approximation $\boldsymbol{\delta T}_{k+1} \approx \mathrm{Ad}_{\hat{\boldsymbol{\Delta}}_k^{-1}} \boldsymbol{F}_{\delta t} \boldsymbol{\delta T}_k + \boldsymbol{\vartheta}$ from Equation 60 yields

$$\begin{aligned}
\boldsymbol{x}_{k+1}^b \approx \hat{\boldsymbol{T}}_{k+1}^{-1} : (\boldsymbol{F}^{\mathrm{CV}}(\hat{\boldsymbol{T}}_k : \boldsymbol{x}_k^b)) &+ \boldsymbol{J}^{\boldsymbol{x}_{k+1}^b}_{\boldsymbol{T}_{k+1}^{-1}} \boldsymbol{J}^{\boldsymbol{T}_{k+1}^{-1}}_{\boldsymbol{T}_{k+1}} (\mathrm{Ad}_{\hat{\boldsymbol{\Delta}}_k^{-1}} \boldsymbol{F}_{\delta t} \boldsymbol{\delta T}_k + \boldsymbol{\vartheta}) \\
&+ \boldsymbol{J}^{\boldsymbol{x}_{k+1}^b}_{\boldsymbol{x}_{k+1}^w} \boldsymbol{\gamma} + \boldsymbol{J}^{\boldsymbol{x}_{k+1}^b}_{\boldsymbol{x}_{k+1}^w} \boldsymbol{F}^{\mathrm{CV}} \boldsymbol{J}^{\boldsymbol{x}_k^w}_{\boldsymbol{T}_k} \boldsymbol{\delta T}_k \\
= \hat{\boldsymbol{T}}_{k+1}^{-1} : (\boldsymbol{F}^{\mathrm{CV}}(\hat{\boldsymbol{T}}_k : \boldsymbol{x}_k^b)) &+ (\boldsymbol{J}^{\boldsymbol{x}_{k+1}^b}_{\boldsymbol{T}_{k+1}^{-1}} \boldsymbol{J}^{\boldsymbol{T}_{k+1}^{-1}}_{\boldsymbol{T}_{k+1}} \mathrm{Ad}_{\hat{\boldsymbol{\Delta}}_k^{-1}} \boldsymbol{F}_{\delta t} + \boldsymbol{J}^{\boldsymbol{x}_{k+1}^b}_{\boldsymbol{x}_{k+1}^w} \boldsymbol{F}^{\mathrm{CV}} \boldsymbol{J}^{\boldsymbol{x}_k^w}_{\boldsymbol{T}_k}) \boldsymbol{\delta T}_k \\
&+ \boldsymbol{J}^{\boldsymbol{x}_{k+1}^b}_{\boldsymbol{T}_{k+1}^{-1}} \boldsymbol{J}^{\boldsymbol{T}_{k+1}^{-1}}_{\boldsymbol{T}_{k+1}} \boldsymbol{\vartheta} + \boldsymbol{J}^{\boldsymbol{x}_{k+1}^b}_{\boldsymbol{x}_{k+1}^w} \boldsymbol{\gamma}.
\end{aligned} \tag{80}$$

This gives the motion model

$$p(\boldsymbol{x}_{k+1}^b | \boldsymbol{x}_k^b) \approx \mathcal{N}(\boldsymbol{x}_{k+1}^b; \hat{\boldsymbol{T}}_{k+1}^{-1} : (\boldsymbol{F}^{\mathrm{CV}}(\hat{\boldsymbol{T}}_k : \boldsymbol{x}_k^b)), \boldsymbol{J}^1 \boldsymbol{\Sigma}_k \boldsymbol{J}^{1\top} + \boldsymbol{J}^2 \boldsymbol{Q}_k \boldsymbol{J}^{2\top} + \boldsymbol{J}^3 \boldsymbol{\Gamma} \boldsymbol{J}^{3\top}), \tag{81}$$

where $\boldsymbol{J}^{1,2,3}$ are the three expressions of Jacobians preceding the corresponding random variables on the last line of Equation 80. The needed Jacobians are

$$\begin{aligned}
\boldsymbol{J}^{\boldsymbol{x}^b}_{\boldsymbol{T}^{-1}} &= \begin{bmatrix} -\hat{\boldsymbol{R}}^\top \hat{\boldsymbol{p}}^w_\times & \boldsymbol{0}_3 & \hat{\boldsymbol{R}}^\top \\ -\hat{\boldsymbol{R}}^\top \hat{\boldsymbol{v}}^w_\times & \hat{\boldsymbol{R}}^\top & \boldsymbol{0}_3 \end{bmatrix}, \\
\boldsymbol{J}^{\boldsymbol{T}^{-1}}_{\boldsymbol{T}} &= -\mathrm{Ad}_{\boldsymbol{T}}, \\
\boldsymbol{J}^{\boldsymbol{x}^b}_{\boldsymbol{x}^w} &= \begin{bmatrix} \hat{\boldsymbol{R}}^\top & \boldsymbol{0}_3 \\ \boldsymbol{0}_3 & \hat{\boldsymbol{R}}^\top \end{bmatrix}, \\
\boldsymbol{J}^{\boldsymbol{x}^w}_{\boldsymbol{T}} &= \begin{bmatrix} -\hat{\boldsymbol{R}} \hat{\boldsymbol{p}}^b_\times & \boldsymbol{0}_3 & \hat{\boldsymbol{R}} \\ -\hat{\boldsymbol{R}} \hat{\boldsymbol{v}}^b_\times & \hat{\boldsymbol{R}} & \boldsymbol{0}_3 \end{bmatrix}.
\end{aligned} \tag{82}$$

The expectation in Equation 81 is affine in $\boldsymbol{x}_k^b$, which makes the motion model the Gaussian affine form required for propagation.

$$\hat{\boldsymbol{T}}_{k+1}^{-1} : (\boldsymbol{F}^{\mathrm{CV}}(\hat{\boldsymbol{T}}_k : \boldsymbol{x}_k^b)) = \begin{bmatrix} \hat{\boldsymbol{R}}_{k+1}^\top \hat{\boldsymbol{R}}_k & \delta t \hat{\boldsymbol{R}}_{k+1}^\top \hat{\boldsymbol{R}}_k \\ \boldsymbol{0} & \hat{\boldsymbol{R}}_{k+1}^\top \hat{\boldsymbol{R}}_k \end{bmatrix} \boldsymbol{x}_k^b + \begin{bmatrix} \hat{\boldsymbol{R}}_{k+1}^\top (\hat{\boldsymbol{p}}_k - \hat{\boldsymbol{p}}_{k+1} + \delta t \hat{\boldsymbol{v}}_k) \\ \hat{\boldsymbol{R}}_{k+1}^\top (\hat{\boldsymbol{v}}_k - \hat{\boldsymbol{v}}_{k+1}) \end{bmatrix} \tag{83}$$

Notice that the first matrix degenerates to the CV model matrix $\boldsymbol{F}^{\mathrm{CV}}$ when the platform does not rotate between time steps, which agrees with reason. The CV body propagation covariance is a function of the CV world covariance, the IMU noise, the platform uncertainty, the IMU measurements, and the linearization points. Writing the expectation in the affine form, a closed form solution can be found to the propagation problem.

4.2 Target measurement model

The platform measures the position of targets. The measurements are in the body frame of the platform. Target measurements are denoted by $\boldsymbol{y}$ and disturbed by zero-mean additive noise $\boldsymbol{m}$.

$$\boldsymbol{y}_k = \boldsymbol{f}_{\boldsymbol{y}}(\boldsymbol{x}_k, \boldsymbol{T}_k) + \boldsymbol{m} = \boldsymbol{p}_k^b + \boldsymbol{m}, \qquad \boldsymbol{m} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{M})$$

As with the motion model, the measurement model depends on the parameterization of the target. Since the platform measures the position of the targets in the body frame, the body formulation is more convenient here. The measurement function is linear, with the measurement matrix $\boldsymbol{H}^m = [\boldsymbol{I}_3, \boldsymbol{0}_3]$.

$$\begin{aligned}
\boldsymbol{y}_k &= \boldsymbol{H}^m \boldsymbol{x}_k^b + \boldsymbol{m} \\
p(\boldsymbol{y}_k | \boldsymbol{x}_k^b) &= \mathcal{N}(\boldsymbol{y}_k; \boldsymbol{H}^m \boldsymbol{x}_k^b, \boldsymbol{M})
\end{aligned} \tag{84}$$

For the world parameterization, the action of the platform is used. This is the regular action of Equation 32. Similarly to the motion model of the CV body, the world measurement model needs

to be approximated with a linearization to take a Gaussian form.

$$
\begin{aligned}
\boldsymbol{y}_k &= \boldsymbol{T}_k^{-1} \cdot \boldsymbol{H}^m \boldsymbol{x}_k^w + \boldsymbol{m} \\
&\approx \hat{\boldsymbol{T}}_k^{-1} \cdot \hat{\boldsymbol{p}}_k^w + [\hat{\boldsymbol{R}}_k^\top, \boldsymbol{0}_3](\boldsymbol{x}_k^w - \hat{\boldsymbol{x}}_k^w) + [\hat{\boldsymbol{R}}_k^\top \hat{\boldsymbol{p}}_{\times,k}^w, \boldsymbol{0}_3, -\hat{\boldsymbol{R}}_k^\top] \operatorname{Ad}_{\hat{\boldsymbol{T}}_k} \boldsymbol{\delta T}_k + \boldsymbol{m} \\
&= \hat{\boldsymbol{T}}_k^{-1} \cdot \boldsymbol{p}_k^w + [(\hat{\boldsymbol{R}}_k^\top (\hat{\boldsymbol{p}}_k^w - \hat{\boldsymbol{p}}_k))_\times, \boldsymbol{0}_3, -\boldsymbol{I}_3] \boldsymbol{\delta T}_k + \boldsymbol{m} \\
&= \hat{\boldsymbol{T}}_k^{-1} \cdot \boldsymbol{p}_k^w + \underbrace{[\hat{\boldsymbol{p}}_{k,\times}^b, \boldsymbol{0}_3, -\boldsymbol{I}_3]}_{:= \boldsymbol{J}_{\boldsymbol{T}}^{\boldsymbol{T}^{-1} \cdot \boldsymbol{x}}} \boldsymbol{\delta T}_k + \boldsymbol{m} \\
p(\boldsymbol{y}_k | \boldsymbol{x}_k^w) &= \int p(\boldsymbol{y}_k | \boldsymbol{x}_k^w, \boldsymbol{T}_k) p(\boldsymbol{T}_k) d\boldsymbol{T}_k \\
&= \int \mathcal{N}(\boldsymbol{y}_k; \boldsymbol{T}_k^{-1} \cdot \boldsymbol{H}^m \boldsymbol{x}_k^w, \boldsymbol{M}) \mathcal{N}_{\mathcal{G}}(\boldsymbol{T}_k; \hat{\boldsymbol{T}}_k, \boldsymbol{\Sigma}_k) d\boldsymbol{T}_k \\
&\approx \int \mathcal{N}(\boldsymbol{y}_k; \hat{\boldsymbol{T}}_k^{-1} \cdot \boldsymbol{p}_k^w + \boldsymbol{J}_{\boldsymbol{T}}^{\boldsymbol{T}^{-1} \cdot \boldsymbol{x}} \boldsymbol{\delta T}_k, \boldsymbol{M}) \mathcal{N}(\boldsymbol{\delta T}_k; \boldsymbol{0}, \boldsymbol{\Sigma}_k) d\boldsymbol{\delta T}_k \\
&= \mathcal{N}(\boldsymbol{y}_k; \hat{\boldsymbol{T}}_k^{-1} \cdot \boldsymbol{p}_k^w, \boldsymbol{J}_{\boldsymbol{T}}^{\boldsymbol{T}^{-1} \cdot \boldsymbol{x}} \boldsymbol{\Sigma}_k \boldsymbol{J}_{\boldsymbol{T}}^{\boldsymbol{T}^{-1} \cdot \boldsymbol{x}, \top} + \boldsymbol{M})
\end{aligned} \tag{85}
$$

This linearized measurement model is now on the Gaussian affine form needed. It is practical to have the entire state $\boldsymbol{x}^w$ in the expectation, not just the position.

$$
p(\boldsymbol{y}_k | \boldsymbol{x}_k^w) = \mathcal{N}(\boldsymbol{y}_k; \hat{\boldsymbol{R}}_k^\top \boldsymbol{H}^m \boldsymbol{x}_k^w - \hat{\boldsymbol{R}}_k^\top \hat{\boldsymbol{p}}_k, \boldsymbol{J}_{\boldsymbol{T}}^{\boldsymbol{T}^{-1} \cdot \boldsymbol{x}} \boldsymbol{\Sigma}_k \boldsymbol{J}_{\boldsymbol{T}}^{\boldsymbol{T}^{-1} \cdot \boldsymbol{x}, \top} + \boldsymbol{M}) \tag{86}
$$

Measurements are typically of the range-bearing type in tracking applications.[6] This is the case, for example, for radar tracking. Range-bearing measurements are, by definition, in polar or spherical coordinates. As the rest of the filter uses Cartesian coordinates, the measurements must be converted. The conversion makes the Cartesian noise dependent on the measurements, as measurements farther away are less precise. Despite this, the target measurements are assumed to be Cartesian with noise independent of the measurement for simplicity in this work.

5 Trackers

The simplest tracking filters on a moving platform neglect any platform uncertainties entirely and treat platform state estimates as true values. This affects performance, as a correct understanding of uncertainties is key to successful estimation.[6] Several tracking algorithms can be conceived, with varying levels of complexity. In [6] seven tracking filters in one dimension are introduced. In this thesis, the correlation-free filters are developed for 3 dimensions and Exponential Gaussian distributions. To the best of the author's knowledge, the correlation-free filters, motion models, and measurement models depending on an Exponential Gaussian distribution over the platform state have not been derived before.

5.1 Correlation-free world filter

The simplest tracking filters that do incorporate the platform uncertainty ignore any correlations between the target and platform state. The correlation-free world filter tracks the target in the world frame, $\boldsymbol{x}_k^w = [\boldsymbol{p}_k^w, \boldsymbol{v}_k^w]^\top$. Ignoring the correlations between the target and platform is equivalent to treating the state distributions of the targets and platform as independent, $p(\boldsymbol{x}, \boldsymbol{T}) = p(\boldsymbol{x})p(\boldsymbol{T})$. Here, the conditioning on all previous measurements is implicit for convenience. This is the case for all of this section. Since the targets and platform are assumed independent, the targets are unaffected by platform update, $p(\boldsymbol{x}^w, \boldsymbol{T}|\boldsymbol{z}) = p(\boldsymbol{z}|\boldsymbol{x}^w, \boldsymbol{T})p(\boldsymbol{x}^w, \boldsymbol{T})/p(\boldsymbol{z}) = p(\boldsymbol{z}|\boldsymbol{T})p(\boldsymbol{x}^w)p(\boldsymbol{T})/p(\boldsymbol{z}) = p(\boldsymbol{T}|\boldsymbol{z})p(\boldsymbol{x}^w)$. The platform state will not be affected by target update, even though these measurements carry information about the joint state of the target and platform. This is ignored to enforce independence.

Similar to the inertial navigation, there are two steps to the tracking filter. In the propagation step the density $p(\boldsymbol{x}_k^w, \boldsymbol{T}_k|\boldsymbol{y}_{1:k-1}, \boldsymbol{z}_{1:k})$ is evaluated. The propagation step is simple. The object is propagated using the CV model, and the platform with the inertial navigation filter.

In the update step, we want to find the density $p(\boldsymbol{x}_k^w|\boldsymbol{y}_{1:k}) \propto \int p(\boldsymbol{y}_k|\boldsymbol{x}_k^w, \boldsymbol{T}_k)p(\boldsymbol{x}_k^w, \boldsymbol{T}_k|\boldsymbol{y}_{1:k-1})d\boldsymbol{T}_k$.

$$
\begin{aligned}
p(\boldsymbol{x}_k^w|\boldsymbol{y}_{1:k}) &\propto \int p(\boldsymbol{y}_k|\boldsymbol{x}_k^w, \boldsymbol{T}_k)p(\boldsymbol{x}_k^w, \boldsymbol{T}_k|\boldsymbol{y}_{1:k-1})d\boldsymbol{T}_k \\
&= \underbrace{\int p(\boldsymbol{y}_k|\boldsymbol{x}_k^w, \boldsymbol{T}_k)p(\boldsymbol{T}_k)d\boldsymbol{T}_k}_{p(\boldsymbol{y}_k|\boldsymbol{x}_k^w)} p(\boldsymbol{x}_k^w|\boldsymbol{y}_{1:k-1}) \\
&\approx \mathcal{N}(\boldsymbol{y}_k; \hat{\boldsymbol{R}}_k^\top \boldsymbol{H}^m \boldsymbol{x}_k^w - \hat{\boldsymbol{R}}_k^\top \hat{\boldsymbol{p}}_k, \boldsymbol{J}_{\boldsymbol{T}}^{\boldsymbol{T}^{-1}\cdot\boldsymbol{x}} \boldsymbol{\Sigma}_k \boldsymbol{J}_{\boldsymbol{T}}^{\boldsymbol{T}^{-1}\cdot\boldsymbol{x},\top} + \boldsymbol{M})\mathcal{N}(\boldsymbol{x}_k^w; \hat{\boldsymbol{x}}_{k|k-1}^w, \boldsymbol{P}_{k|k-1}) \\
&= \mathcal{N}(\begin{bmatrix} \boldsymbol{I}_3 & -\hat{\boldsymbol{R}}_k^\top \boldsymbol{H}^m \\ \boldsymbol{0}_3 & \boldsymbol{I}_3 \end{bmatrix} \begin{bmatrix} \boldsymbol{y}_k \\ \boldsymbol{x}_k^w \end{bmatrix}; \begin{bmatrix} -\hat{\boldsymbol{R}}_k^\top \hat{\boldsymbol{p}}_k \\ \hat{\boldsymbol{x}}_{k|k-1}^w \end{bmatrix}, \begin{bmatrix} \boldsymbol{J}_{\boldsymbol{T}}^{\boldsymbol{T}^{-1}\cdot\boldsymbol{x}} \boldsymbol{\Sigma}_k \boldsymbol{J}_{\boldsymbol{T}}^{\boldsymbol{T}^{-1}\cdot\boldsymbol{x},\top} + \boldsymbol{M} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{P}_{k|k-1} \end{bmatrix}) \\
&\propto \mathcal{N}(\boldsymbol{x}_k^w; \hat{\boldsymbol{x}}_{k|k-1}^w + \boldsymbol{K}(\boldsymbol{y}_k - \boldsymbol{h}(\begin{bmatrix} \hat{\boldsymbol{x}}_{k|k-1}^w \\ \hat{\boldsymbol{T}}_k \end{bmatrix})), (\boldsymbol{I} - \boldsymbol{K}\hat{\boldsymbol{R}}_k^\top \boldsymbol{H}^m)\boldsymbol{P}_{k|k-1})
\end{aligned} \tag{87}
$$

with $\boldsymbol{K} = \boldsymbol{P}_{k|k-1}\boldsymbol{H}^{m\top}\hat{\boldsymbol{R}}_k\boldsymbol{S}^{-1}$, $\boldsymbol{S} = \hat{\boldsymbol{R}}_k^\top \boldsymbol{H}^m \boldsymbol{P}_{k|k-1}\boldsymbol{H}^{m\top}\hat{\boldsymbol{R}}_k + \boldsymbol{J}_{\boldsymbol{T}}^{\boldsymbol{T}^{-1}\cdot\boldsymbol{x}} \boldsymbol{\Sigma}_k \boldsymbol{J}_{\boldsymbol{T}}^{\boldsymbol{T}^{-1}\cdot\boldsymbol{x},\top} + \boldsymbol{M}$. The result follows from the standard Kalman filter equations.

5.2 Correlation-free body filter

For the correlation-free body filter, the measurement update is trivial, but the propagation is not. The targets are parameterized in the body frame of the platform $\boldsymbol{x}_k^b = [\boldsymbol{p}_k^b, \boldsymbol{v}_k^b]^\top$. The goal is for the propagation step to preserve Gaussianity on the target state. This is achieved using the linearization in Equation 81. To complete the propagation, $p(\boldsymbol{x}_{k+1}|\boldsymbol{y}_{1:k}) = \int p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k)p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k})d\boldsymbol{x}_k$ needs to be evaluated. The matrices of Equation 83 are denoted $\boldsymbol{A}$ and $\boldsymbol{b}$ respectively. The full

covariance from Equation 81 with the Jacobians of Equation 82 is denoted $\boldsymbol{B}$.

$$
\begin{aligned}
p(\boldsymbol{x}_{k+1}^b|\boldsymbol{y}_{1:k}) &= \int p(\boldsymbol{x}_{k+1}^b|\boldsymbol{x}_k^b)p(\boldsymbol{x}_k^b|\boldsymbol{y}_{1:k})d\boldsymbol{x}_k^b \\
&= \int \mathcal{N}(\boldsymbol{x}_{k+1}^b; \boldsymbol{A}\boldsymbol{x}_k^b + \boldsymbol{b}, \boldsymbol{B})\mathcal{N}(\boldsymbol{x}_k^b; \hat{\boldsymbol{x}}_k^b, \boldsymbol{P}_k)d\boldsymbol{x}_k^b \\
&\propto \int \exp\left(-\frac{1}{2} \begin{bmatrix} \boldsymbol{x}_{k+1}^b - \boldsymbol{A}\boldsymbol{x}_k^b - \boldsymbol{b} \\ \boldsymbol{x}_k^b - \hat{\boldsymbol{x}}_k^b \end{bmatrix}^\top \begin{bmatrix} \boldsymbol{B} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{P}_k \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{x}_{k+1}^b - \boldsymbol{A}\boldsymbol{x}_k^b - \boldsymbol{b} \\ \boldsymbol{x}_k^b - \hat{\boldsymbol{x}}_k^b \end{bmatrix} \right) d\boldsymbol{x}_k^b \\
&= \int \exp\Bigg(-\frac{1}{2} \begin{bmatrix} \boldsymbol{x}_{k+1}^b - \boldsymbol{A}\hat{\boldsymbol{x}}_k^b - \boldsymbol{b} \\ \boldsymbol{x}_k^b - \hat{\boldsymbol{x}}_k^b \end{bmatrix}^\top \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ -\boldsymbol{A}^\top & \boldsymbol{I} \end{bmatrix} \\
&\qquad\qquad \begin{bmatrix} \boldsymbol{B} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{P}_k \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{I} & -\boldsymbol{A} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_{k+1}^b - \boldsymbol{A}\hat{\boldsymbol{x}}_k^b - \boldsymbol{b} \\ \boldsymbol{x}_k^b - \hat{\boldsymbol{x}}_k^b \end{bmatrix} \Bigg) d\boldsymbol{x}_k^b \\
&= \int \exp\Bigg(-\frac{1}{2} \begin{bmatrix} \boldsymbol{x}_{k+1}^b - \boldsymbol{A}\hat{\boldsymbol{x}}_k^b - \boldsymbol{b} \\ \boldsymbol{x}_k^b - \hat{\boldsymbol{x}}_k^b \end{bmatrix}^\top \\
&\qquad\qquad \begin{bmatrix} \boldsymbol{A}\boldsymbol{P}_k\boldsymbol{A}^\top + \boldsymbol{B} & \boldsymbol{A}\boldsymbol{P}_k \\ (\boldsymbol{A}\boldsymbol{P}_k)^\top & \boldsymbol{P}_k \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{x}_{k+1}^b - \boldsymbol{A}\hat{\boldsymbol{x}}_k^b - \boldsymbol{b} \\ \boldsymbol{x}_k^b - \hat{\boldsymbol{x}}_k^b \end{bmatrix} \Bigg) d\boldsymbol{x}_k^b \\
&\propto \mathcal{N}(\boldsymbol{x}_{k+1}^b; \boldsymbol{A}\hat{\boldsymbol{x}}_k^b + \boldsymbol{b}, \boldsymbol{A}\boldsymbol{P}_k\boldsymbol{A}^\top + \boldsymbol{B})
\end{aligned} \tag{88}
$$

The equations above need the platform to be propagated with the targets. This is reasonable as the targets are parameterized relative to the platform. Something a bit less obvious is that the targets must be propagated with the platform too. This is because the motion model developed is only valid for IMU rate time steps. Pre-integrating measurements allows for the propagation rate to be disconnected between the two problems. Pre-integrating the motion model mean is easy, but the noise requires some work.[9], [22] This is not explored in this thesis, so the tracker and inertial navigation run at the same rate as the IMU.

The platform measures the position of the targets in it's own frame, so the measurement function is linear.

$$
p(\boldsymbol{y}_k|\boldsymbol{x}_k^b) = \mathcal{N}(\boldsymbol{y}_k; \boldsymbol{H}^m\boldsymbol{x}_k^b, \boldsymbol{M}) \tag{89}
$$

This makes for a straight-forward application of the Kalman filter. Since no correlations are tracked between target and platform, meaning they are assumed independent, no change happens to the platform state. This is also the case for the targets when the platform is updated. The proof is the same as for the correlation-free world filter.

As the targets are parameterized in the body frame, their uncertainty must be converted to the world frame before being visualized or communicated. The usual $\boldsymbol{x}^w = \boldsymbol{T} : \boldsymbol{x}^b$ can be used, but the nonlinearities are not easily dealt with. A linearization is possible with $\boldsymbol{x}^w \approx \hat{\boldsymbol{T}} : \boldsymbol{x}^b + \boldsymbol{J}_{\boldsymbol{T}}^{\boldsymbol{T}:\boldsymbol{x}}\boldsymbol{\delta T}$. Another approach is to embed the target state in tangent space of the platform pose. Defining the extended target state vector $\tilde{\boldsymbol{x}}^b = [0, 0, 0, \boldsymbol{v}^{b\top}, \boldsymbol{p}^{b\top}]^\top$, and correspondingly for the nine dimensional extended covariance $\tilde{\boldsymbol{P}}$, the target state is on the form of the tangent space of $\mathrm{SE}_2(3)$. The simplest way to convert between the target state and the $\mathrm{SE}_2(3)$ tangent space equivalent is to use the conversion matrix $\boldsymbol{H}^c = \begin{bmatrix} \boldsymbol{0}_3 & \boldsymbol{0}_3 \\ \boldsymbol{0}_3 & \boldsymbol{I}_3 \\ \boldsymbol{I}_3 & \boldsymbol{0}_3 \end{bmatrix}$. Then $\tilde{\boldsymbol{x}}^b = \boldsymbol{H}^c\boldsymbol{x}^b$, and $\tilde{\boldsymbol{P}} = \boldsymbol{H}^c\boldsymbol{P}\boldsymbol{H}^{c\top}$. Converting back is equally simple, $\boldsymbol{x}^b = \boldsymbol{H}^{c\top}\tilde{\boldsymbol{x}}^b$ and $\boldsymbol{P} = \boldsymbol{H}^{c\top}\tilde{\boldsymbol{P}}\boldsymbol{H}^c$. This sets the covariance of the rotational states to zero, along with the covariance between the rotational states and the vectorial states. To the best of the author's knowledge, this embedding of a target state in the tangent space of the tracker has not been done before. The pose of the target can now be described as

$$
\begin{aligned}
\boldsymbol{T}^w &= \boldsymbol{T}\operatorname{Exp}\tilde{\boldsymbol{x}}^b = \hat{\boldsymbol{T}}\operatorname{Exp}\boldsymbol{\delta T}\operatorname{Exp}\tilde{\boldsymbol{x}}^b \\
&\approx \hat{\boldsymbol{T}}\operatorname{Exp}(\boldsymbol{\delta T})\hat{\boldsymbol{T}}^b\operatorname{Exp}(\mathcal{J}_r(\hat{\tilde{\boldsymbol{x}}}^b)\tilde{\boldsymbol{\delta x}}^b) \\
&= \hat{\boldsymbol{T}}\hat{\boldsymbol{T}}^b\operatorname{Exp}(\operatorname{Ad}_{\hat{\boldsymbol{T}}^{b,-1}}\boldsymbol{\delta T})\operatorname{Exp}(\tilde{\boldsymbol{\delta x}}^b) \\
&\approx \hat{\boldsymbol{T}}^w\operatorname{Exp}(\operatorname{Ad}_{\hat{\boldsymbol{T}}^{b,-1}}\boldsymbol{\delta T} + \tilde{\boldsymbol{\delta x}}^b)
\end{aligned} \tag{90}
$$

where $\boldsymbol{T}^{b,w}$ is the $\mathrm{SE}_2(3)$ pose of the target in the body or world frame, $\tilde{\boldsymbol{\delta x}}^b$ is the zero mean

random variable with the covariance of $\tilde{\boldsymbol{x}}^b$, and $\hat{\tilde{\boldsymbol{x}}}^b$ is the mean of $\tilde{\boldsymbol{x}}^b$. The Jacobian disappears because the extended target state and covariance makes the product $\mathcal{J}_r(\hat{\tilde{\boldsymbol{x}}}^b)\tilde{\boldsymbol{P}}\mathcal{J}_r(\hat{\tilde{\boldsymbol{x}}}^b)^\top = \tilde{\boldsymbol{P}}$. The method for composing two distributions on the manifold is from [29] and generalizes to dependent distributions.

There are a few things to note about the method outlined above. The added rotational state in the target space defines a target body frame. Since the covariance of the target is supposed to be in the body frame of the platform, but the covariance will be defined in the target body frame, the added rotational state of the platform must be $\boldsymbol{I}_3$. This is achieved by prepending the tangent space vector with zeros. This ensures that the orientation of the body frame of the platform and the target coincide.

A natural thought might be to use the same approach for the body CV model. Since the composition operation for the vectorial quantities is exactly the same as the action used, this is possible. In fact, all we need for the $\mathrm{SE}_2(3)$ CV model is already defined. The $\boldsymbol{f}_{\delta t}$ function is the CV analogy on $\mathrm{SE}_2(3)$, with the $\boldsymbol{F}_{\delta t}$ matrix for the error coordinates. The CV model is $\boldsymbol{T}^b_{k+1} = \boldsymbol{T}^{-1}_{k+1}\boldsymbol{f}_{\delta t}(\boldsymbol{T}_k\boldsymbol{T}^b_k)\oplus\tilde{\boldsymbol{\gamma}}$, where $\tilde{\boldsymbol{\gamma}}$ is the extended $\boldsymbol{\gamma}$, $\tilde{\boldsymbol{\gamma}} = \boldsymbol{H}^c\boldsymbol{\gamma}$. There is a difference in how the noise is applied in this approach and Equation 78. Here, the noise is in the body frame of the platform, not in the world frame. Due to the symmetry of the noise covariance matrix of the CV model, this makes no difference.

$$\begin{bmatrix}\boldsymbol{R} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{R}\end{bmatrix}\begin{bmatrix}\delta t^3/3\boldsymbol{I}_3 & \delta t^2/2\boldsymbol{I}_3\\ \delta t^2/2\boldsymbol{I}_3 & \delta t\boldsymbol{I}_3\end{bmatrix}\begin{bmatrix}\boldsymbol{R}^\top & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{R}^\top\end{bmatrix} = \begin{bmatrix}\delta t^3/3\boldsymbol{I}_3 & \delta t^2/2\boldsymbol{I}_3\\ \delta t^2/2\boldsymbol{I}_3 & \delta t\boldsymbol{I}_3\end{bmatrix} \tag{91}$$

The mean of this CV model is $\hat{\boldsymbol{T}}^b_{k+1} = \hat{\boldsymbol{T}}^{-1}_{k+1}\boldsymbol{f}_{\delta t}(\hat{\boldsymbol{T}}_k\hat{\boldsymbol{T}}^b_k)$. Using this, and the adjoint rule of Equation 22 repeatedly, the error can be found.

$$\begin{aligned}
\hat{\boldsymbol{T}}^{b,-1}_{k+1}\boldsymbol{T}^b_{k+1} &= (\hat{\boldsymbol{T}}^{-1}_{k+1}\boldsymbol{f}_{\delta t}(\hat{\boldsymbol{T}}_k\hat{\boldsymbol{T}}^b_k))^{-1}\boldsymbol{T}^{-1}_{k+1}\boldsymbol{f}_{\delta t}(\boldsymbol{T}_k\boldsymbol{T}^b_k)\operatorname{Exp}\boldsymbol{\gamma}\\
&= \boldsymbol{f}_{\delta t}(\hat{\boldsymbol{T}}_k\hat{\boldsymbol{T}}^b_k)^{-1}\hat{\boldsymbol{T}}_{k+1}\boldsymbol{T}^{-1}_{k+1}\boldsymbol{f}_{\delta t}(\boldsymbol{T}_k\boldsymbol{T}^b_k)\operatorname{Exp}\boldsymbol{\gamma}\\
&= \operatorname{Exp}(-\operatorname{Ad}_{\boldsymbol{f}_{\delta t}(\hat{\boldsymbol{T}}_k\hat{\boldsymbol{T}}^b_k)^{-1}}\operatorname{Ad}_{\hat{\boldsymbol{T}}_{k+1}}\boldsymbol{\delta T}_{k+1})\boldsymbol{f}_{\delta t}(\hat{\boldsymbol{T}}^{b,-1}_k\hat{\boldsymbol{T}}^{-1}_k)\boldsymbol{f}_{\delta t}(\boldsymbol{T}_k\boldsymbol{T}^b_k)\operatorname{Exp}\boldsymbol{\gamma}\\
&= \operatorname{Exp}(-\operatorname{Ad}_{\boldsymbol{f}_{\delta t}(\hat{\boldsymbol{T}}_k\hat{\boldsymbol{T}}^b_k)^{-1}}\operatorname{Ad}_{\hat{\boldsymbol{T}}_{k+1}}\boldsymbol{\delta T}_{k+1})\operatorname{Exp}(\operatorname{Ad}_{\boldsymbol{f}_{\delta t}(\hat{\boldsymbol{T}}^{b,-1}_k)}\boldsymbol{F}_{\delta t}\boldsymbol{\delta T}_k)\\
&\qquad\qquad \circ \operatorname{Exp}(\boldsymbol{F}_{\delta t}\boldsymbol{\delta T}^b_k)\operatorname{Exp}\boldsymbol{\gamma}\\
\tilde{\boldsymbol{\delta T}}^b_{k+1} &= \operatorname{Log}(\hat{\boldsymbol{T}}^{b,-1}_{k+1}\boldsymbol{T}^b_{k+1}) \approx -\operatorname{Ad}_{\boldsymbol{f}_{\delta t}(\hat{\boldsymbol{T}}_k\hat{\boldsymbol{T}}^b_k)^{-1}}\operatorname{Ad}_{\hat{\boldsymbol{T}}_{k+1}}\boldsymbol{\delta T}_{k+1}\\
&\qquad\qquad + \operatorname{Ad}_{\boldsymbol{f}_{\delta t}(\hat{\boldsymbol{T}}^{b,-1}_k)}\boldsymbol{F}_{\delta t}\boldsymbol{\delta T}_k + \boldsymbol{F}_{\delta t}\boldsymbol{\delta T}^b_k + \boldsymbol{\gamma}\\
&= -\operatorname{Ad}_{\boldsymbol{f}_{\delta t}(\hat{\boldsymbol{T}}_k\hat{\boldsymbol{T}}^b_k)^{-1}}\operatorname{Ad}_{\hat{\boldsymbol{T}}_{k+1}}(\operatorname{Ad}_{\hat{\boldsymbol{\Delta}}^{-1}_k}\boldsymbol{F}_{\delta t}\boldsymbol{\delta T}_k + \boldsymbol{\vartheta})\\
&\qquad\qquad + \operatorname{Ad}_{\boldsymbol{f}_{\delta t}(\hat{\boldsymbol{T}}^{b,-1}_k)}\boldsymbol{F}_{\delta t}\boldsymbol{\delta T}_k + \boldsymbol{F}_{\delta t}\boldsymbol{\delta T}^b_k + \boldsymbol{\gamma}\\
&= (\operatorname{Ad}_{\boldsymbol{f}_{\delta t}(\hat{\boldsymbol{T}}^{b,-1}_k)} - \operatorname{Ad}_{\boldsymbol{f}_{\delta t}(\hat{\boldsymbol{T}}_k\hat{\boldsymbol{T}}^b_k)^{-1}}\operatorname{Ad}_{\hat{\boldsymbol{T}}_{k+1}}\operatorname{Ad}_{\hat{\boldsymbol{\Delta}}^{-1}_k})\boldsymbol{F}_{\delta t}\boldsymbol{\delta T}_k\\
&\qquad\qquad + \operatorname{Ad}_{\boldsymbol{f}_{\delta t}(\hat{\boldsymbol{T}}_k\hat{\boldsymbol{T}}^b_k)^{-1}}\operatorname{Ad}_{\hat{\boldsymbol{T}}_{k+1}}\boldsymbol{\vartheta} + \boldsymbol{F}_{\delta t}\boldsymbol{\delta T}^b_k + \boldsymbol{\gamma}
\end{aligned} \tag{92}$$

A first order approximation is used to combine all the exponentials. This shows that the error is approximately zero mean, which means that the CV estimator is unbiased. There is an issue with this method, the propagated mean does now generally have a non-identity rotational state. This is an issue, as the uncertainty is then not in the platform frame, but in the target frame. This can be solved by using the adjoint one more time.

$$\begin{aligned}
\boldsymbol{T}^b_{k+1} &= \begin{bmatrix}\boldsymbol{R}^b_{k+1} & \boldsymbol{v}^b_{k+1} & \boldsymbol{p}^b_{k+1}\\ \boldsymbol{0}_{2\times1} & & \boldsymbol{I}_2\end{bmatrix} = \begin{bmatrix}\boldsymbol{I}_3 & \boldsymbol{v}^b_{k+1} & \boldsymbol{p}^b_{k+1}\\ \boldsymbol{0}_{2\times1} & & \boldsymbol{I}_2\end{bmatrix}\begin{bmatrix}\boldsymbol{R}^b_{k+1} & \boldsymbol{0}_{3\times1} & \boldsymbol{0}_{3\times1}\\ \boldsymbol{0}_{2\times1} & & \boldsymbol{I}_2\end{bmatrix}\\
&= \begin{bmatrix}\boldsymbol{I}_3 & \hat{\boldsymbol{v}}^b_{k+1} & \hat{\boldsymbol{p}}^b_{k+1}\\ \boldsymbol{0}_{2\times1} & & \boldsymbol{I}_2\end{bmatrix}\underbrace{\begin{bmatrix}\hat{\boldsymbol{R}}^b_{k+1} & \boldsymbol{0}_{3\times1} & \boldsymbol{0}_{3\times1}\\ \boldsymbol{0}_{2\times1} & & \boldsymbol{I}_2\end{bmatrix}}_{=\boldsymbol{T}_{\hat{\boldsymbol{R}}^b_{k+1}}}\operatorname{Exp}\tilde{\boldsymbol{\delta T}}^b_{k+1}\\
&= \begin{bmatrix}\boldsymbol{I}_3 & \hat{\boldsymbol{v}}^b_{k+1} & \hat{\boldsymbol{p}}^b_{k+1}\\ \boldsymbol{0}_{2\times1} & & \boldsymbol{I}_2\end{bmatrix}\operatorname{Exp}(\operatorname{Ad}_{\boldsymbol{T}_{\hat{\boldsymbol{R}}^b_{k+1}}}\tilde{\boldsymbol{\delta T}}^b_{k+1})\boldsymbol{T}_{\hat{\boldsymbol{R}}^b_{k+1}}
\end{aligned} \tag{93}$$

At this point, the rotational state of the target can be discarded, as it is of no interest in the target tracking and the covariance has been returned to the body frame of the platform. The final error

estimate is then

$$\begin{aligned}\boldsymbol{\delta T}_{k+1}^{b} &\approx \mathrm{Ad}_{\boldsymbol{T}_{\hat{\boldsymbol{R}}_{k+1}^{b}}}(\mathrm{Ad}_{\boldsymbol{f}_{\delta t}(\hat{\boldsymbol{T}}_{k}^{b,-1})} - \mathrm{Ad}_{\boldsymbol{f}_{\delta t}(\hat{\boldsymbol{T}}_{k}\hat{\boldsymbol{T}}_{k}^{b})^{-1}}\,\mathrm{Ad}_{\hat{\boldsymbol{T}}_{k+1}}\,\mathrm{Ad}_{\hat{\boldsymbol{\Delta}}_{k}^{-1}})\boldsymbol{F}_{\delta t}\boldsymbol{\delta T}_{k} \\ &\quad + \mathrm{Ad}_{\boldsymbol{T}_{\hat{\boldsymbol{R}}_{k+1}^{b}}}\,\mathrm{Ad}_{\boldsymbol{f}_{\delta t}(\hat{\boldsymbol{T}}_{k}\hat{\boldsymbol{T}}_{k}^{b})^{-1}}\,\mathrm{Ad}_{\hat{\boldsymbol{T}}_{k+1}}\,\boldsymbol{\vartheta} + \mathrm{Ad}_{\boldsymbol{T}_{\hat{\boldsymbol{R}}_{k+1}^{b}}}\boldsymbol{F}_{\delta t}\boldsymbol{\delta T}_{k}^{b} + \mathrm{Ad}_{\boldsymbol{T}_{\hat{\boldsymbol{R}}_{k+1}^{b}}}\boldsymbol{\gamma} \\ &= (\mathrm{Ad}_{\boldsymbol{T}_{\hat{\boldsymbol{R}}_{k+1}^{b}}\boldsymbol{f}_{\delta t}(\hat{\boldsymbol{T}}_{k}^{b,-1})} - \mathrm{Ad}_{\boldsymbol{T}_{\hat{\boldsymbol{R}}_{k+1}^{b}}\boldsymbol{f}_{\delta t}(\hat{\boldsymbol{T}}_{k}\hat{\boldsymbol{T}}_{k}^{b})^{-1}\boldsymbol{G}_{\delta t}\boldsymbol{f}_{\delta t}(\hat{\boldsymbol{T}}_{k})})\boldsymbol{F}_{\delta t}\boldsymbol{\delta T}_{k} \\ &\quad + \mathrm{Ad}_{\boldsymbol{T}_{\hat{\boldsymbol{R}}_{k+1}^{b}}\boldsymbol{f}_{\delta t}(\hat{\boldsymbol{T}}_{k}\hat{\boldsymbol{T}}_{k}^{b})^{-1}\hat{\boldsymbol{T}}_{k+1}}\,\boldsymbol{\vartheta} + \mathrm{Ad}_{\boldsymbol{T}_{\hat{\boldsymbol{R}}_{k+1}^{b}}}\boldsymbol{F}_{\delta t}\boldsymbol{\delta T}_{k}^{b} + \boldsymbol{\gamma}. \end{aligned} \tag{94}$$

where $\mathrm{Ad}_{\mathcal{X}}\,\mathrm{Ad}_{\mathcal{Y}} = \mathrm{Ad}_{\mathcal{XY}}$, Equation 91 and Equation 58 has been used. It can be shown that this model is, in fact, equivalent to the linearization in Equation 80 by calculating the adjoints and Jacobians. The calculation is not done here, but will be shown numerically.

Going back and forth to the manifold raises the question whether modeling the targets on $\mathrm{SE}_2(3)$ is a better solution. There are a few reasons why this is not done. From a practical viewpoint, initializing a target's orientation might prove difficult. The relative position measurements used for the target measurements do not carry any information on the orientation of a target. Velocity can be estimated using the delta position of two measurements. Assuming the orientation to be in the direction of the movement still leaves a degree of freedom. Another approach is to initialize targets with a huge rotational uncertainty to cover all possible orientations with a reasonable probability mass. This ignores the concentrated Gaussian assumption. Even if this is fine, there is still the question of how the CV model works on the orientation. Is the orientation driven by white noise, or is it necessary to include an angular velocity state, thereby requiring the use of the $\mathrm{SE}_3(3)$ manifold for targets? These are all interesting questions, which could have been explored in this thesis, but the benefits of this approach over the regular CV model are not clear. The reason why the orientation is modeled on the platform is because it is needed to estimate the frame of the sensors. This is not the case for the targets. A potential gain of the manifold approach is a better understanding of a target's motion, as the encoding of the uncertainty in the exponential coordinates better models the nature of the movement.[23]

5.3 Correlated filters

Filters tracking the correlation between the platform and the target state should promise better performance. When the states are correlated, the information gained about one state will affect the other. This means that platform updates will change the distribution of the targets. There are many ways to parameterize the combined state vector. Parameterizing the targets in the world frame is probably the most straightforward. However, their error can still be local to the platform. This gives the simple propagation of the CV world filter, with the added benefit of the Exponential Gaussian distribution over the states. This is in some ways a best of both worlds from the correlation-free filters, as the Gaussian body filter uncertainty can be viewed as an Exponetial Gaussian in the world frame, as shown in Equation 90. A drawback of correlation filters is the added computational cost. The size of the state vector and covariance matrix grows with each added target, instead of adding a new state and covariance matrix per target. The total number of covariance matrix entries tracked is $(9+6N)^2$ and $9^2 + N \times 6^2$, respectively, where N is the number of targets. The complexity grows linearly in the correlation-free filters, but quadratically in the correlated filters.

Embedding the state vectors of the targets in the state vector of the platform creates a new manifold. Two different ways to do the embedding come to mind. The full SE-approach is the manifold $\mathrm{SE}_{2+2N}(3)$, where two new three-dimensional vector states are added for each of the N targets. They are the velocity and position of each target. As the vectorial states of the SE-group are in the world frame, the targets should be parameterized in the world frame too. Extending the $\mathrm{SE}_2(3)$ filter to accommodate this should not be too difficult.

A full ESKF approach can also be taken. The manifold $\mathrm{SO}(3)\times(\mathbb{R}^3 \times \mathbb{R}^3)^{N+1}$ can be used for this. Generalizing the combined tracking and state estimation task to any matrix manifold should be possible, as is done for the state estimators in code written for this work. Defining the correct

operations on the group should suffice. Another choice is a mix of the SE and ESKF approach. The manifold $\mathrm{SE}_2(3)\times(\mathbb{R}^3\times\mathbb{R}^3)^N$ can be the starting point for such a filter.

With the new manifolds, the groundwork is laid for different variations of the correlated filters. Still, there are some choices to be made. A full SLAM-like filter can be created, where the platform affects the targets, and the targets affect the platform. In some ways, this is the optimal solution to the problem, but it is also the most computationally expensive. Another approach is a Schmidt-Kalman filter. The Schmidt-Kalman filter denies the targets any possibility to affect the platform by setting the appropriate part of the Kalman gain to zero. This ensures that the platform state and covariance are unaffected by the target measurements. This might seem like an odd choice, but if the platform estimation is considered accurate without the target measurements, this can help mitigate the effect of biases, inaccurate target measurements, and wrong data association on the platform state.[7] Schmidt-Kalman is also less computationally expensive than the SLAM method. [6] implement both filters in one dimension, although not in an on-manifold way.

The implementation of correlated filters is left as future work.

6 Evaluation method

The performance of the filters derived in this work is evaluated on simulated data. In this section, the approach to the evaluation is presented.

6.1 Performance metrics

To evaluate the performance and consistency of the inertial navigation filter and trackers, the normalized estimation error squared (NEES) metric is used. This is possible for simulations as the ground truth is available. The usual metric is defined as

$$\epsilon_k = (\boldsymbol{x}_k - \hat{\boldsymbol{x}}_k)^\top \boldsymbol{P}_k^{-1} (\boldsymbol{x}_k - \hat{\boldsymbol{x}}_k). \tag{95}$$

This definition does not work when the state is exponentially distributed as the minus is not defined on the manifold. However, using the $\ominus$ operator, it is straightforward to define a NEES metric on the manifold. This metric is a NEES metric on the error coordinates of the Lie group.

$$\epsilon_k = (\boldsymbol{T}_k \ominus \hat{\boldsymbol{T}}_k)^\top \boldsymbol{\Sigma}_k^{-1} (\boldsymbol{T}_k \ominus \hat{\boldsymbol{T}}_k) \tag{96}$$

The NEES can be viewed as the inner product of the whitened state distribution with itself. Since a covariance matrix is symmetric, it has a spectral decomposition. According to the spectral theorem $\boldsymbol{P} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^\top$, where $\boldsymbol{V}^{-1} = \boldsymbol{V}^\top$. We have $\boldsymbol{P}^{-1} = \boldsymbol{V}\boldsymbol{\Lambda}^{-1}\boldsymbol{V}^\top = \boldsymbol{V}\sqrt{\boldsymbol{\Lambda}^{-1}}\sqrt{\boldsymbol{\Lambda}^{-1}}\boldsymbol{V}^\top$. This makes $(\boldsymbol{x} - \hat{\boldsymbol{x}})^\top \boldsymbol{P}^{-1} (\boldsymbol{x} - \hat{\boldsymbol{x}}) = \boldsymbol{\delta x}^\top \boldsymbol{V}\sqrt{\boldsymbol{\Lambda}^{-1}}\sqrt{\boldsymbol{\Lambda}^{-1}}\boldsymbol{V}^\top \boldsymbol{\delta x} = (\sqrt{\boldsymbol{\Lambda}^{-1}}\boldsymbol{V}^\top \boldsymbol{\delta x})^\top \sqrt{\boldsymbol{\Lambda}^{-1}}\boldsymbol{V}^\top \boldsymbol{\delta x} = \boldsymbol{l}^\top \boldsymbol{l}$ where $\boldsymbol{l}$ is a standard normal random variable $\boldsymbol{l} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. The square of a standard normally distributed variable is a χ^2 distributed variable. This means that if the filter model is correct, ϵ_k is a χ^2 distributed random variable. The distribution has n degrees of freedom, where n is the number of dimensions of the error. This is the mean of the distribution. Multiple Monte-Carlo simulations will be used. Summing the ϵs over N different realizations or N different time steps, the new random variable is also χ^2 distributed and has Nn degrees of freedom. The average NEES (ANEES) should therefore be a scaled χ^2 distribution. Based on this distribution, a confidence interval can be calculated.[14] A 95% confidence interval will be used.

The NEES is a measure of consistency. Good NEES scores are possible without great performance, as long as the covariance is of appropriate size. The root mean square error (RMSE) is used to evaluate the accuracy of the estimates. For an n-dimensional quantity the RMSE for N time steps is computed according to

$$\mu = \sqrt{\frac{1}{Nn} \sum_k^N (\boldsymbol{x}_k - \hat{\boldsymbol{x}}_k)^\top (\boldsymbol{x}_k - \hat{\boldsymbol{x}}_k)}, \tag{97}$$

$$\mu = \sqrt{\frac{1}{Nn} \sum_k^N (\boldsymbol{T}_k \ominus \hat{\boldsymbol{T}}_k)^\top (\boldsymbol{T}_k \ominus \hat{\boldsymbol{T}}_k)}. \tag{98}$$

6.2 Covariance hyper-ellipses

Another useful tool to get a feel for the performance of the filter and trackers are plotting the covariance ellipses. A covariance ellipse is a closed line of points with equal probability density. For normally distributed two dimensional random variables, these lines are elliptical. Using the whitening process above in reverse, drawing the ellipses is easy. For a standard normally distributed two dimensional variable, the covariance ellipses are perfect circles. The circle corresponding to one standard deviation away from the mean has a radius of one, and similarly for other radii. After computing the points of the circles the inverse whitening transform $\boldsymbol{x} = \hat{\boldsymbol{x}} + \boldsymbol{V}\sqrt{\boldsymbol{\Lambda}}\boldsymbol{l}$ transforms back to the correlated coordinates.

For exponentially distributed variables, the approach is very similar. The covariance ellipses are computed for the zero-mean error coordinates $\boldsymbol{\delta T}$. Then, using the Exponential map the mean is

perturbed by the points on the ellipse $\boldsymbol{T} = \hat{\boldsymbol{T}} \oplus \boldsymbol{V}\sqrt{\boldsymbol{\Lambda}}\boldsymbol{l}$. The Exponential map is a continuous map, which means the continuity of the ellipses are preserved. However, as it is non-linear the shape is not.

Creating points on a circle is straight-forward, but the approach outlined above is not constrained to two dimensional random variables. In three dimensional space, the circle correspond to a sphere. The error coordinates of $\mathrm{SE}_2(3)$ are nine dimensional, meaning the covariance hyper-ellipse are actually transformed 8-spheres. Generating the discretized coordinates of a hyper-sphere can be done using hyper-spherical coordinates.[30]

$$
\begin{aligned}
\theta_1, \ldots, \theta_{n-2} \in [0, \pi], \theta_{n-1} \in [0, 2\pi) \\
c_1 &= r\cos(\theta_1) \\
c_2 &= r\sin(\theta_1)\cos(\theta_2) \\
&\vdots \\
c_{n-1} &= r\sin(\theta_1)\cdots\sin(\theta_{n-2})\cos(\theta_{n-1}) \\
c_n &= r\sin(\theta_1)\cdots\sin(\theta_{n-2})\sin(\theta_{n-1})
\end{aligned}
$$

where r is the radius of the hyper-sphere, corresponding to the number of standard deviations the hyper-sphere is away from the mean.

Discretizing a nine dimensional surface quickly becomes computationally unfeasible when increasing the resolution to useful levels. A smarter approach is taken in [9]. Extracting only the three Eigenvectors of $\boldsymbol{V}$ corresponding to the three greatest Eigenvalues, three ellipses in the hyperspace can be constructed. The coordinates of the ellipses are given by $\boldsymbol{c} = r\sqrt{\lambda_i}\boldsymbol{v}_i\cos(\theta) + r\sqrt{\lambda_j}\boldsymbol{v}_j\sin\theta, \theta \in [0, 2\pi)$ for all three combinations of the three Eigenvector and Eigenvalue pairs (1 and 2, 1 and 3, and 2 and 3).

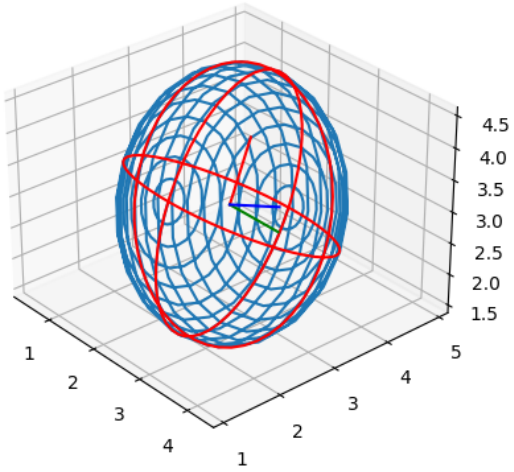
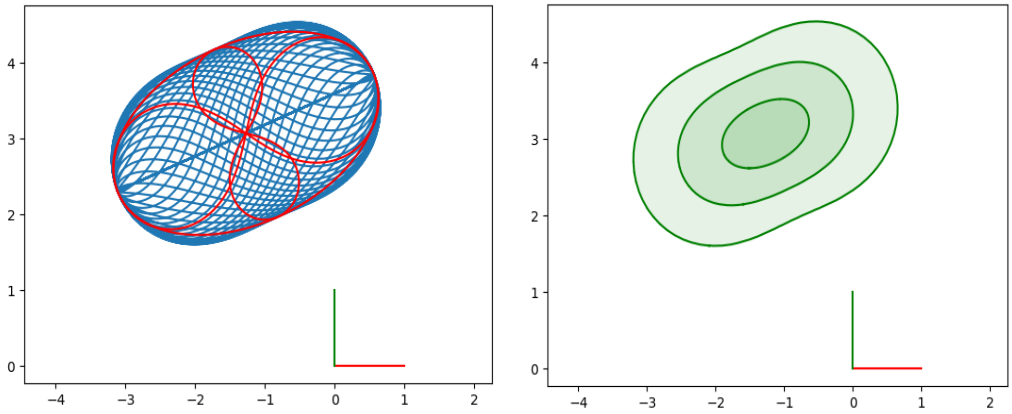


Figure 9: A discretized covariance hyper-ellipse in three dimensional space (blue) with the three covariance ellipses (red) corresponding to the three greatest Eigenvalues. The Eigenvectors are also plotted.

The SE(2) manifold has a three-dimensional tangent space. If the platform only rotates around the z-axis, and the position and velocity is in the xy-plane, the $\mathrm{SE}_2(3)$ covariance of the position can be equivalently described by SE(2). Since the Exponential map of $\mathrm{SE}_2(3)$ has no interaction between the velocity and positional states, the velocity state and covariance can be dropped before using the Exponential map when plotting the position, and the other way around. As the rotation is limited to the z-axis and the movement to the xy-plane, only the three states of the SE(2) are needed. For this reason, some of the trajectories used in the simulation will be constrained to follow this. Having the state described on SE(2) makes it possible to discretize the tangent space with reasonable accuracy and project onto the manifold using the Exponential map. Extracting the position of the



(a) The positional state of the projected covariance in Figure 9. The projection is the SE(3) Exponential map.

(b) Creating a polygon around the projected positions give the covariance polygons, the analogy to the covariance ellipses on the manifold. Here plotted for one, two, and three standard deviations. Method from [31].

Figure 10: The steps to creating a covariance polygon. The polygon is slightly larger than the greatest covariance of the Eigenvector approach.

projected points gives the covariance polygon for the position. This approach allows for plotting the positional state of $\mathrm{SE}_2(3)$ in a manner similar to regular covariance ellipses of normally distributed variables. Figure 10a and Figure 10b illustrate the steps to create the covariance polygon. The three covariance ellipses are also projected to visualize how they summarize the shape well. The marginalization of the rotational state makes the lines cross each other.

For the trajectories not constrained to be compatible with the SE(2) approach to plotting, the Eigenvector ellipses approach is still possible. It can easily be generalized to use more Eigenvectors to generate hyperspheres from the n most import Eigenvectors. As long as n is less than the dimensionality of the tangent space, it is computationally more efficient than a brute-force approach. The brute-force discretization of the entire covariance hypersphere is also possible, but this often requires such a reduced number of points in each dimension to be computationally feasible that the results are useless.

6.3 Simulation

To create data for evaluation, measurements will be simulated. Using simulated data over real data has some advantages. When simulating, there is no need to tune the covariance matrices of the measurement and process noise to match the properties of the real system, as the simulated data can be drawn from distributions of our choice. The main drawback is that any unmodeled properties of the real system are not included in the analysis. This should not be an issue for the filter and trackers in this work, as there is no actual modeling of any physical system, except the sensors. This means that the only real challenge skipped when simulating is to tune the measurement and process noise. The perhaps most important advantage is the availability of ground truth. Without ground truth, the NEES metric would be unavailable, and the statistical properties of the innovation would have to suffice for analyzing the consistency of the system.[14]

There are two different random variables in the propagation step. The initial state has a random variable $\boldsymbol{T}_0 = \hat{\boldsymbol{T}}_0 \oplus \boldsymbol{\delta T}_0$, and every IMU measurement has a noise $\boldsymbol{\eta}$. Drawing $\boldsymbol{\delta T}_0$ and $\boldsymbol{\eta}_{1,\dots,N}$ from their corresponding normal distributions, the state can be propagated N times. Drawing a great number of random vectors, many different realizations can be propagated. This Monte Carlo approach creates data for the NEES and ANEES statistics. Defining the ground truth IMU measurements, the ground truth state can be found through integration and Equation 24.

Assuming the $\mathrm{SE}_2(3)$ model to accurately model the uncertainty, the covariance of the Monte

Carlo simulated states, calculated with Algorithm 1, should be similar to the covariance of the unperturbed state. Along the lines of [9], the Frobenius norm will be used to compare the simulated and Monte Carlo calculated covariance matrices. To quantify the error, the Frobenius norm of the difference between the exact and averaged covariance matrix is calculated. The Frobenius norm is the root of the sum of squared entries in the matrix.[32]

Analyzing the spread of the simulated platforms, the performance can be evaluated. The different variants of the increment matrix defined in Equation 51, Equation 52, and Equation 53 will be compared by Monte Carlo simulations. To avoid the effect of the discretization caused by the piece-wise constant measurements assumption, constant measurements can be used.

The targets will be propagated using the CV-world model of Equation 77 where the additive noise will be drawn from the distribution of $\boldsymbol{\gamma}$. This will also be the case for the body tracker, which will make it possible to analyze the effects of the linearizations. The true body pose can easily be recovered by using the ground truth position of the platform. The performance of the body tracker can be analyzed in both the body frame and the world frame. For comparison with other filters, the same frame must be chosen. The analysis is done in the world frame in this work, although performance and consistency in the body frame is perhaps equally important for systems relying on data association.

7 Inertial navigation results

This section summarizes the results of the inertial navigation filter. The propagation agrees with [9], but that article does not implement an update step of the filter. Therefore, the update step receives more attention. In addition, the effect of the different discretization techniques is explored. For comparison, the ESKF of [8] is simulated, implemented as the $\mathrm{SO}(3)\times\mathbb{R}^3 \times \mathbb{R}^3$ filter.

7.1 Discretization

The different definitions of the increment matrices from the IMU measurements will be explored. Two of the methods use a locally constant assumption and the third uses a globally constant IMU measurement assumption. Of the local methods Equation 27 should be the most accurate, but the effort of calculating the correction term of Equation 45 is not unlikely offset by the noise. The correction term also requires an approximation to be able to transform the noise to the exponential coordinates of the increment matrix (see Equation 51). The other approach uses $\mathcal{J}_l$ as an approximation of $\mathcal{K}_l$ in Equation 27c. This is equivalent to approximating $\boldsymbol{C}$ from Equation 45 as the identity. The third approach uses Equation 28. This is the method used in [9]. Both the noise and the increment matrices differ slightly from method to method.

The best performing method depends on the trajectory. If the acceleration is globally constant, then the constant global method will, of course, outperform the others. Similarly, if the acceleration is locally constant, the local methods will outperform the global method. Generally, neither of the assumptions are true. The body experiences changes in angular velocity and acceleration during time steps, so they are not constant locally or globally.

Six different trajectories are tested. In one trajectory, the acceleration is constant locally. In another, it is constant globally. In the remaining four, the acceleration and angular velocity change either quickly or slowly, for a total of four possible combinations. The error is calculated as the mean of the Euclidean distance between the calculated trajectory and the ground truth at each time step as a percentage of the trajectory length. This metric is chosen to compensate for the increased difference caused by greater accelerations. Each trajectory is simulated for 20 seconds with a step length of 0.1 seconds. The results are summarized in Table 5.

There are a few things to note about these results. As expected, the constant local and constant global methods discretize exactly on their respective trajectories. The constant local approximation method achieves almost exactly the same results as the unapproximated method on all trajectories. This indicates that the effort of calculating the correction matrix $\boldsymbol{C}$ is not justified. Although this analysis does not consider the corresponding noise matrices for each method, it is not unreasonable to assume that this result is transferable to them. This is because the approximation tends to the exact matrix when $\boldsymbol{C}$ tends to the identity matrix, which is exactly what the result found above indicates.

On the trajectory defined by the quickly varying gyro- and accelerometer measurements, the constant global method diverges completely. This is shown in Figure 11. With this exception, the

Trajectory name \ Method	Constant local	Constant local approx	Constant global
Constant local acceleration	0.00%	0.01%	4.92%
Constant global acceleration	3.33%	3.34%	0.00%
Quick, quick	1.89%	1.80%	113.38%
Slow, slow	0.10%	0.10%	0.52%
Quick, slow	0.14%	0.14%	0.20%
Slow, quick	1.76%	1.76%	1.51%

Table 5: The error between the position of the ground truth trajectory and the trajectory calculated from the IMU measurements. The error is measured using the mean of the 2-norm of the difference as percentage of the ground truth trajectory length. The last four trajectory names reflect the speed of the gyro and acceleration changes, respectively.

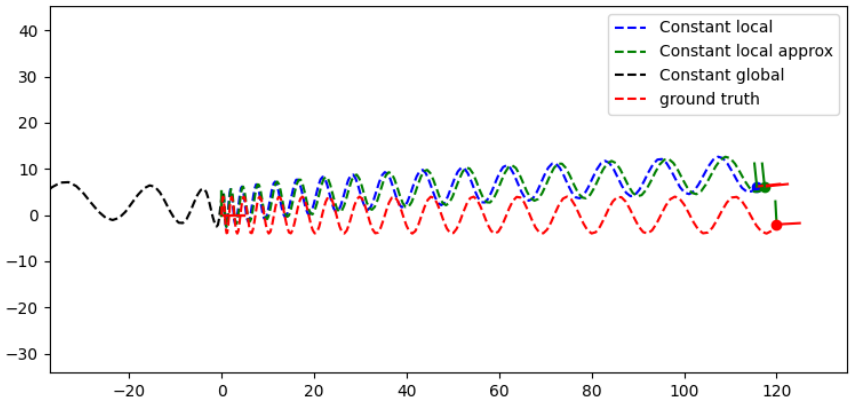


Figure 11: In some scenarios the constant global discretization diverges entirely.

constant global method performs similarly to the other methods, although perhaps slightly worse on average.

Based on the results in this section, the constant local approximation method is used to calculate the increment matrix going forward in this work for the $\mathrm{SE}_2(3)$ filter. This allows for defining the increment matrix as $\mathbf{\Delta} = \mathrm{Exp}[\boldsymbol{\omega}\delta t, \boldsymbol{a}\delta t, \frac{1}{2}\boldsymbol{a}\delta t^2]^\top$. With the $\mathrm{SO}(3)\times\mathbb{R}^3 \times \mathbb{R}^3$ Exponential map, this creates the increments of the constant global approximation, which will be used for this filter. This is for simplicity in the implementation, as the increments can then be defined equally for both filters with the Exponential map making the difference. The constant global approximation is also the typical formulation used in the ESKF.[8] Using the constant local approximation assumption makes sense, since motors accelerating the platform will rotate with the platform. The global assumption is better suited for accelerations caused by external forces, such as wind, however, these should be less important.

7.2 Propagation

The results of this section are similar to [9]. Using Algorithm 2, the filter estimate is propagated. The key element in the filter is the Exponential map. Modeling the uncertainty on the exponential coordinates, the non-linear properties of the map are preserved. Figure 13 shows this effect. Here, the propagation step has been run for 300 steps of 0.05 seconds, with noise only around the gyro z-axis for both the ESKF and the $\mathrm{SE}_2(3)$ filter. Clearly, both filters have similar magnitude on their covariance, but the "banana"-shape of the $\mathrm{SE}_2(3)$ filter better model the nature of the true

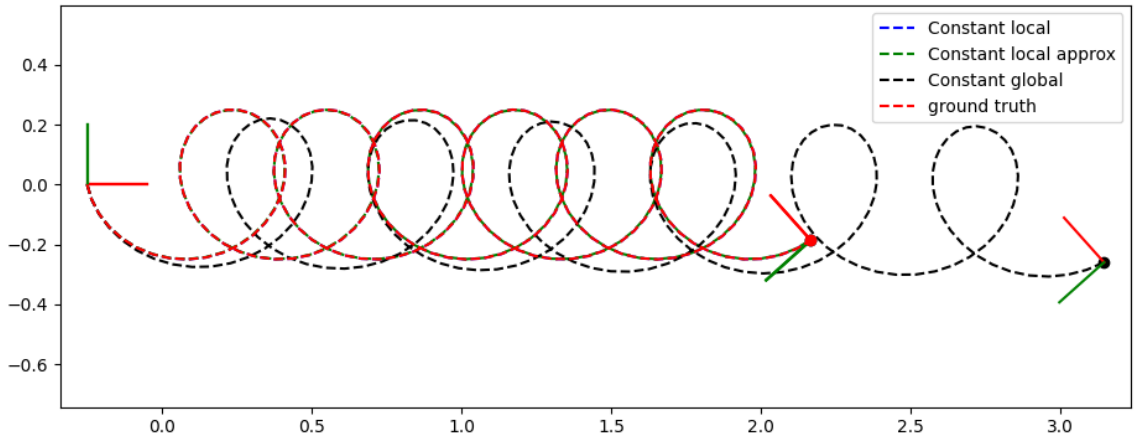


Figure 12: On the constant local acceleration trajectory, the constant global discretization approach performs poorly. The other methods are not visible, as they are covered by the ground truth.

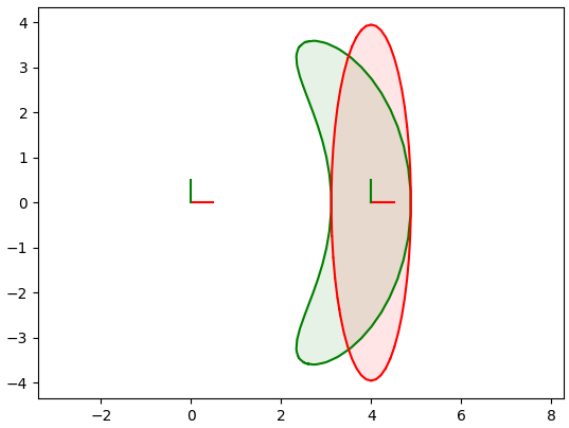


Figure 13: The difference between using the ESKF (red) and the filter in this work (green) is the dependence between the vectorial and rotational states in the Exponential map. This creates a "banana"-distribution over the vectorial states, instead of the normal Gaussian.

distribution. With noise only on the z-axis, the probable positions are along an arc since the samples will have traveled the same distance. This is verified by Monte Carlo simulations. In Figure 14, the means of 1000 samples are propagated with noisy IMU measurements. A full propagation, including the covariance, is done with the ground-truth measurements, for comparison. Using Algorithm 1, the mean and covariance of the samples are calculated. This is done for both filters. In the figure, the $\mathrm{SE}_2(3)$ Monte Carlo distribution is shown in yellow, and the ESKF Monte Carlo distribution is shown in orange. The main advantage of the $\mathrm{SE}_2(3)$ filter is shape of the covariance polygon, allowing it to include samples that deviate a lot from the ground-truth. If the full propagation had used noisy measurements, instead of the ground-truth, any of these samples could have been the estimated mean of the filter. When this is the case, the ESKF will have significantly less probability mass on the true mean than the $\mathrm{SE}_2(3)$ filter, making the update step less effective.

The Monte Carlo distributions are the best fitting distributions on the chosen group. Using these as the true distribution assumes that the samples actually are distributed according to the group distribution. Clearly, this is not the case. There is no way for the platform to be outside the arc of radius 112.5 m, but this area has probability mass. This non-Gaussian phenomenon is not accurately modeled by either $\mathrm{SE}_2(3)$ or $\mathrm{SO}(3)\times\mathbb{R}^3 \times \mathbb{R}^3$.

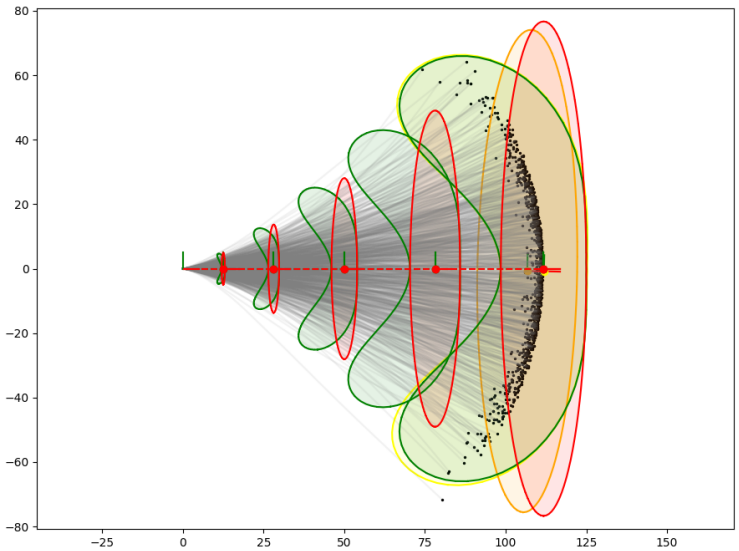
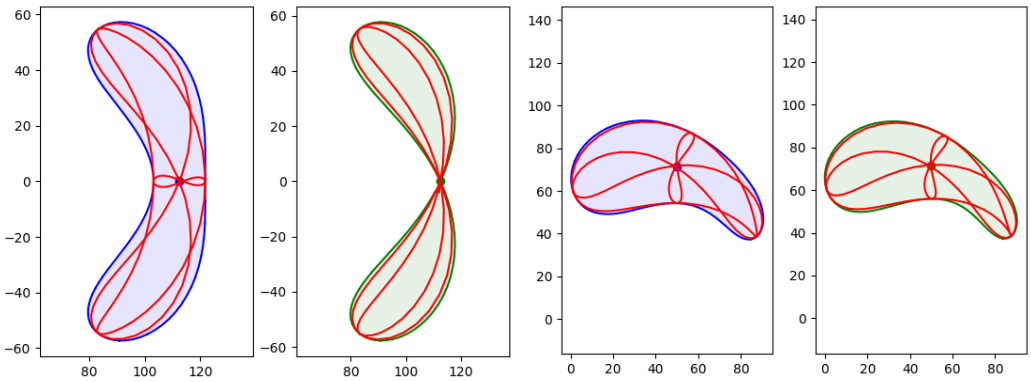


Figure 14: The $\mathrm{SE}_2(3)$ filter (green) is a better model of the true distribution of the Monte Carlo samples than the ESKF (red).



(a) The fourth order contribution creates a spread along the x-axis. This is shown by the existence of the covariance ellipsis spanned by the 2nd and 3rd greatest Eigenvectors.

(b) In some scenarios the fourth order contribution makes little difference.

Figure 15: The fourth-order contribution in the covariance propagation is only included in the left plot of each subplot. The left subplot is the same example as in [9].

Interestingly, the Monte Carlo mean only coincides with the filter mean for the $SE_2(3)$ filter. The orange Monte Carlo covariance ellipsis is slightly offset from the red ESKF covariance ellipsis of the final state. This is not the case for the $SE_2(3)$ filter. [9] proves this to be caused by the lack of dependence between the rotational and vectorial states in the Exponential map of the ESKF.

The fourth-order contribution of Equation 59 should be justified given the size of the expression and the number of first-order approximations used elsewhere in the filter. In Figure 15, the effect of the fourth-order contribution is shown for two different propagation examples. In the first example, the contribution clearly makes a difference. It captures the probability of the IMU noise randomly steering the platform estimate off-course and then back on-course. This makes the estimate fall a bit short of the ground truth, but on the right course. The next example shows that the fourth-order contribution does not always capture any meaningful probability. In [9], the fourth-order contribution is dropped to allow for the development of a preintegration theory. Preintegration is not considered in this work, but it does indicate that the fourth-order contribution is omissable.

The more noise on the translation compared to the orientation, the more Gaussian the $SE_2(3)$ distribution becomes. Figure 16 shows an example of this. In this example, the accelerometer noise dominates the gyro noise. The $SE_2(3)$ distribution is then similar to the Gaussian ESKF distribution.

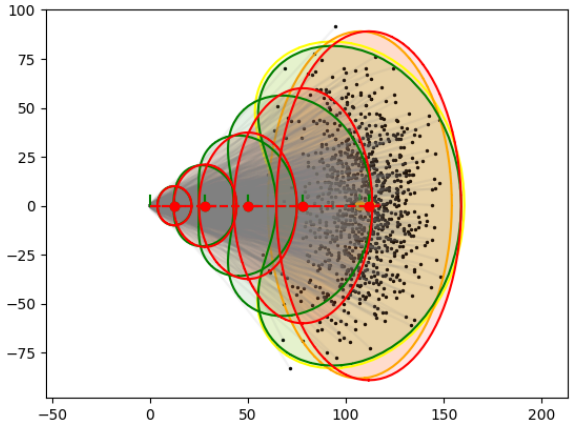


Figure 16: With more noise on the accelerometer compared to the gyro, the "banana"-effect diminishes, shrinking the difference between the ESKF and the $SE_2(3)$ filter.

In agreement with [9], the $SE_2(3)$ filter propagation is especially suited for cases where gyro noise dominates accelerometer noise, since it better models the connection between rotational and vectorial uncertainty. When this is the case, the "banana"-shape of the distribution makes a difference for accurate modeling. Also, when the initial uncertainty is greater than the sensor noise, the filter is more accurate, since Equation 58 then has a closed-form solution.

1000 different platforms are simulated with randomly sampled inputs. As expected, approximately 95% of the NEESs are within the 95% confidence bounds. The NEESs are also averaged for each of the random platforms, giving the ANEES. In addition, the ANEESs are averaged across platforms, giving the average ANEES (AANEES). The results are summarized in Table 6. Although the NEESs are within the bounds, most of the ANEESs are not. Many are below the bounds, indicating that the filter is under-confident. Figure 14 hints at why this might be the case. The Gaussanity of the error coordinates does not capture the unsymmetrical properties of the distribution of the samples. For there to be probability mass on the inside of the arc most samples get close to, there has to be probability mass on the outside, even though there is no chance of any samples being there. This makes the predicted covariance greater than needed. This is less of an issue compared to an over-confident filter since an over-confident filter will, to a greater degree, reject new information received in the update step. Figure 17 shows the NEES at each time step for 5 random samples. The filter seems over-confident, as the NEES curve is mostly in the lower part of the plots. This makes the ANEES lower than the expected values, despite the fact that the NEESs are within the bounds roughly the expected 95% of the time.

	NEES	ANEES	AANEES
$SE_2(3)$	94.0%	8.0%	Below bound
ESKF	90.8%	8.0%	Above bound

Table 6: The percentage of NEES and ANEES within the confidence bounds. The AANEES is a single value, and can therefore only be below, above or within its bounds.

7.3 Update

At the arrival of GNSS measurements, the update step is run according to Algorithm 3. The new information reduces the uncertainty of the state of the platform. The error dynamics of the filter ensures local convergence to the true state, with asymptotic stability.[11] This property is not shared by the ESKF.

Analyzing the update step with a comparison to the Monte Carlo distribution is less straightforward than the propagation. The updated Monte Carlo distribution should be some kind of weighted sum of the propagated samples, based on their probability to generate the GNSS measurement. NEES and plots will suffice for evaluating the performance.

The reset step of the filter is sometimes not included. This is the case, for example, in [11]. This is equivalent to assuming the update to small enough for the Jacobian to be close to identity.

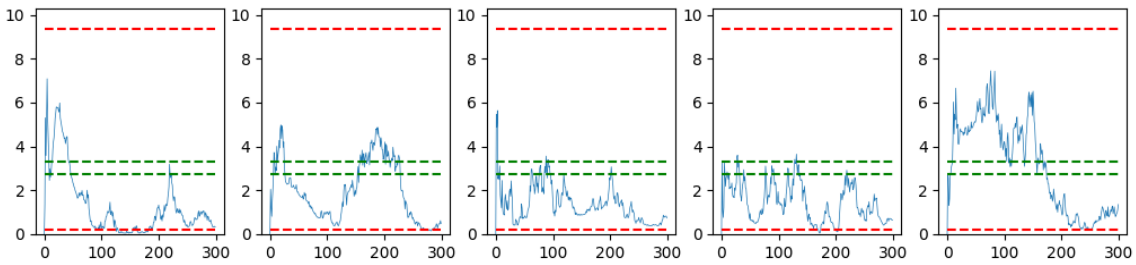
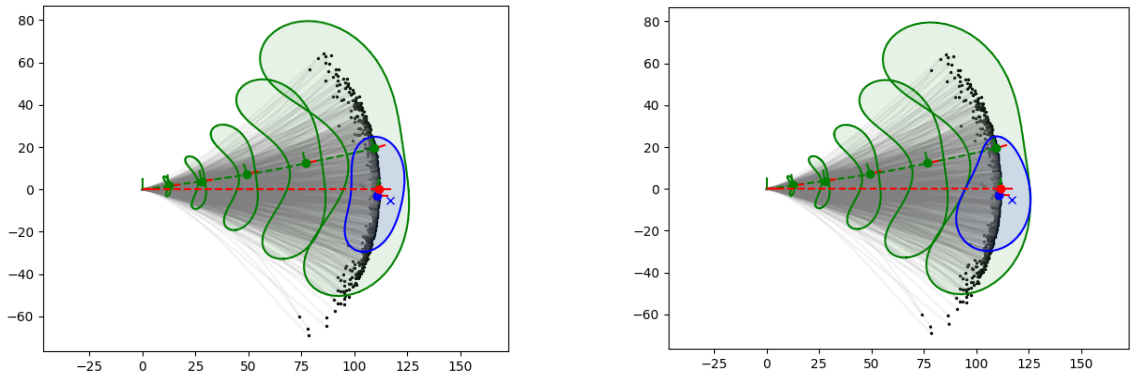


Figure 17: Some NEES curves for the $SE_2(3)$ filter. The red lines are NEES 95% confidence bounds. The green lines are the ANEES bounds. Clearly, the ANEES is lower than the expectation. This is often the case for the filter.



(a) The updated covariance without the reset.

(b) The updated covariance with the reset.

Figure 18: The same update with and without the reset step of Equation 75.

The bigger the update is, the more important the reset is, as it reparameterizes the error in the new local frame. Figure 18 is a comparison of the same update with and without the reset. The updated covariance has a significantly different shape after the reset step. However, as indicated by the neglect of the reset in [11], it is not important for the local stability properties of the filter.

To evaluate the performance of the filter, a sample trajectory is used. The trajectory emulates an airplane taking off and then performing some maneuvers. First, the plane circles above the take-off, before doing big banked turns with straights in between. The trajectory is continuous, but to avoid discretization errors it has been discretized. This is done by numerically estimating the angular velocity and acceleration, and then using Equation 26 to recalculate the trajectory.

Figure 19 shows the $\mathrm{SE}_2(3)$ filter (red) and ESKF (green) on the take-off part of the data. The significant ellipsis of the covariances of each filter right before and after the update steps are plotted. The curved distributions of the $\mathrm{SE}_2(3)$ filter better capture the actual uncertainty distribution, allowing for better update steps. This leads to the $\mathrm{SE}_2(3)$ converging on the ground truth, despite erroneous initial orientation. In this example, the trajectory has been discretized to remove any discretization errors.

To stress the stability properties of the filter, a simulation where the filter is initialized 90 degrees off from the ground truth pose, with small initial covariance, is simulated. Figure 20 shows how

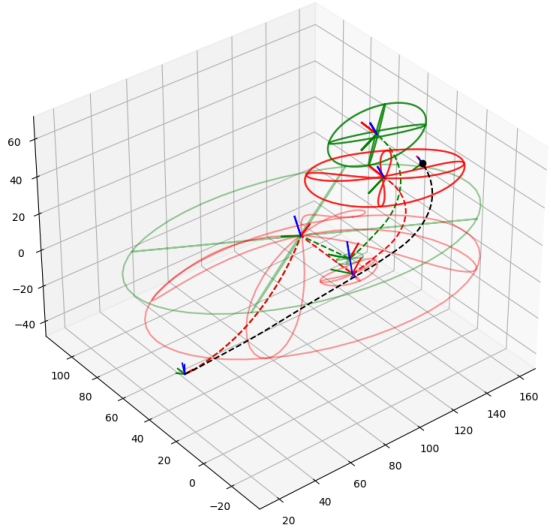


Figure 19: The curved distributions of the $\mathrm{SE}_2(3)$ allow for better update step accuracy.

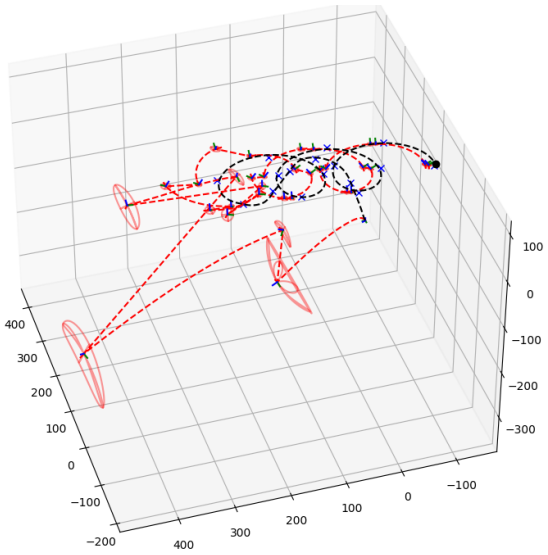


Figure 20: The $SE_2(3)$ filter converges to the correct state, despite very wrong initial conditions. This is a result of the stability properties of the filter that are not shared by typical EKFs. The initial uncertainty is artificially low compared to the error in this example, which hinders the effectiveness of the update step.

the filter converges to the correct pose, despite the erroneous initial conditions. The ESKF does not have the stability properties of the $SE_2(3)$, which manifests itself as the ESKF not being able to converge on the true trajectory.

Parameter	Value	Unit
Accelerometer noise std	1×10^{-2}	m/s^2
Gyro noise std	1×10^{-2}	rad/s
Initial covariance (per state)	0.2	rad, m/s, m
IMU frequency	100	Hz
GNSS frequency	0.2	Hz
GNSS noise std	2	m

Table 7: The parameters used in the simulation on the sample trajectory.

More detailed analysis is done on the full trajectory. The values used in the filter are summarized in Table 7. The ESKF is simulated using the same parameters as a comparison. The root mean square error (RMSE) of the estimates compared to the ground truth are calculated, as well as the NEES. The orientation is converted into Euler angles for the RMSE calculation. The NEES are evaluated on the total state estimate, and the position, velocity, and orientation individually. This makes it possible to evaluate the performance of the filters on each. The NEES statistics are reported as a percentage of the NEESs that are within the confidence bounds over 50 Monte Carlo runs. The results are summarized in Table 8.

The performance of the two filters is similar, but the $SE_2(3)$ filter slightly outperforms the ESKF on RMSE. In terms of consistency, the $SE_2(3)$ filter is far better. Calculating the NEES values without the first 5 seconds of each simulation, the filters are equally consistent. These figures can be found in Table 9. In this case, both are around the expected 95%. The fact that the $SE_2(3)$ filter is able to be consistent from the beginning is due to the better motion modeling, creating the curved distributions in Figure 19. The difference in the beginning is also evident in Figure 21, where the estimates of both filters are plotted against the ground truth. The curves overlap mostly, with only slight difference in the z-axis. As both filters use a tangent space representation of the orientation uncertainty, it is not surprising that the performance is almost identical. The uncertainty of the vectorial states is parameterized differently in the two filters. On the sample trajectory, the approach of the $SE_2(3)$ proves slightly better.

	$SE_2(3)$	ESKF
RMSE position	3.15	3.68
RMSE velocity	0.96	1.04
RMSE orientation	0.08	0.08
NEES total	93.2%	81.9%
NEES position	94.2%	84.7%
NEES velocity	93.6%	84.3%
NEES orientation	94.3%	84.9%

Table 8: The RMSE and NEES results of the $SE_2(3)$ and ESKF filter estimates on the sample trajectory. The RMSE of the orientation is computed by converting to Euler angles. The NEES is reported as the percentage of the values within the 95% confidence bounds, and an average of 50 Monte Carlo runs.

	$SE_2(3)$	ESKF
NEES total	95.2%	93.1%
NEES position	95.2%	93.7%
NEES velocity	94.9%	94.3%
NEES orientation	95.3%	93.7%

Table 9: The NEES scores when skipping the first 5 seconds of the simulation, averaged over 50 Monte Carlo runs.

The consistency scores of the $SE_2(3)$ are better than those of the ESKF, and around the expected 95%. However, the ANEES of the full trajectory is above the expected range. Removing the first 5 seconds produces an ANEES below the expected range for the $SE_2(3)$. The ESKF ANEES is still above the confidence bounds. Inspecting Figure 22, noting the logarithmic scale, the ANEES of both filters can be seen to be above the expected range. The consistency is poor in the beginning, especially for the ESKF that reaches values hundred times higher than expected. Considering that the ESKF is widely used in both literature and practice, and the $SE_2(3)$ filter outperforms the ESKF, it seems like a good argument for the capabilities of the filter.

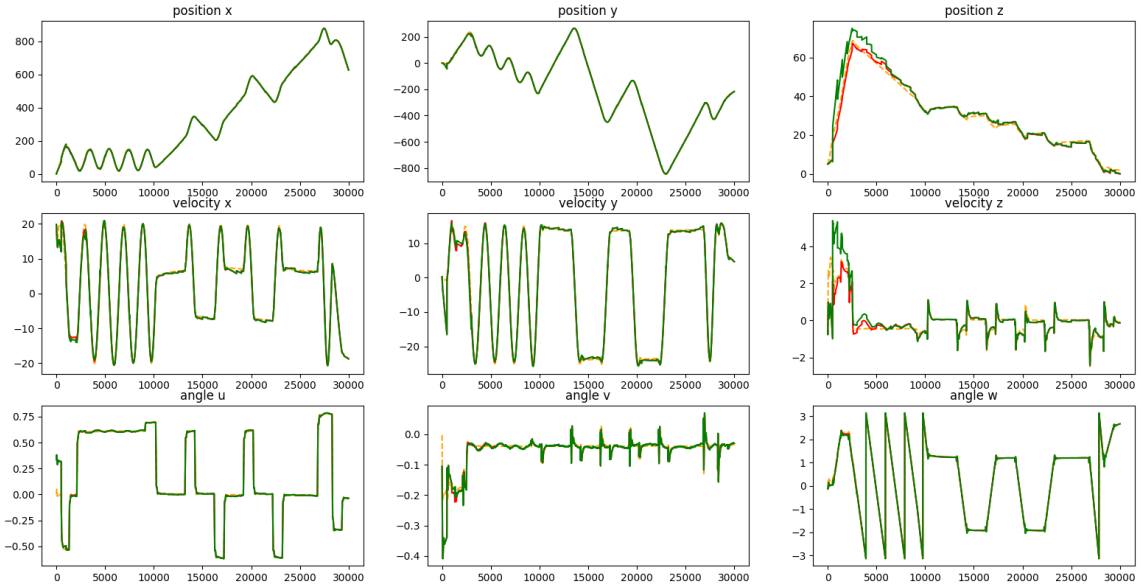


Figure 21: The estimation of the position, velocity and orientation. The ground truth is plotted in orange, but is mostly covered by the $SE_2(3)$ estimates plotted in red and the ESKF plotted in green. The orientation is represented by Euler angles.

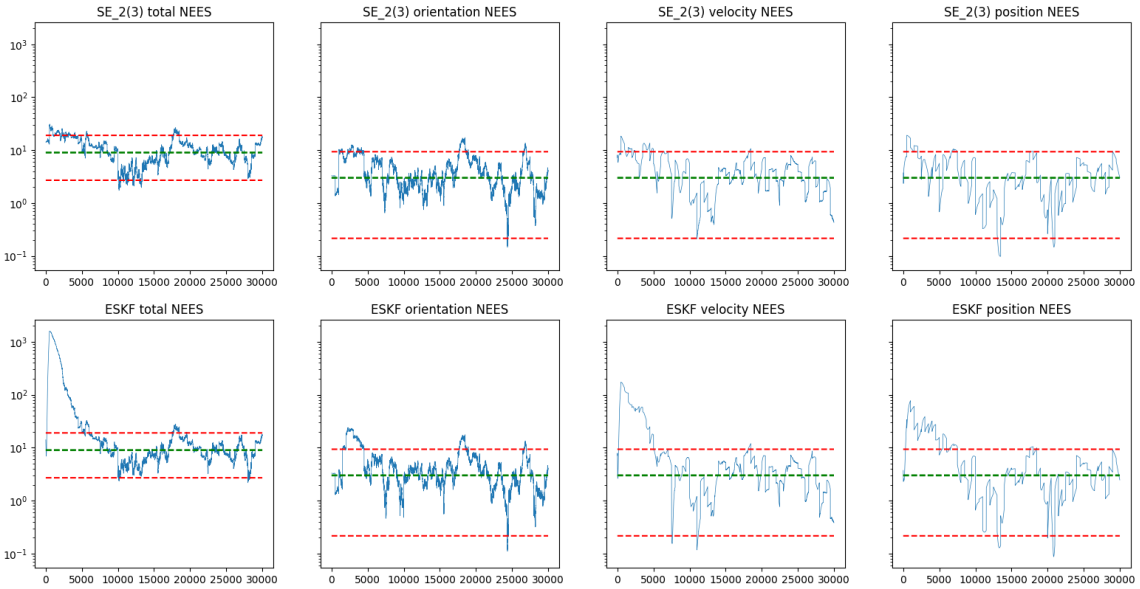


Figure 22: The NEESs of the $\mathrm{SE}_2(3)$ filter and the ESKF filter on the sample trajectory with the parameters of Table 7.

8 Tracking results

The trackers derived in Section 5 compensate for the uncertainty of the platform from which the tracking is performed. Especially the rotation uncertainty of the platform is vital to a correct understanding of the distribution of the target state. This is the case because small errors in orientation can give a very wrong understanding of the position of the target, as the relative measurements are in the local frame of the platform. Small angular errors create big discrepancies for long-range measurements. Figure 23 explains this visually. Position uncertainty affects measurements the same regardless of the range of the measurement. Assuming the state of the platform to be known dramatically underestimates the distribution of the target when the state of the platform is uncertain. As a baseline for comparing the trackers, a tracker assuming the estimates of the platform to be the ground truth is used. This will be referred to as the "naive" tracker.

The performance of the filters are tested on both the $\mathrm{SE}_2(3)$ and ESKF filters. This creates many different possible combinations of inertial navigation and tracker. To reduce the number of different combinations, the body tracker is investigated first, so a single variant can be used for the comparison with the other filters. Still, there are eight variants. Combining the $\mathrm{SE}_2(3)$ and ESKF with the body and world trackers gives four possible combinations. In addition, a naive tracker is included for each of the inertial navigation filters and motion models. The naive trackers assume the inertial navigation filter estimates are the ground truth, and uses the corresponding motion model. The implementation follows from the tracker equations by setting the platform uncertainty covariance $\mathbf{\Sigma}$ to zero.

Parameter	Value	Unit
Simulation time	100	s
Accelerometer noise std	1×10^{-2}	$\mathrm{m/s^2}$
Gyro noise std	1×10^{-2}	rad/s
Initial covariance (per state)	0.2	$\mathrm{rad}, \mathrm{m/s}, \mathrm{m}$
IMU frequency	100	Hz
GNSS frequency	0.2	Hz
GNSS noise std	2	m
Target measurement frequency	1	Hz
Target measurement noise std	5	m
Initial target covariance (body, per state)	2	$\mathrm{m/s}, \mathrm{m}$
Target CV noise std	2	m/s

Table 10: The parameters used in the evaluation of the full inertial navigation and tracker solution.

Table 10 summarizes the parameters used in the evaluation. The initial covariance of the target requires some explanation. Giving the world and body filters the same initial covariance means

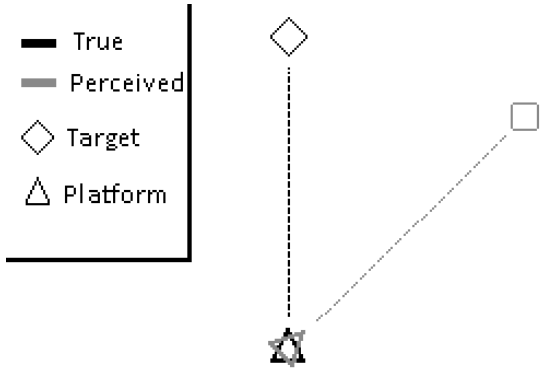


Figure 23: If the estimated orientation of the platform is off, the perceived world position of the target can be very wrong.

the world filter is initially far less uncertain than the body filter. This is because the body filter adds the platform uncertainty on the conversion to the world frame. Of course, the platform uncertainty is a part of the propagation step too, but this is to calculate the body motion, not the world uncertainty. To give all trackers the same initial conditions, the initial target covariance is defined for the body filter. The initial body state is then converted to the world frame, where the platform uncertainty is injected. This state is used as the world tracker initial state. The simulation is run for 100 seconds. At this point the inertial navigation filters have usually converged on the true trajectory, making all trackers reasonably accurate.

8.1 Equivalence of body filter variants

In Section 5.2, two different ways were derived to convert the target uncertainty in the body frame into the world frame. In addition, the body filter propagation step was derived in two ways; through linearizations (Equation 88) and on-manifold linearizations (Equation 92, Equation 93). The claim that these two approaches are identical was made and is backed up in this section.

Figure 24 visualizes the two different approaches to converting the body uncertainty to the world frame. In the left plot, the platform has some uncertainty on the target (yellow), and some uncertainty on its own state (red). The conversion to the world frame can be done through linearization or on-manifold linearization. These two are shown in the right plot in, respectively, green and red.

The state of the platform seems Gaussian in the position coordinates in the left plot. Fitting a Gaussian distribution on the platform in the left plot and using this to convert the body uncertainty gives the blue distribution on the right. This is included to show the importance of the rotational uncertainty. The platform state looks Gaussian, since the rotational uncertainty is not easily plotted, however, the rotational state of the platform makes a big difference. Comparing the uncertainty of the converted states to a naive body approach emphasizes the effect of the platform uncertainty. This is in fact what the yellow covariance ellipsis is. The yellow ellipsis is even smaller than the blue one (note the scale difference of the plots). Clearly, the naive approach underestimates the uncertainty of the target in the world frame.

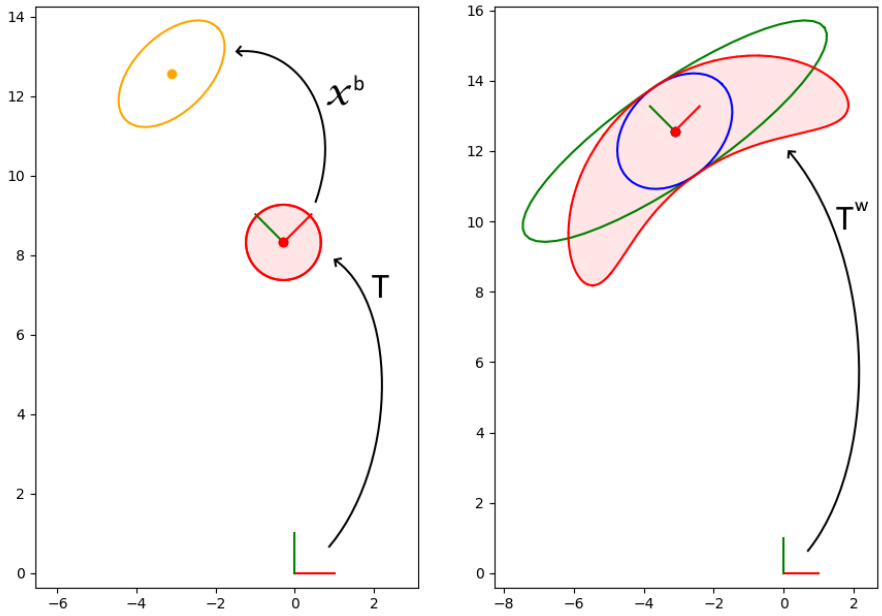
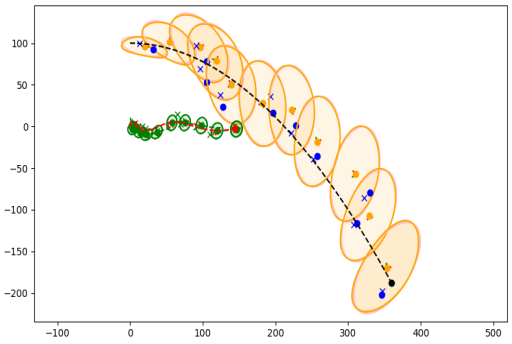
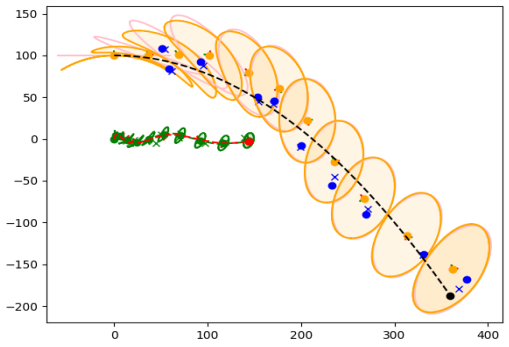


Figure 24: The different approaches to transforming the target from the platform's body frame to the world frame. Note that the scale difference between the plots, the blue ellipsis is bigger than the yellow.



(a) The covariance polygons from the two different approaches are nearly indistinguishable.



(b) Even when the noise is only on the gyro, making the initial polygons different, the tracking converges to something Gaussian.

Figure 25: Converting the body filter estimates to the world frame with the two different approaches, for two different noise configurations. The blue crosses are the actual measurement positions, the blue dots are the perceived positions.

The on-manifold conversion to the world frame approach "gives" an orientation to the target, which it does not actually have. This is an issue when calculating the NEES, as the ground truth of the target does not include any orientation. To fix this issue, the ground truth is given the same rotational state, which makes the rotational error zero. After using the logarithmic map of Equation 96, the rotational state and covariance are dropped, and the NEES is computed for position and velocity. This is equivalent to using the (:) action of the inverse manifold target state on the ground truth, and computing the NEES of this "error", taking into account the different ordering of the states.

Figure 25 shows the body tracker estimates converted to the world frame. Both methods are plotted in both plots, but this is difficult to see as they are very similar, especially as the target distribution converges on a Gaussian distribution. Inspecting the initial polygons of Figure 25b, the curvature of the manifold approach is visible. Considering the similarity of the results of the two approaches it should not be surprising that the NEES scores are similar. Actually, they are identical. The Logarithmic map of the manifold NEES unwraps the curvature of the manifold approach, yielding exactly the same result as the linearization approach. Considering they both represent the same tracker, they should have the same consistency property, explaining this result. This is not to say which should be used for other systems, e.g. collision avoidance, as the world distribution is different. Considering the similarity of the results, and that they both converge to something Gaussian, it should not make a big difference. However, investigating this is outside the scope of this thesis.

In Section 5.2 the claim was made that the two different body propagation equations derived are equivalent. This was not proven, but follows from inserting the correct values and some algebraic manipulation. Instead, this is verified through simulation. A platform tracking two targets, one with each method, is simulated. The 2-norm of the difference of the estimated means of each target at each time step is calculated. The same is done for the covariances with the Frobenius norm. Some care must be taken, since the manifold approach uses the $\mathrm{SE}_2(3)$ ordering of states. The average difference over 100 seconds of simulation is on the order of 10^{-12} for both norms. This is on the level of the precision that a computer can offer.

As all the variations of the propagation step and the world conversion of the body filter are equivalent, only one is examined going forward. The linearization approach is taken for both.

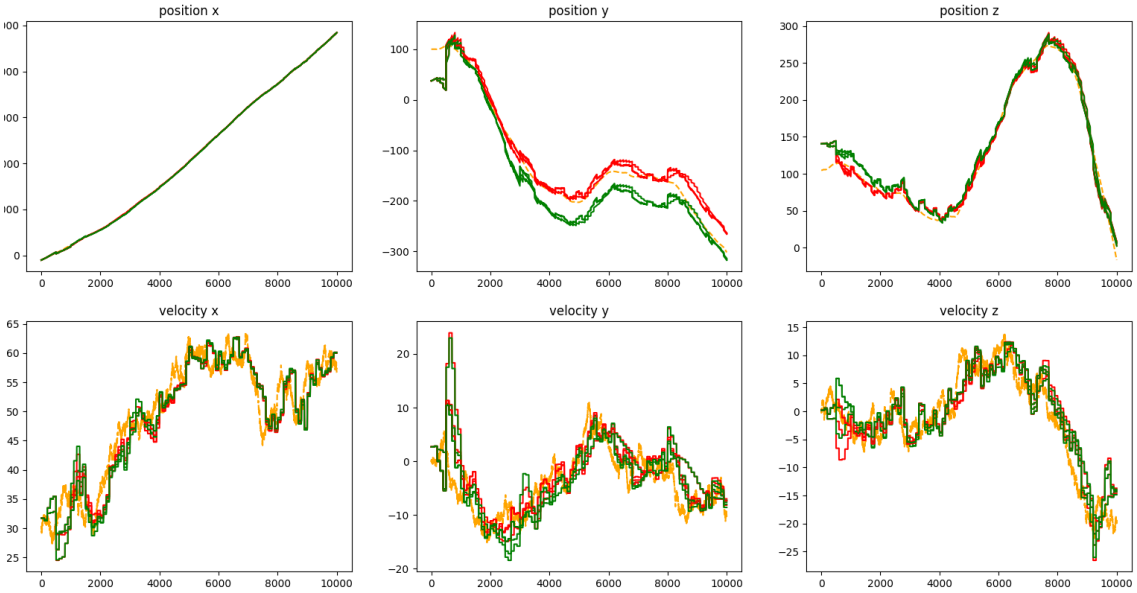


Figure 26: The estimates of the position and velocity for all trackers. The orange line is the ground truth. All the $SE_2(3)$ based trackers are plotted in red, whilst the ESKF based trackers are in green.

8.2 Tracker performance

Figure 26 shows the estimates of all 8 trackers on a sample run of the trackers. All trackers based on the same inertial navigation filter are plotted in the same color. The $SE_2(3)$ filters are in red, and the ESKFs are in green. At each received measurement, the target state estimates are updated. The smaller steps are updates from the target measurements. The big correction after 500 time steps, 5 seconds, is a consequence of the GNSS update of the platform. Although this update only corrects the state of the platform, the following target measurement update step uses this new information to correct the target estimates. Based on the estimates, it is not obvious which filters are which. There are some trackers that perform worse than the others in the beginning of the velocity estimates, but they are quickly corrected. The main difference seems to be between the filters in the position estimates. This is not surprising and is a consequence of the inertial navigation filters' performance. From Table 8, both inertial navigation filters accurately predict their velocity and orientation. The position scored slightly less well. This position error is injected into the tracker estimates. The difference between the naive and compensating trackers is in the uncertainty calculations. This affects the estimates only in the update step. In the beginning, the uncertainty is large enough for the update to be dominated by the measurements. After some time, the naive filters become over-confident, making the updates less efficient. The naive filters are visible in the velocity plot of the y-direction. Some of the graphs lag behind the others towards the end of the simulation, reacting slower to changes in the velocity.

The RMSE of the trackers are summarized in Table 11. As expected, the difference is greater between the inertial navigation filters used than between the trackers. The naive trackers use the same propagation equations for the mean of the predictions as the other tracker. The update step differs, but clearly does not make a big difference on the estimated mean.

	$SE_2(3)$	ESKF
World	12.0	16.0
Body	11.5	15.8
Naive world	12.0	16.1
Naive body	11.5	15.8

Table 11: RMSE of the trackers for a 100 seconds long run of the trackers.

8.3 World and body parameterization

Table 11 indicates that the body filters outperform the world filters. There are some properties of the body filters that might explain this. The body filter injects the uncertainty of the platform estimate at every propagation step. The world filter does this at every update step. Since the propagation step is typically run many times per update step, the platform uncertainty is injected more often in the body frame trackers. This can make the body filters more "connected" with the platform state. On conversion to the world frame, the body uncertainty is injected with the platform uncertainty again, further increasing the covariance. Additionally, the body filters are indirectly influenced by the GNSS updates of the platform. Since the platform state changes at GNSS updates, but the body frame estimates does not, the corresponding world coordinates of the target estimates change with the platform. The following target measurement updates are then run from better initial guesses. Figure 27 shows both of these phenomena. The true target trajectory is plotted in orange. The blue line is the $SE_2(3)$ body tracker, and the green line is the ESKF world tracker. After a few time steps the initial covariance of the world filter has been reduced, due to the target measurement updates. The body filter covariance is greater than the initial covariance. Then a GNSS update of the platform is done, leading to a sudden convergence of the body tracker to the ground truth. The world tracker does not react to the change until the next target measurement is received. The perceived locations of the target measurements for the two filters are plotted with x's in the corresponding color of the trackers. The difference is caused by the worse performance of the ESKF.

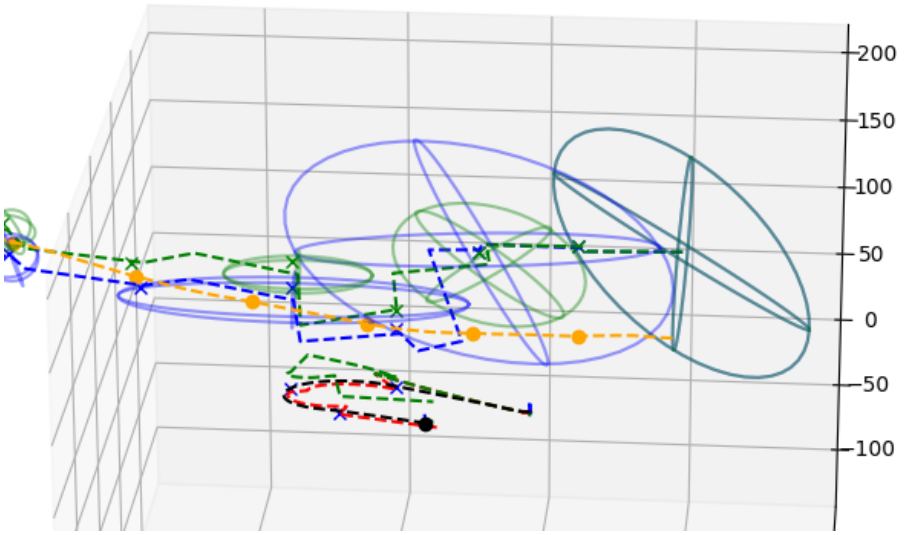


Figure 27: The $SE_2(3)$ body filter (blue) plotted against the ESKF world filter (green). The orange is the ground truth trajectory of the target. The orange dots mark the location of the target at each target measurement.

8.4 Tracker consistency

The increased uncertainty of the body trackers compared to the world trackers should be reflected in the NEES of the trackers. The consistency score captures whether a filter is over- or under-confident. In Figure 28 the NEES scores for all trackers are plotted for the same simulation as in Figure 26. As expected, the naive trackers are over-confident; their NEES scores are far above the confidence bounds for much of the simulation. Splitting the target state into the position and velocity states reveal that this is mainly caused by overconfidence in the positional estimates. The NEES of the velocity is shown in Figure 29, where all the trackers seem to score reasonably well. The position NEES is plotted in Figure 30, with curves almost identical in shape to the total NEES.

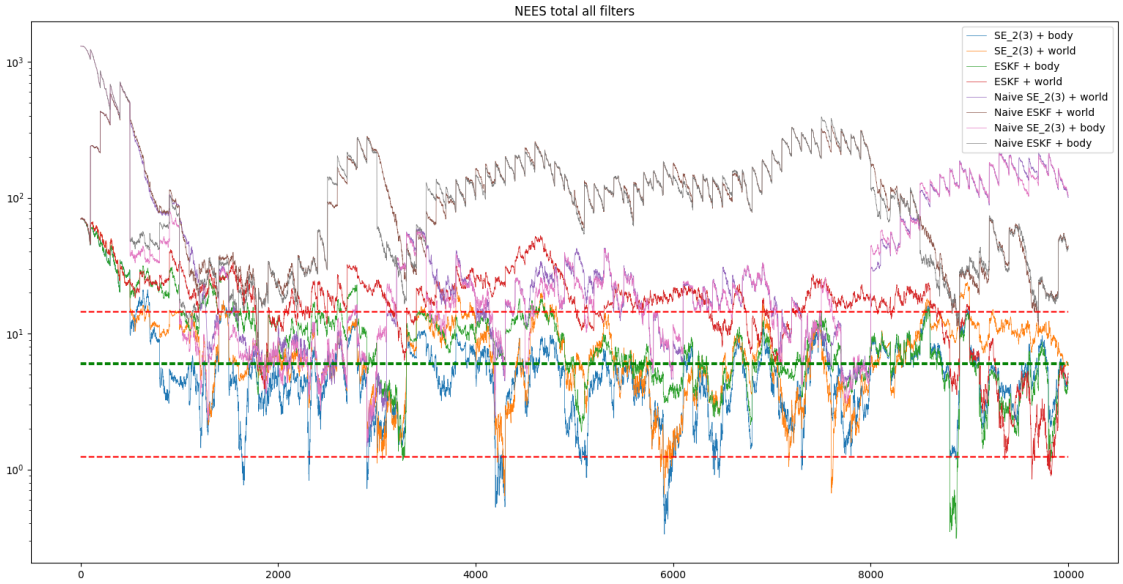


Figure 28: The total NEES of all the trackers.

In the beginning of the simulation, before the inertial navigation filters converge on the correct trajectories, the errors of the filters are inherited by the trackers. As with the NEES of the filters in Figure 22, the NEESs are above the confidence bounds in the beginning of the simulation. After this transient phase, the ESKF body tracker and both $SE_2(3)$ trackers seem consistent.

For better statistical properties, 100 Monte Carlo simulations are run with all trackers tracking the same target. Table 12 summarize the performance of the trackers for the entire state, position, and velocity, respectively. The results confirm the intuition of the plots. The $SE_2(3)$ body tracker is more consistent than the other trackers. The naive body tracker also outperforms the naive world tracker. The increased uncertainty of the body filters make them consistent. The naive trackers are very inconsistent and greatly underestimate the covariance of their state estimates, as shown by the low consistency scores. Interestingly, all trackers achieve consistent velocity estimates.

	Total state		Position		Velocity	
	$SE_2(3)$	ESKF	$SE_2(3)$	ESKF	$SE_2(3)$	ESKF
World	71.0%	54.9%	78.3%	64.1%	94.3%	92.6%
Body	91.1%	80.0%	91.3%	79.6%	94.7%	93.0%
Naive world	28.0%	19.4%	37.0%	27.7%	87.2%	83.8%
Naive body	30.8%	21.7%	37.5%	27.6%	92.4%	90.4%

Table 12: The percentage of NEES within the 95% confidence bounds for the total state, position, and velocity of all trackers over 100 Monte Carlo runs.

As with the inertial navigation filters, the NEESs of the trackers are also computed for the simulations ignoring the first 5 seconds. Ignoring the transient effects, giving the filters time to converge to the true platform state, should shrink the gap between the $SE_2(3)$ and ESKF trackers. This should improve the accuracy and shrink the covariance estimates of the trackers. In the calculation of the NEESs these two effects cancel. This means that the NEES including the transient period and the NEES of only the following "converged" period should be similar if both periods are captured correctly in the tracking estimates. Table 13 report the scores of these "converged" NEESs. Compared to the entire simulation, the two inertial navigation filters are slightly more equal. The SE_2 body filter scores above 95%. This indicates that in this scenario, the correlation free $SE_2(3)$ and body tracker models the true problem well. That the consistency increases when ignoring the transient period is expected. The filters are more inconsistent in this period, something that influences the trackers. The naive filters perform about the same with and without the

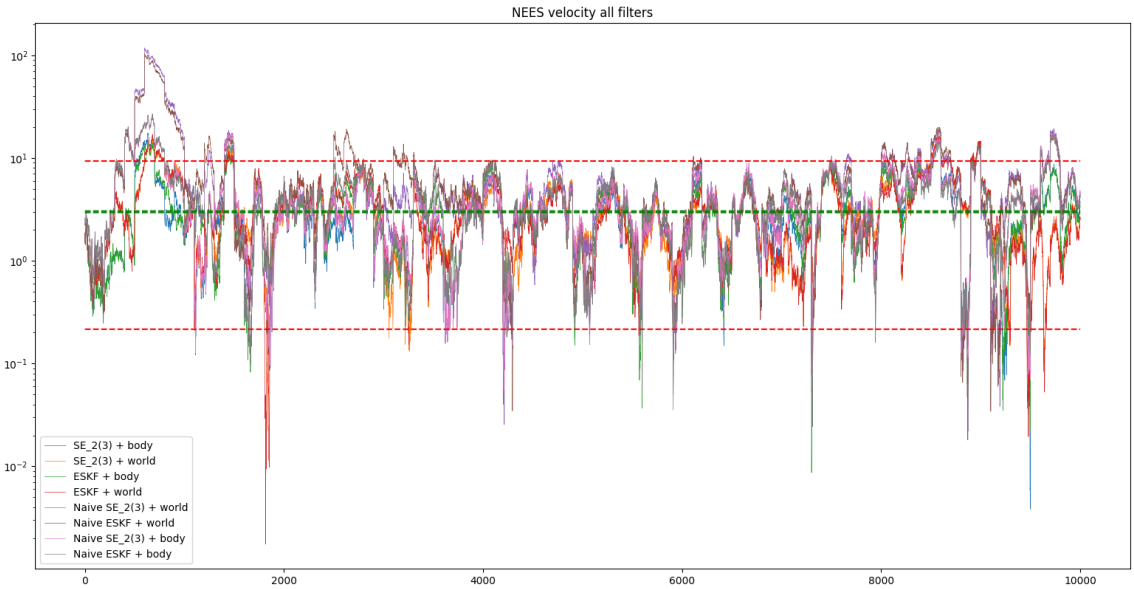


Figure 29: The NEES of the velocity estimates of all trackers.

transient period consistency-wise. Even ignoring the transient period, both the world filters show consistency issues.

In applications using data assosication to pair targets and measurements, distributions over the target states are important for proper functioning. As target measurements are defined in the body frame of the platform, it is convenient if the targets are parameterized in the body frame as well. The alternative is to convert the measurements to the world frame, injecting the platform covariance in the measurement uncertainty. In this work, the performance and consistency of the trackers in the body frame of the platform have not been explored. Converting the world trackers to the body frame requires an injection of the platform covariance in the target uncertainty equivalent to the converse conversion. In [6] the inflated covariance of a tracker converted to the other frame was shown to have better consistency properties. The "extra" uncertainty injected in the conversion can, in the consistency metric, compensate for errors caused by the lack of modeling all dependencies in the problem.

With the combination of decent RMSE performance and consistency properties, the $SE_2(3)$ body tracker seems to be the best performing filter. Generally, the $SE_2(3)$ based trackers are more accurate and consistent than their respective ESKF based counterparts. Considering the importance of the inertial navigation filter for tracker performance, the choice of $SE_2(3)$ over the popular ESKF is reasonable. This is especially the case when the noise characteristics are dominated by the initial orientation noise, or noise on the gyro. In these cases the $SE_2(3)$ captures the error dynamics of the inertial navigation significantly better than the ESKF.

The performance of the body filter is an indication that using a manifold approach to the correlated tracking problem is a good idea, as the error is then in the body frame of the platform. Combining this with the more accurate approach of correlated filters should improve the RMSE and NEES

	$SE_2(3)$	ESKF
World	73.4%	60.2%
Body	95.5%	89.0%
Naive world	26.6%	19.6%
Naive body	28.0%	20.8%

Table 13: The percentage of NEES within the 95% confidence bounds for the entire state of all trackers over 100 Monte Carlo runs when skipping the first 5 seconds of the tracking in the evaluation.

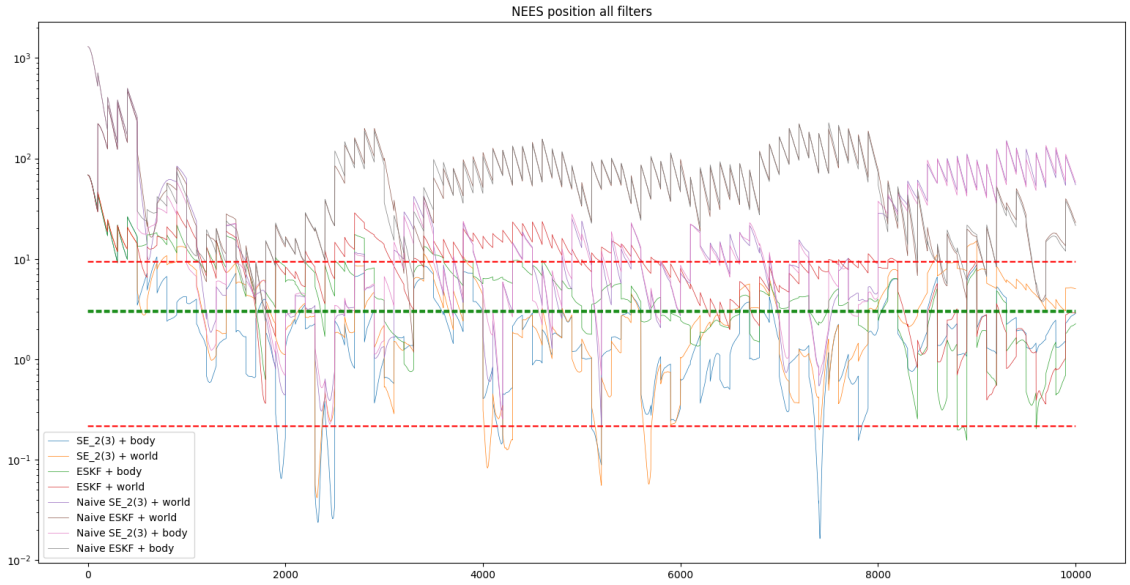


Figure 30: The NEES of the position estimates of all trackers.

scores, as dependencies ignored in the correlation-free trackers are modeled. A correlated manifold approach also makes the conversion from body to world frame implicit, since it is done in the error injection during the update step run after a target measurement is received. This makes the process of converting from the body frame to the world frame superfluous, avoiding the extra covariance injected in the estimates in the conversion.

9 Future research

The aim of this work was to unify the theory of target tracking and inertial navigation from a Lie-theoretic point of view. The performance of the developed trackers proved decent, but still leaves much to be desired. It was the authors intention to implement some correlated filters, but there was not enough time. In this section, a few points for future development are highlighted.

As mentioned, correlated filters are an obvious next step going forward from what is done in this work. Correlated filters should offer a leap in performance compared to the uncorrelated versions. Assuming the estimated pose of targets is independent from the platform they are observed from is not obviously reasonable. The manifold approach offered by Lie theory lends itself to being extended with target states. Some thoughts on this implementation can be found in Section 5.3.

The implementation in this thesis has the potential for massive performance gains. An immediately available technique is preintegration of IMU measurements. In [9] the required theory is developed. This does involve dropping the fourth-order contribution in the propagation step, but this should not affect the properties of the filter significantly.

A potentially interesting application of this work is collaboration between many platforms. [23] describes how many platforms can collaborate to estimate targets and themselves with higher accuracy on Exponentially distributed platforms. Having platforms collaborate is practical for better accuracy, but also to overcome challenges such as target occlusion. Having platforms that seamlessly collaborate with nearby platforms is a great advantage for autonomous navigation.

A few other obvious extensions of this work exist. IMUs should be paired with a bias estimation scheme, as they often suffer from slowly drifting bias. In reality, the sensor frames and the body frames do not coincide. Therefore, lever arm compensation for the GNSS sensor and offset compensation for the IMU should also be implemented. The targets are assumed to maneuver according to a CV model. In practice, this might not be the case. Including the interacting multiple models algorithm allows for defining other possible motion models for the target, and having the tracker estimate the most probable state based on all the models.

In this work, the origin of the target measurements is assumed to be known. This requires some kind of recognition scheme and sensor, e.g. a camera. For many sensors, this is not the case. If measurements are taken using a radar, it is impossible to be certain from which target a measurement originates. In this case, the trackers must be paired with a data association algorithm, such as the JIPDA.[2], [26] The JIPDA assumes Gaussian distributions over the target states. The JIPDA therefore requires some work to work with Exponential Gaussian distributions, but the gut feeling of the author is that this should be possible, either through linearizations, or having the JIPDA work on the Exponential coordinates. Two techniques that have proved one and the same multiple times in this work. Care must be taken so that false tracks and erroneous data associations do not affect the platform estimate.

10 Conclusion

A tracker compensating for platform movement was developed using a Lie-theoretic approach for both body and world parameterization of targets. The integration of these trackers with an inertial navigation filter on the Lie group $SE_2(3)$ demonstrated significant improvements on the RMSE and NEES metrics compared to the popular ESKF filter, and naive implementations that do not account for inertial navigation uncertainty. In total, eight tracking architectures were explored. The body parameterized trackers were shown to perform better than their world parameterized counterparts. The better performance of the body trackers on the RMSE was theorized to be due to the implicit correction of the body estimates caused by the platform update step.

To work with the Exponential Gaussian distribution over the platform state, a derivation of compatible motion and measurement models was required. Furthermore, a novel approach was presented to convert the target estimates from the body frame of the platform to the world frame, using the tangent planes of the $SE_2(3)$ group. The method preserves the world frame "banana"-shape of the Gaussian distributions parameterized in the body frame of the platform. This is not the case for the more obvious linearization approach to the conversion.

As the work aimed at laying a foundation for tracking theory from a moving platform with a Lie-theoretic approach, the building blocks for the joint inertial navigation and tracking problem were presented. The implementation of correlated trackers was left for future work. Regardless, the good performance of the body trackers hint at the possible performance gain of correlated trackers, as a Lie group approach will make the error parameterized in the body frame of the platform. To this end, some possible manifolds and tracking architectures for the correlated filters were presented, indicating a path for future research.

Appendix

A Fourth order covariance contribution

The fourth order term of Equation 59 is defined in [9]. To help with the definition of $\boldsymbol{S}_{4\text{th}}$, two operators are introduced.

$$\begin{aligned} \ll \boldsymbol{A} \gg &:= -\text{trace}(\boldsymbol{A})\boldsymbol{I}_3 + \boldsymbol{A} \\ \ll \boldsymbol{A}, \boldsymbol{B} \gg &:= \ll \boldsymbol{A} \gg \ll \boldsymbol{B} \gg + \ll \boldsymbol{B}\boldsymbol{A} \gg \end{aligned}$$

Let $\boldsymbol{\Sigma} = \text{Ad}_{\hat{\boldsymbol{\Delta}}_{\delta t}^{-1}} \boldsymbol{F}_{\delta t} \boldsymbol{\Sigma}_k (\text{Ad}_{\hat{\boldsymbol{\Delta}}_{\delta t}^{-1}} \boldsymbol{F}_{\delta t})^\top$, which is the first term of the covariance in Equation 59. The fourth order covariance contribution is then given by

$$\boldsymbol{S}_{4\text{th}} = \frac{1}{12}\left(\boldsymbol{A}_{\boldsymbol{\Sigma}}\boldsymbol{Q} + \boldsymbol{Q}\boldsymbol{A}_{\boldsymbol{\Sigma}}^T + \boldsymbol{A}_{\boldsymbol{Q}}\boldsymbol{\Sigma} + \boldsymbol{\Sigma}\boldsymbol{A}_{\boldsymbol{Q}}^T\right) + \frac{1}{4}\boldsymbol{B}, \tag{99}$$

where $\boldsymbol{Q}$ is the covariance of the noise $\boldsymbol{\vartheta}$ and

$$\begin{aligned} \boldsymbol{A}_{\boldsymbol{\Sigma}} &= \begin{bmatrix} \ll \boldsymbol{\Sigma}_{\phi\phi} \gg & \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} \\ \ll \boldsymbol{\Sigma}_{\nu\phi} + \boldsymbol{\Sigma}_{\phi\nu} \gg & \ll \boldsymbol{\Sigma}_{\phi\phi} \gg & \boldsymbol{0}_{3\times3} \\ \ll \boldsymbol{\Sigma}_{\rho\phi} + \boldsymbol{\Sigma}_{\phi\rho} \gg & \boldsymbol{0}_{3\times3} & \ll \boldsymbol{\Sigma}_{\phi\phi} \gg, \end{bmatrix}, \\ \boldsymbol{A}_{\boldsymbol{Q}} &= \begin{bmatrix} \ll \boldsymbol{Q}_{\phi\phi} \gg & \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} \\ \ll \boldsymbol{Q}_{\nu\phi} + \boldsymbol{Q}_{\phi\nu} \gg & \ll \boldsymbol{Q}_{\phi\phi} \gg & \boldsymbol{0}_{3\times3} \\ \ll \boldsymbol{Q}_{\rho\phi} + \boldsymbol{Q}_{\phi\rho} \gg & \boldsymbol{0}_{3\times3} & \ll \boldsymbol{Q}_{\phi\phi} \gg \end{bmatrix}, \\ \boldsymbol{B} &= \begin{bmatrix} \boldsymbol{B}_{\phi\phi} & \boldsymbol{B}_{\nu\phi}^\top & \boldsymbol{B}_{\rho\phi}^\top \\ \boldsymbol{B}_{\nu\phi} & \boldsymbol{B}_{\nu\nu} & \boldsymbol{B}_{\nu\rho} \\ \boldsymbol{B}_{\rho\phi} & \boldsymbol{B}_{\nu\rho}^\top & \boldsymbol{B}_{\rho\rho} \end{bmatrix}, \end{aligned}$$

with

$$\begin{aligned} \boldsymbol{B}_{\phi\phi} =& \ll \boldsymbol{\Sigma}_{\phi\phi}, \boldsymbol{Q}_{\phi\phi} \gg, \\ \boldsymbol{B}_{\nu\phi} = \boldsymbol{B}_{\phi\nu}^T =& \ll \boldsymbol{\Sigma}_{\phi\phi}, \boldsymbol{Q}_{\phi\nu} \gg + \ll \boldsymbol{\Sigma}_{\nu\phi}, \boldsymbol{Q}_{\phi\phi} \gg, \\ \boldsymbol{B}_{\rho\phi} = \boldsymbol{B}_{\phi\rho}^T =& \ll \boldsymbol{\Sigma}_{\phi\phi}, \boldsymbol{Q}_{\phi\rho} \gg + \ll \boldsymbol{\Sigma}_{\rho\phi}, \boldsymbol{Q}_{\phi\phi} \gg, \\ \boldsymbol{B}_{\nu\nu} =& \ll \boldsymbol{\Sigma}_{\phi\phi}, \boldsymbol{Q}_{\nu\nu} \gg + \ll \boldsymbol{\Sigma}_{\phi\nu}, \boldsymbol{Q}_{\nu\phi} \gg \\ &+ \ll \boldsymbol{\Sigma}_{\nu\phi}, \boldsymbol{Q}_{\nu\phi} \gg + \ll \boldsymbol{\Sigma}_{\nu\nu}, \boldsymbol{Q}_{\phi\phi} \gg, \\ \boldsymbol{B}_{\nu\rho} = \boldsymbol{B}_{\rho\nu}^T =& \ll \boldsymbol{\Sigma}_{\nu\rho}, \boldsymbol{Q}_{\phi\phi} \gg + \ll \boldsymbol{\Sigma}_{\phi\rho}, \boldsymbol{Q}_{\nu\phi} \gg \\ &+ \ll \boldsymbol{\Sigma}_{\nu\phi}, \boldsymbol{Q}_{\rho\phi} \gg + \ll \boldsymbol{\Sigma}_{\phi\phi}, \boldsymbol{Q}_{\nu\rho} \gg, \\ \boldsymbol{B}_{\rho\rho} =& \ll \boldsymbol{\Sigma}_{\phi\phi}, \boldsymbol{Q}_{\rho\rho} \gg + \ll \boldsymbol{\Sigma}_{\phi\rho}, \boldsymbol{Q}_{\rho\phi} \gg \\ &+ \ll \boldsymbol{\Sigma}_{\rho\phi}, \boldsymbol{Q}_{\phi\rho} \gg + \ll \boldsymbol{\Sigma}_{\rho\rho}, \boldsymbol{Q}_{\phi\phi} \gg . \end{aligned}$$

Bibliography

[1] D. Reid, 'An algorithm for tracking multiple targets', *IEEE Transactions on Automatic Control*, vol. 24, no. 6, pp. 843–854, 1979. DOI: 10.1109/TAC.1979.1102177.

[2] E. F. Brekke, A. G. Hem and L.-C. N. Tokle, 'Multitarget tracking with multiple models and visibility: Derivation and verification on maritime radar data', *IEEE Journal of Oceanic Engineering*, vol. 46, no. 4, pp. 1272–1287, 2021. DOI: 10.1109/JOE.2021.3081174.

[3] C.-C. Wang, C. Thorpe, S. Thrun, M. Hebert and H. Durrant-Whyte, 'Simultaneous localization, mapping and moving object tracking', *The International Journal of Robotics Research*, vol. 26, no. 9, pp. 889–916, 2007. DOI: 10.1177/0278364907081229. eprint: https://doi.org/10.1177/0278364907081229. [Online]. Available: https://doi.org/10.1177/0278364907081229.

[4] J. Sola, 'Towards Visual Localization, Mapping and Moving Objects Tracking by a Mobile Robot', Theses, Institut National Polytechnique (Toulouse), Feb. 2007. [Online]. Available: https://hal.science/tel-04596530.

[5] S. J. Julier and A. Gning, 'Bernoulli filtering on a moving platform', in *2015 18th International Conference on Information Fusion (Fusion)*, 2015, pp. 1511–1518.

[6] E. Brekke and E. Wilthil, 'Suboptimal kalman filters for target tracking with navigation uncertainty in one dimension', Mar. 2017, pp. 1–11. DOI: 10.1109/AERO.2017.7943601.

[7] R. Novoselov, S. Herman, S. Gadaleta and A. Poore, 'Mitigating the effects of residual biases with schmidt-kalman filtering', in *2005 7th International Conference on Information Fusion*, vol. 1, 2005, 8 pp. DOI: 10.1109/ICIF.2005.1591877.

[8] J. Solà, *Quaternion kinematics for the error-state kalman filter*, 2017. arXiv: 1711.02508 [cs.RO].

[9] M. Brossard, A. Barrau, P. Chauchat and S. Bonnabel, *Associating uncertainty to extended poses for on lie group imu preintegration with rotating earth*, 2021. arXiv: 2007.14097 [cs.RO].

[10] S. Bonnabel, 'Left-invariant extended kalman filter and attitude estimation', in *2007 46th IEEE Conference on Decision and Control*, 2007, pp. 1027–1032. DOI: 10.1109/CDC.2007.4434662.

[11] A. Barrau and S. Bonnabel, 'The invariant extended kalman filter as a stable observer', *IEEE Transactions on Automatic Control*, vol. 62, no. 4, pp. 1797–1812, 2017. DOI: 10.1109/TAC.2016.2594085.

[12] J. Solà, J. Deray and D. Atchuthan, *A micro lie theory for state estimation in robotics*, Licensed under CC BY-NC-SA 4.0, https://creativecommons.org/licenses/by-nc-sa/4.0/, 2021. arXiv: 1812.01537 [cs.RO].

[13] Wikipedia contributors, *Euler's rotation theorem — Wikipedia, the free encyclopedia*, [Online; accessed 29-March-2024], 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Euler%27s_rotation_theorem&oldid=1215740868.

[14] E. F. Brekke, *Fundamentals of Sensor Fusion*. n.p, 2023, Lecture notes in course TTK4250 Sensor Fusion at NTNU. Can be obtained by contacting the author.

[15] Wikipedia contributors, *Group (mathematics) — Wikipedia, the free encyclopedia*, https://en.wikipedia.org/w/index.php?title=Group_(mathematics)&oldid=1221104259, [Online; accessed 21-May-2024], 2024.

[16] T. Lindstrøm and K. Hveberg, *Flervariabel analyse med lineær algebra*. Gyldendal akademiske, 2015.

[17] Wikipedia contributors, *Rodrigues' rotation formula — Wikipedia, the free encyclopedia*, https://en.wikipedia.org/w/index.php?title=Rodrigues%27_rotation_formula&oldid=1224740055, [Online; accessed 21-May-2024], 2024.

[18] Wikipedia contributors, *Group action — Wikipedia, the free encyclopedia*, https://en.wikipedia.org/w/index.php?title=Group_action&oldid=1211636850, [Online; accessed 30-March-2024], 2024.

[19] E. Eade, *Lie groups for 2d and 3d transformations*, https://ethaneade.com/lie.pdf, [Online; accessed 05-April-2024], 2017.

[20] T. Barfoot and P. Furgale, 'Associating uncertainty with three-dimensional poses for use in estimation problems', *Robotics, IEEE Transactions on*, vol. 30, pp. 679–693, Jun. 2014. DOI: 10.1109/TRO.2014.2298059.

[21] G. Bourmaud, R. Mégret, M. Arnaudon and A. Giremus, 'Continuous-discrete extended kalman filter on matrix lie groups using concentrated gaussian distributions', *Journal of Mathematical Imaging and Vision*, 2015.

[22] C. Forster, L. Carlone, F. Dellaert and D. Scaramuzza, 'On-manifold preintegration for real-time visual–inertial odometry', *IEEE Transactions on Robotics*, vol. 33, no. 1, pp. 1–21, 2017. DOI: 10.1109/TRO.2016.2597321.

[23] A. W. Long, K. C. Wolfe, M. Mashner and G. S. Chirikjian, 'The banana distribution is gaussian: A localization study with exponential coordinates', in *Robotics: Science and Systems*, 2012. [Online]. Available: https://www.roboticsproceedings.org/rss08/p34.pdf.

[24] Wikipedia contributors, *Baker–campbell–hausdorff formula — Wikipedia, the free encyclopedia*, https://en.wikipedia.org/w/index.php?title=Baker%E2%80%93Campbell%E2%80%93Hausdorff_formula, [Online; accessed 10-June-2024], 2023.

[25] H. Masnadi-Shirazi, A. Masnadi-Shirazi and M.-A. Dastgheib, *A step by step mathematical derivation and tutorial on kalman filters*, 2019. arXiv: 1910.03558 [stat.OT].

[26] E. Hjermstad, E. F. Brekke and L.-C. N. Tokle, *Towards accurate multi-target tracking*, https://github.com/erlingh99/prosjektoppgave/blob/master/Prosjektoppgave.pdf, [Written by the Author as a semester project in the fall/winter 2023], 2023.

[27] E. F. Brekke, 'Assignment 1', Lecture notes in course TK8102 Nonlinear state estimation at NTNU. Can be obtained by contacting the author., 2020.

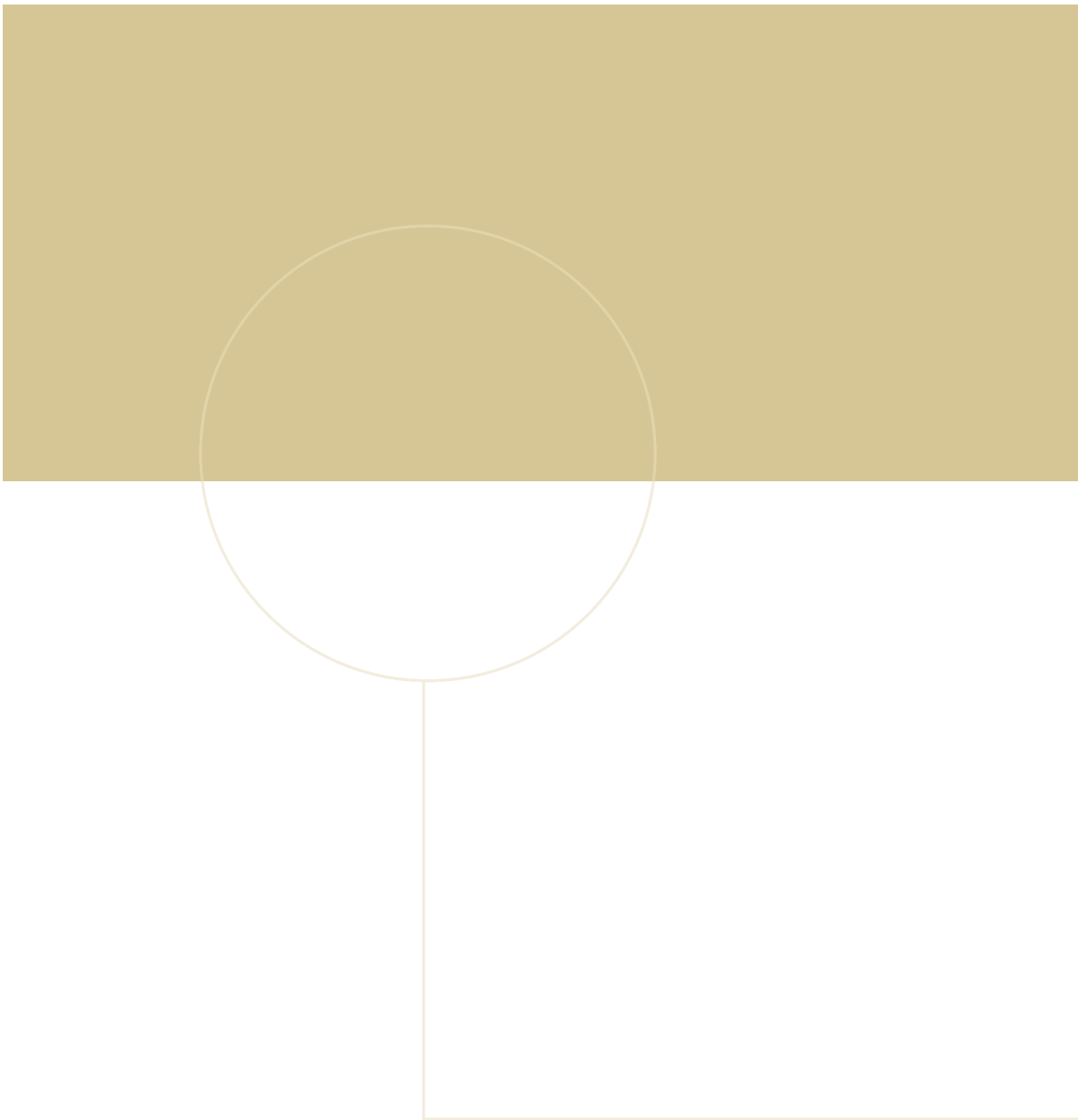
[28] N. F. Toda, J. L. Heiss and F. H. Schlee, 'Spars: The system, algorithm, and test results', *Robotics: Science and Systems, Aerospace Corp*, vol. 1, pp. 361–370, 1969. [Online]. Available: https://malcolmdshuster.com/HTF_Toda-Heiss-Schlee_SPARS_1969_FLM.pdf.

[29] J. G. Mangelson, M. Ghaffari, R. Vasudevan and R. M. Eustice, 'Characterizing the uncertainty of jointly distributed poses in the lie algebra', *IEEE Transactions on Robotics*, vol. 36, no. 5, pp. 1371–1388, 2020. DOI: 10.1109/TRO.2020.2994457.

[30] Wikipedia contributors, *N-sphere — Wikipedia, the free encyclopedia*, https://en.wikipedia.org/w/index.php?title=N-sphere&oldid=1222162211, [Online; accessed 21-May-2024], 2024.

[31] T. V. Haavardsholm, 'Lie theory in practice', Lecture notes in course TTK21 Introduction to Visual Simultaneous Localization and Mapping - VSLAM at NTNU., 2023.

[32] Wikipedia contributors, *Matrix norm — Wikipedia, the free encyclopedia*, https://en.wikipedia.org/wiki/Matrix_norm#Frobenius_norm, 2023.



NTNU

Norwegian University of
Science and Technology