Grant agreement no: FP7-610603

European Robotic Pedestrian Assistant 2.0 (EUROPA2)

Start of the project: 01.10.2013

Duration: 3 years

DELIVERABLE 5.1(i)

Life-Long Self-Calibration and Fault Detection

Due date: month 12 (September 2014)

Lead contractor organization: ETHZ

Dissemination Level: PUBLIC

Contents

1 Introduction

Within the EUROPA2 project, one major focus is life-long operation in urban environments (WP5). One part is the capability to continuously self-calibrate the robot sensors and detect failures during operation. This includes slow changes in the sensors' intrinsic or extrinsic calibration parameters as well as the detection of sensor failure (task 5.1).

This document is the intermediate deliverable D5.1(i) on the life-long self-calibration approach for the EUROPA robot and presents its status after project month 12. The final deliverable D5.1 is due after project month 30.

The original plan, as formulated in Annex I, was to first implement a state of the art but offline and in-laboratory calibration system; recorded sensor data from a well known and specifically modified environment will be post processed on an external system to retrieve calibration information with less time constraints than an online system would have. After that, we would implement an online and real time system able to calibrate in an unknown environment and compare its performance with the offline system.

Currently, the developments in the context of D5.1(i) deviated from the plan Annex I because we started with an approach that is both an online and offline system - mostly depending on input parameters. Furthermore the delay in hardware specification would have meant wasted effort when going too much into details concerning these sensors. Thus we are less far as originally planned with the offline system but are ahead of the plan for the online system.

The document is divided in two major parts. First, a description of the EUROPA2 calibration problem and second, our approach and progress to solve it.

2 The EUROPA2 calibration problem

The EUROPA2 platform has many different sensors, as depicted in Figure 1. This makes the full calibration problem a complex task. We will detail in the following on three major challenges of the general calibration problem that are crucial to tackle for both an accurate offline, in-laboratory and online self-calibration of the EUROPA2 platform.

- Sensor time delay calibration
- Observability aware calibration
- Sensor data associations

We spent most of our research activity in the last 12 months on the first two topics.

2.1 Continuous time estimation to estimate time delays

One common loss of accuracy is due to unknown time delays of sensor readings. Combining information of different sensors requires precise knowledge of the relative timing of those readings to achieve high accuracy [12]. The faster the world's sensed properties relative to the sensor are changing the more important this gets.

Sensor readings are often acquired over unreliable channels (for example Ethernet) introducing unknown and random delays in the range of milliseconds. This renders the time of receiving too

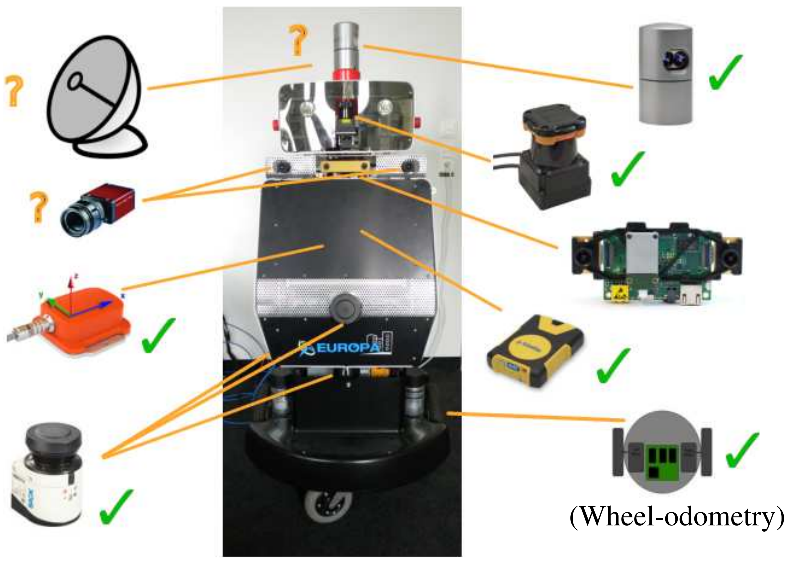


Figure 1: EUROPA2 sensor setup. The green checks indicate which sensor is already running in the EUROPA2 configuration on the ETHZ platform. The question marks indicate sensors or sensor placement, which have not been specified yet.

inaccurate, especially for example for sweeping sensors or high rate sensors like an IMU. This fundamental problem has led to the introduction of hardware timestamps committed by the sensors alongside their measurements based on an internal clock. This drastically reduces the problem of random delays. But the internal clocks of different sensors are typically not synchronized. Therefore problems of unknown time offsets and clock drift remain unsolved. In both cases this problem introduces important calibrations parameters to accurately associate sensor timestamps.

All the discrete-time state representations have problems with estimating such parameters, as associating measurements with a certain time is part of their core concept. For that reason we decided to develop continuous-time state representations; the state trajectory is represented by a parametric family of differentiable functions into the state space rather than by a discrete time series of states. This method has been demonstrated to allow accurate estimation of time delays [3]. While these families are well known and often easy to implement for state space components of Euclidean spaces, great difficulties arise in non-Euclidean cases, especially for higher (> 1) degree of differentiability. The most important example in robotics is the Lie group SO(3), which is also a crucial component to model the attitude and extrinsic calibration of the EUROPA2 platform.

2.2 Observability aware calibration

When attempting to estimate the state and calibration parameters of a robot during normal operation, some directions in the state or calibration space may be unobservable. Informally, this means that there is not enough information in the data to correctly estimate these parameters. For example, when two cameras do not share an overlapping field of view, planar motion renders the extrinsic calibration problem degenerate; the transformation between cameras only becomes observable under general 3D motion. This can become an important issue because unobservable or barely observable directions may appear well observable in the presence of noise. Even if our hypothetical platform is piloted only on a plane (a degenerate case), noise in the measurements can make unobservable parameters appear observable. We call these parameters numerically unobservable to mirror the related concept of numerically rank-deficient matrices. This issue has been largely ignored by the community. To the best of our knowledge, [11], is the first published self-calibration algorithm able to cope with this issue.

2.3 Sensor data associations

A very different kind of challenge is to associate the sensor readings. Either sensors with themselves but from different times (self associations) or across sensors. The cross-sensor data associations are fundamental to be able to automatically determine the extrinsic calibration of sensors without relying on movement of the platform. The self associations over time are essential to observe the sensors movement relative to an environment assumed to be mostly static. For a complete extrinsic calibration it is necessary to have potential associations between the sensors such that all sensors are (indirectly) connected.

In the associations lies the big difference between self-calibration and in-laboratory calibration. In-laboratory calibration is typically based on specific markers in the environment that are easy to accurately detect and localize in the spatial sensor readings. While self-calibration needs to work in the unmodified operation environment and therefor needs to find association in a given environment instead of by exploiting known features for well chosen markers.

The availability of good concepts and algorithms concerning sensor association in operation environments is very dependent on the pair of sensors types. Best studied is camera-camera followed by LIDAR-LIDAR. All the cross-sensor data associations are more difficult and in general less well solved. This is especially true for Radar sensors. To the best of our knowledge there is no publicly available research about Radar to any other sensor extrinsic calibration.

3 Our Approach

In the following we will explain several parts of our approach in terms of role and progress with respect to offline and online self-calibration for the EUROPA2 platform.

3.1 Sensor data associations for self-calibration

Our current concept concerning sensor data associations for EUROPA2 self-calibration is as depicted in Figure 2. To retrieve and quantify associations for the connecting edges we plan to

1. build on the mutual information (MI) maximization concept from [14],
2. use Iterative Closest Point (ICP) based point to point associations (section 3.6),
3. use Binary robust invariant scalable keypoints (BRISK) features ([8]),
4. think of a new solution - there seems to be no state of the art for these to problems and small scale Radars,
5. extend the MI concept of [14] if the sensor output turns out to be suitable for that.

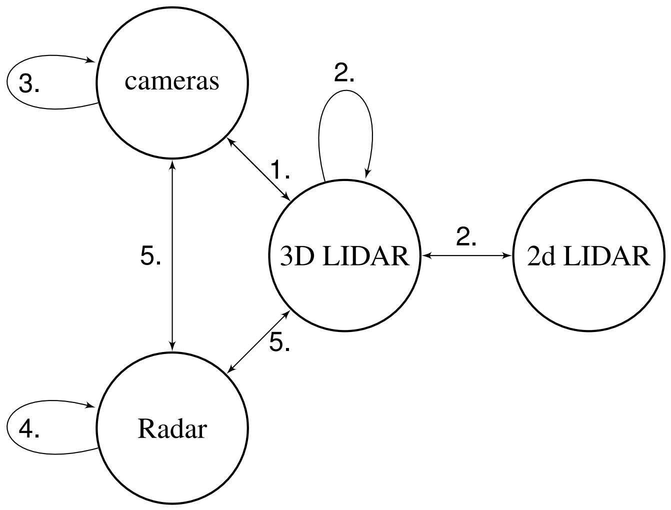


Figure 2: Sensor data associations for the EUROPA platform. The selected associations are listed in section 2.3

For the external calibration of the IMU and Odometry relative to the 3D sensors stereo camera, 3D LIDAR and Radar we will build on the fact that all of them are able to (partially) observe the 3D motion of the platform due to there self associations. The assumption that they all observe their own movement while being rigidly connected yields the possibility to infer (partially) the extrinsic calibration.

3.2 In-laboratory offline calibration

The status of offline in-laboratory calibration is as follows:

- GPS: we rely on factory calibration for intrinsics and won't need extrinsic
- IMU intrinsic: we rely on factory calibration
- Cameras intrinsic and extrinsic: our calibration tool, Kalibr, already ready to use; blocked due to missing final camera specification (see section 3.5)
- LIDARs' intrinsic: We plan to build on the ideas of [16] and [1] for the tilting Hokuyo
- Radar intrinsic: Apparently no state of the art publicly available. We postponed thinking of a new technique until the device is specified and the data demands intrinsic calibration
- Extrinsic calibration (including time delays) and wheel odometry intrinsics: We will use a batch maximum likelihood estimator for the calibration parameters jointly with a time continuous state trajectory combined with highly accurate external position tracks of the platform and sensor specific markers provided by a Vicon system. This allows precise offline calibration while keeping the redundant effort for the online implementation low.

3.3 Estimating time delays using Lie-group B-splines

Our solution to the problem of time continuous estimation (section 2.1) for non-vector-space variables is to exploit the recent ideas to generalize the concept of B-splines to Lie groups [7]. This idea allows parametric representation of $SO(3)$-trajectories that won't "fall off" the manifold or run into singularities. Furthermore it allows to choose the minimal degree of differentiability (in time) over the whole trajectory. This is especially important for physical modeling where velocities and accelerations need to be available for the state trajectories. We developed the necessary analytical expressions to employ those generalized B-splines within nonlinear optimization relying on first order derivatives, like Gauss-Newton or Levenberg-Marquardt and we evaluated the accuracy of that approach for attitude estimation. The only work we are aware of employing this for state estimation is [10]. But they use scalar based auto differentiation (part of Google's Ceres) to retrieve the necessary Jacobians, which is too computational intensive for our goal of real-time use. We further extended our code base to allow estimation of unknown time delays in batch estimations using these B-splines to represent the state space trajectories. Our preliminary work, completed before the EUROPA2 project started, [4] and [15] uses traditional B-splines to represent a minimal parameterization of $SO(3)$ and is therefore limited by not being rotation invariant.

Further details can be found in the attached paper draft "Continuous-Time Estimation of attitude using B-splines on Lie groups"

3.4 Observability aware and incremental self-calibration

To tackle the problem of seemingly observable calibration / state space directions we build upon and improve our former work as published in [11]. A succeeding journal paper, partially founded by the EUROPA2 project, is currently under review and attached to this document. Our approach exploits the algebraic links between the Gauss-Newton algorithm, the Fisher information matrix, and nonlinear observability analysis to automatically detect directions in parameter space that are numerically unobservable and avoid updating our parameters in these directions; at any given time, directions in the parameter space that are observable will be updated based on the latest information, while unobservable directions will remain at their initial guess.

Furthermore measurements containing novel information about the calibration parameters are detected using an information gain test and added to a working set that is used to estimate the parameters. The result is an algorithm that listens to an incoming stream of data, automatically accumulating a batch of data that may be used to calibrate a full robot system.

Thanks to the these properties we are already well prepared for lifelong self-calibration as redundant information gets thrown away instead of constantly increasing the computational demands of the calibration algorithm. We plan to also exploit this property to do fault detection.

For further details behind the algorithm see the attached paper "Online Self-Calibration for Robotic Systems".

3.5 Kalibr = Camera + IMU calibration

Kalibr is a collection of tools that we developed for calibrating multi-camera systems as well as sensor units comprised of one or multiple cameras and an IMU. Similarly to the well established camera calibration toolbox by Jean-Yves Bouguet [2], it allows for intrinsic and extrinsic calibration of camera systems. Kalibr extends this functionality by supporting both N-camera systems as well as different

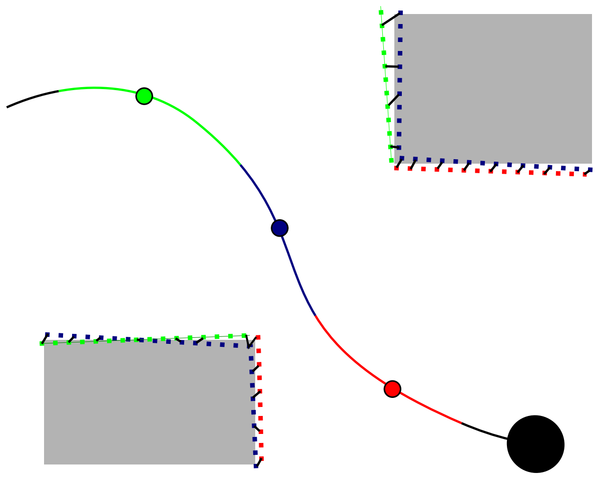


Figure 3: A top view illustration of the grouping (illustrated via coloring) of LIDAR point measurements base on there timestamp along the trajectory of a moving platform (big black dot). The small colored rectangles represent the LIDAR measurements taken from the correspondingly colored segments of the trajectory. The black association lines between them (found using ICP) are used in a batch optimization step to introduce error terms for the state trajectory and calibration parameters minimizing the point-to-plane distance between measurements of different LIDARs and from different time segments.

projection and distortion models, which makes it well suited for both pinhole and unified projection cameras using the projection model from [13] as well as for heterogeneous systems, where both types of cameras are present. The ROS camera calibration toolbox [5] introduced a heuristic for selecting a set of dissimilar views of the calibration target from a stream of images to reduce redundancy in the calibration data and improve results. Kalibr takes this approach a step further by incorporating a more theoretically grounded, information theoretical measure by using our incremental calibration tools (section 3.4) for novel view selection.

In contrast to other EKF-based calibration methods [6, 9], the camera/IMU calibration tool also employs our continuous-time batch estimation framework and simultaneously estimates both the rigid transformation between the cameras and the IMU as well as the temporal offset between these sensors. In previous research [3], this method has proven to accurately estimate time offsets of a fraction of the sampling time and emphasized the importance of accounting for exposure periods when timestamping camera images.

3.6 Novel extrinsic batch self-calibration algorithm for LIDARs

Our LIDAR-LIDAR, IMU, and odometry extrinsic calibration concept was specifically developed for the EUROPA2 platform and is planned to be stepwise extended up to the full featured EUROPA2 calibration system. The big problem of spinning LIDAR measurements, especially for multi-beam LIDAR like the Velodyne 32E, is the fact that every point of the many points per second (700k/s for the latter) is measured at a different time. So in theory a moving platform has a different pose for each laser point. An exact ML estimator jointly estimating the platform's trajectory is thus still intractable, especially for the goal of online capability. Our approximation concept is an iterative batch

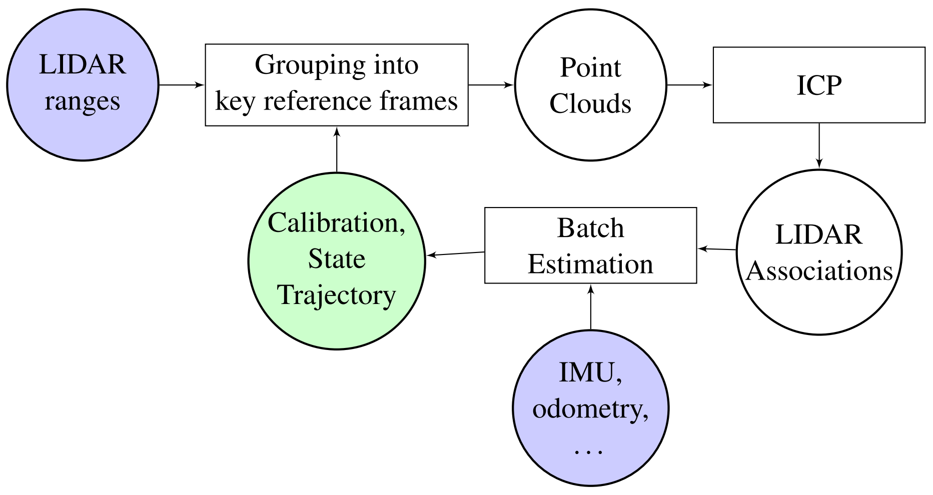


Figure 4: Sketch of the algorithm of iterative refinement of LIDAR-LIDAR associations to jointly calibrate IMU, odometry and the LIDARs

optimization. We subsampled consecutive LIDAR measurements over short duration (about a second each) into consecutive point clouds, as depicted in Figure 3, taking a current best guess about the platforms pose into account to transform all the measurement of a group into a single key-reference-frame. The current best guess is initialized based on IMU and odometry only. Between these point clouds we search for corresponding points using ICP by minimizing point-to-plain distances. These associations are fed into our continuous time batch estimator, taking IMU and odometry measurements into account. The resulting improved trajectory is used as the next best guess to recalculate the point clouds and point to point associations to be fed to the batch estimator. This process is repeated until convergence. This outer loop is illustrated in Figure 4.

This way we can calibrate for the LIDAR-LIDAR-IMU-Odometry extrinsic calibration and the odometry intrinsic calibration (wheel diameters and distance). This system is already functional and works quite well except for the unforeseen problem of the shaking in the passive base joint. See section 3.6.1 for an explanation. However, we still believe in this method to be suitable and extendible with further sensor data associations up to the full featured online self extrinsic and intrinsic calibration system.

3.6.1 Calibration of the passive base joint

There is a difficulty concerning the extrinsic calibration of the odometry with respect to the rest of the system. The EUROPA platform has a passive revolute joint in its base allowing its body to shake back and forth without moving or lifting the main wheels and the front caster wheel. The elasticity of the back caster wheel is high enough to allow significant shaking in this joint. This motion is visible to observers, even in the case of moderate movements. It is not negligible for accurate state estimation as the position of this joint is essential for the relative pose of the odometry frame to all the other sensor frames. Unfortunately, the dynamics of this joint are hard to model accurately due to their highly nonlinear behavior, lack of specification, and weak observability, as the only sensor on the lower part is the wheel odometry.

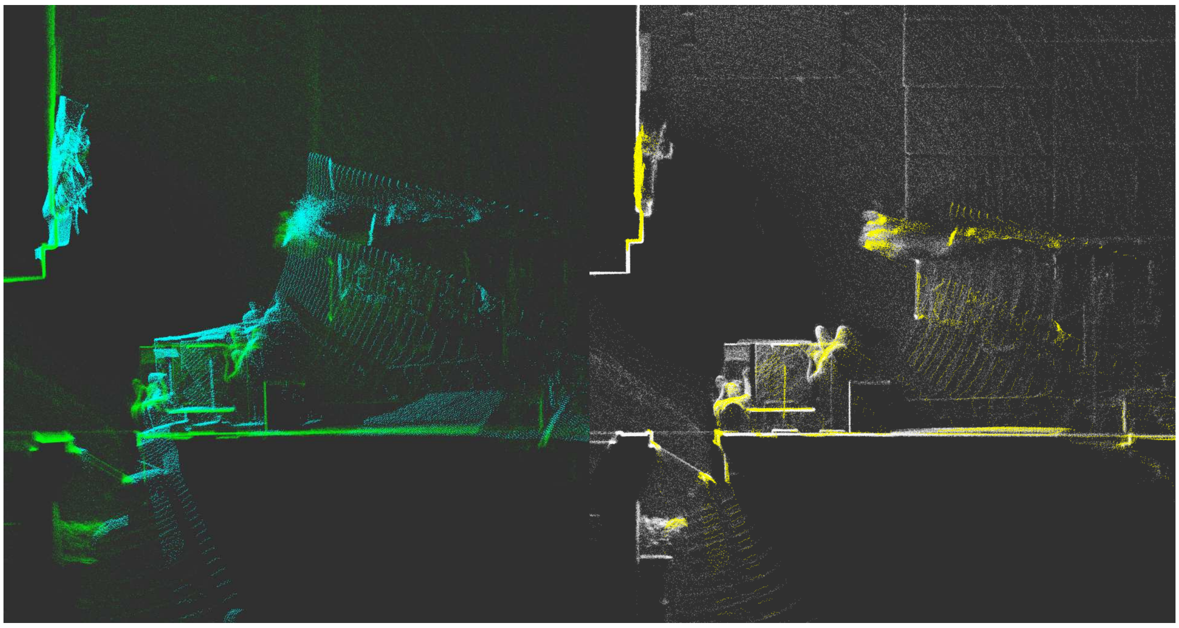


Figure 5: Two top views, before and after LIDAR extrinsic self-calibration, showing subsampled point clouds of the almost-vertical front 2d-Sick (cyan and yellow) and 3D-Velodyne (green and white) taken over about 10s of a slow left turning forward movement in a room. A comparison shows the effect of the current state of our self-calibration implementation: in the left image the two clouds are less crisp and badly coaligned, after being transformed to the image coordinate frame based only on odometry and IMU measurements and manual extrinsic calibration of the LIDARs; the two clouds in the right image, after batch optimization including LIDAR-LIDAR data associations, are crisper and much better coaligned.

4 Conclusion

Our work on the EUROPA2 self-calibration is far ahead the original plan while the development of an offline system is behind partially because we focused too much on the fundamental problems of online self-calibration and partially due to the lack of hardware specification. As the offline calibration has much lower research risk we are confident that there will be no delays for the final deliverable.

References

[1] Hatem Said Alismail and Brett Browning. Automatic calibration of spinning actuated lidar internal parameters. *Journal of Field Robotics*, 2014.

[2] Jean-Yves Bouguet. Camera calibration toolbox. `http://www.vision.caltech.edu/bouguetj/calib_doc/`.

[3] P. Furgale, T.D. Barfoot, and G. Sibley. Continuous-time batch estimation using temporal basis functions. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 2088–2095, May 2012.

[4] Paul Furgale, Joern Rehder, and Roland Siegwart. Unified temporal and spatial calibration for multi-sensor systems. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1280–1286, Tokyo, Japan, 3–7 November 2013.

[5] Patrick Mihelich James Bowman. Ros camera calibration. `http://wiki.ros.org/camera_calibration`.

[6] Jonathan Kelly and Gaurav S Sukhatme. Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration. *The International Journal of Robotics Research*, 30(1):56–79, 2011.

[7] Myoung-Jun Kim, Myung-Soo Kim, and Sung Yong Shin. A general construction scheme for unit quaternion curves with simple high order derivatives. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '95, pages 369–376, New York, NY, USA, 1995. ACM.

[8] Stefan Leutenegger, Margarita Chli, and Roland Yves Siegwart. Brisk: Binary robust invariant scalable keypoints. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2548–2555. IEEE, 2011.

[9] M. Li and A. I. Mourikis. Online temporal calibration for camera-imu systems: Theory and algorithms. *International Journal of Robotics Research*, 33(7):947–964, June 2014.

[10] Steven Lovegrove, Alonso Patron-Perez, and Gabe Sibley. Spline fusion: A continuous-time representation for visual-inertial fusion with application to rolling shutter cameras. In *Proceedings of the British Machine Vision Conference*. BMVA Press, 2013.

[11] Jerome Maye, Paul Furgale, and Roland Siegwart. Self-supervised calibration for robotic systems. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 473–480, Gold Coast, Australia, 23–26 June 2013.

[12] C McManus, P T Furgale, B E Stenning, and T D Barfoot. Lighting-invariant visual teach and repeat using appearance-based lidar. *Journal of Field Robotics*, 30(2):254–287, 2013.

[13] C. Mei and Patrick Rives. Single view point omnidirectional camera calibration from planar grids. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 3945–3950, April 2007.

[14] Gaurav Pandey, James R. McBride, Silvio Savarese, and Ryan M. Eustice. Automatic extrinsic calibration of vision and lidar by maximizing mutual information. *Journal of Field Robotics, Special Issue on Calibration for Field Robotics*, 2014. In Press.

[15] Joern Rehder, Paul Beardsley, Roland Siegwart, and Paul Furgale. Spatio-temporal laser to visual/inertial calibration with applications to hand-held, large scale scanning. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 459–465, Chicago, IL, USA, 14–18 September 2014.

[16] Robert Zlot and Michael Bosse. Efficient large-scale three-dimensional mobile mapping for underground mines. *Journal of Field Robotics*, 31(5):758–779, 2014.

5 Attached Publications

The remainder of this documents contains the following EUROPA2 publications:

- Continuous-Time Estimation of attitude using B-splines on Lie groups. Hannes Sommer, James Richard Forbes, Roland Siegwart, and Paul Furgale. About to be submitted to: Journal of Guidance, Control, and Dynamics.
- Online Self-Calibration for Robotic Systems. Jerome Maye, Paul Furgale, and Roland Siegwart. Submitted to: International Journal of Robotics Research.

Continuous-Time Estimation of attitude using B-splines on Lie groups

Hannes Sommer[1], Roland Siegwart[2], and Paul Furgale[3]
Swiss Federal Institute of Technology Zurich (ETH), Zurich, 8092, Switzerland

James Richard Forbes[4]
University of Michigan, Ann Arbor, Michigan 48109-2140

Filtering algorithms are the workhorse of spacecraft attitude estimation but recent research has shown that the use of batch estimation techniques can result in higher accuracy per unit of computational cost. This paper presents an approach for singularity-free batch estimation of attitude in continuous time using B-Spline curves on SO(3). We extend existing theory of unit-length quaternion B-splines to any Lie group and arbitrary B-spline order. For unit-length quaternion splines, we derive the equations for angular velocity, and angular acceleration. We then show how to use these curves in for batch attitude estimation using Gauss-Newton, including efficient curve initialization, a parameter update step that preserves the unit-length constraint within an unconstrained optimization framework, and the derivation of Jacobians of the state and its derivatives with respect to the update parameters. We then evaluate our algorithm on two problems: spacecraft attitude estimation using a thee-axis magnetometer, a sun sensor, and (i) a three-axis gyroscope, and (ii) a continuous-time vehicle dynamics model based on Euler's equation We compare our algorithm against a standard extended Kalman filter (EKF) and a recently-published batch attitude estimation algorithm. The results show that B-splines have equal or superior performance over all test cases and provide two key tuning parameters—the number of knots and the spline order—that an engineer can use to trade off accuracy and computational efficiency when choosing a spline representation for a given estimation problem.

[1] PhD Candidate, Autonomous Systems Lab (ASL), LEE Building, Leonhardstrasse 21; hannes.sommer@mavt.ethz.ch
[2] Professor, Autonomous Systems Lab (ASL), LEE Building, Leonhardstrasse 21; rsiegwart@ethz.ch
[3] Senior Researcher, Autonomous Systems Lab (ASL), LEE Building, Leonhardstrasse 21; paul.furgale@mavt.ethz.ch
[4] Assistant Professor, Department of Aerospace Engineering, FXB Building, 1320 Beal Street. Member AIAA; forbesrj@umich.edu

I. Introduction

Recent work on *online* state estimation in robotics has moved away from filtering algorithms toward batch processing based on Gauss-Newton. Strasdat et al. [60] showed that, using modern sparse matrix methods, batch processing results in higher accuracy per unit of computational work. This move toward batch processing has resulted in a number of new approaches to state estimation such as efficient and accurate iterative fixed-lag smoothing [41], and incremental batch estimation [34]. The benefits of treating data in batch were clearly shown by Vandersteen et al. [63] who presented a moving-horizon estimator for spacecraft attitude estimation based on an efficient sparse matrix solution to the Gauss-Newton problem. While this method shows excellent results when compared to filtering algorithms, it requires an estimate of the attitude at each measurement time, limiting it to low-rate sensors.

The requirement to estimate the state at each measurement time is fundamentally tied to the discrete-time approximation made by the vast majority of estimators. However, Furgale et al. [24] proposed an alternate approach, leaving the batch estimation problem in continuous time and approximating the state as a weighted sum of a finite number of temporal basis functions (such as a B-spline curve). A primary benefit to this approach is that it decouples the number of parameters in the estimation problem from the number of measurements as the vehicle state may be queried at any time.

In discrete time, the question of how to parameterize a three-degree-of-freedom orientation—a member of the noncommutative group SO(3)—has been subject of decades of research and debate. Algorithms such as the q-Method [65, pp. 426–428], the QUEST algorithm [59], and the more recent ESOQ and ESOQ2 [51, 52], which are batch methods that are computationally light and suitable for online and real time use, use unit-length quaternions. Although one of the first applications of the Kalman filter employed Euler angles [19], the vast majority of Kalman-type filters (i.e., the extended Kalman filter (EKF) or unscented Kalman filter (UKF)) use unit-length quaternions or a combination of unit-length quaternions and Gibbs parameters [3, 13, 40, 58]. The question is no less important in continuous time, where we must decide how to parameterize a continuously time-varying orientation. Furgale et al. [24] use the simple approach of defining a 3×1 B-spline of Cayley-Gibbs-Rodrigues parameters [7]. The B-spline provides analytical formulas for parameter rates, allowing the computation of angular velocity at any point [30].

However, analogously to the discrete-time case, any curve through the parameter space of a minimal attitude parameterization will suffer from a number of problems. In the absence of other information, a batch estimator will produce an answer that takes the shortest distance *in parameter space*, which is not necessarily the same as the shortest distance in the space of rotations. Consequently, the estimate produced may be dependent on the coordinate frame in which we choose to express the problem, as this coordinate frame decides what part of parameter space the answer lives in. Furthermore, every minimal parameterization of rotation has a singularity and so, when using this approach, there may be a danger of approaching this singularity during the estimation process.

Kim et al. [37] propose a method of generating B-splines on SO(3) using unit-length quaternions and their associated exponential map. The curves generated by this approach are valid unit-length quaternions at every point, and time derivatives of the curve are found by a straightforward use of the properties of the exponential map. Consequently, we would like to evaluate the approach proposed in Furgale et al. [24] with the one proposed in Kim et al. [37]. To do so, we must further develop the theory presented in Kim et al. [37] to be suitable for estimation. Specifically, we offer the following contributions:

1. we extend the unit-length quaternion B-spline approach of Kim et al. [37] to apply to *any* Lie group and present all derivations needed to build an estimator for a broad subclass of Lie groups;
2. we prove that this construction has the desirable property of bi-equivariance—applying a group operation from the left and right to each control vertex applies this same operation to every part of the curve—which ensures that the result of the estimation is independent of the chosen coordinate system;
3. we present a method of including arbitrary nonlinear continuous-time motion models in the estimator;
4. we develop a method of initializing the spline control vertices to act as an initial guess for batch, nonlinear minimization;
5. we derive the specific equations for unit-length quaternion curves including:
   (a) the equation for angular velocity of a body,
   (b) the equation for angular acceleration of a body, and
   (c) analytical Jacobians that relate small changes in the (unit-length quaternion) spline control vertices to small changes in orientation, angular velocity, and angular acceleration;

6. finally, we compare a family of spline-based estimators to standard approaches on two simulated spacecraft-attitude-estimation problems—one using measured dynamics from a gyroscope, and the other using dynamics based on Euler's equation. In both cases, we compare against a standard Multiplicative Extended Kalman Filter (MEKF) and the batch discrete-time estimator recently proposed by Vandersteen et al. [63].

II. Related Work

An increasing body of literature in robotics shows that state estimation based on batch optimization has a higher accuracy than filtering approaches per unit of computational work. A primary study in this is provided by Strasdat et al. [60] who show that, for the particular problem of camera-based simultaneous localization and mapping, keyframe-based optimization that chooses a subset of informative measurements outperforms filtering. Similar results have been shown for other problems, especially the fusion of visual and inertial measurements. Leutenegger et al. [41] directly use nonlinear optimization for online visual inertial sensor fusion showing higher accuracy than an EKF approach while still running at sensor rate. Mourikis et al. [53] take a different approach, proposing a filtering framework that keeps multiple clones of the state and uses batch optimization of landmark locations as an efficient update step for spacecraft entry, descent, and landing. Their follow on work shows the clear benefit of optimizing over temporally consecutive state clones when compared to pure filtering [42].

These "online batch" algorithms follow the established formula for continuous-time filtering with discrete measurements first proposed by Moore and Tam [50]. For the continuous system dynamics, they use measured dynamics (from the inertial measurement unit) in the place of the filter's control input and implement an integration scheme between exteroceptive observations. This inherently links the size of the state vector to the number of observations. For some problems, there is no drawback to this coupling. For others, such as the alignment of pushbroom imagery from satellites (see Poli and Toutin [56] and references therein), the processing of data from rolling shutter cameras [28, 54, 57], or continuously-moving sweeping laser scanners [1, 8, 17, 61]. To maintain a tractable estimation problem, each of these papers uses a continuous-time state representation to decouple the size of the state vector from the measurement times, either parametric curves [1, 8, 17, 54, 56, 57] or Gaussian process (GP) regression [61]. These two competing approaches both consider all measurements at once and they both use Gauss-Newton to derive a maximum likelihood estimate for the state variables. The key difference is on the state representation.

Tong et al. [62] describe the Gaussian Process Gauss Newton (GPGN) algorithm, which uses a finite set of samples of the state at different times as the set of parameters to estimate. The GP prediction equation is used to look up the state in between the sample times. The GP covariance function acts as a regularizer to keep the state estimate smooth. This approach has three shortcomings. First, the GP prediction equations require one to build and invert a dense matrix the size of the state vector, limiting the applicability to large problems. Second, the covariance function of the GP which defines the smoothness of the solutions must be chosen ahead of time, and doesn't represent a physical process. Hence it is currently unknown how to incorporate nonlinear continuous-time systems models into GPGN. Finally, it is not yet clear how to define a GP and a covariance function on a Lie group such as SO(3). For these reasons, we pursue a parametric approach, which allows us to naturally handle each of these issues.

The parametric curve approach uses a weighted sum of known temporal basis functions to represent the state. Although there have been many isolated publications using parametric curves over the years, Furgale et al. [24] showed how it relates back to continuous-time maximum likelihood estimation [32], presented the mathematics in their most general basis-function form, and expanded the derivation to include an arbitrary nonlinear continuous-time system model. However, in this and follow-on work they limited their applications to problems where it was possible to use curves in attitude parameter space and simple, white-noise motion system models (c.f. Oth et al. [54] and Furgale et al. [23]).

Continuous-time representation of attitude has mostly used either curves through parameter space [23, 24, 54, 56] or spherical linear interpolation on SO(3) [17, 28, 57]. Linear interpolation is singularity free but curves are only C^0 continuous, which is not a good fit for smooth vehicle motion. Anderson and Barfoot [1] model the trajectory as a curve through velocity space, which is able to represent attitude change with no singularities, but requires numerical integration of the curve if the pose is required. Kim et al. [37] present a general approach for cubic B-spline unit-quaternion curves but their target was computer graphics. This approach was adapted for state estimation by Lovegrove et al. [43], who use the same curve construction technique to build curves on SE(3). In contrast, we extend this curve construction scheme to arbitrary B-spline order (second order corresponds to spherical linear interpolation) and provide general formulas for the time derivatives and Jacobians needed for optimization.

To the best of our knowledge, this is the first work to include an arbitrary nonlinear system model into basis-function formulated state estimation. A similar optimization process was previously proposed by [29] for trajectory generation for mobile robot planning on rough terrain. They estimated a polynomial curve through the control inputs of a wheeled mobile robot that would take the robot from the current pose to the goal pose. They forward-simulated the vehicle dynamics using numerical integration, and used the error between the final pose and the goal pose to iteratively update the polynomial coefficients until convergence. In contrast, we assume the control inputs are known and use a numerical integration scheme to apply the system model to the parameterized state.

Spacecraft attitude estimation is an ideal problem to test advances in the mathematical representation of orientation and, for this reason, it has been at the center of attitude estimation research for decades. Early spacecraft-centric batch attitude determination methods, such as the q-Method [65, pp. 426–428] and QUEST [59], are solutions to Wahba's problem [64]. The q-Method and QUEST, as well as ESOQ and ESOQ2 [51, 52], employ unit-length quaternions when estimating attitude. Other solutions to Wahba's problem, such as Farrel et al. [18], Forbes and de Ruiter [21], Markley [46], estimate the attitude rotation matrix rather than an attitude parameterization. Wahba's problem in SO(n) is considered in de Ruiter and Forbes [16]. There is a vast literature devoted to spacecraft attitude estimation using Kalman-type filters such as the EKF and the UKF [14]. Unit-length quaternions are the preferred attitude description when Kalman filtering due to the absence of singularities; for example, see Bar-Itzhack and Oshman [3], Bar-Itzhack and Reiner [4], Choukroun et al. [10], Crassidis and Markley [13], Lefferts et al. [40], Shuster [58]. Often a three-parameter attitude representation, such as Gibbs parameters, are used together with a quaternion in a multiplicative framework [13, 47, 48]. Rather than trying to circumvent various issues associated the unit-length quaternion constraint within multiplicative filtering structure, Chee and Forbes [9], Zanetti et al. [66] revisit the derivation of the discrete-time Kalamn filter and directly incorporate the unit-length quaternion constraint (which is in effect a norm constraint) into the derivation. Normalization of an unconstrained estimate is shown to be optimal. Forbes et al. [22] presents a continuous-time generalization of Zanetti et al. [66]. Various authors have also considered estimating the rotation matrix directly in both stochastic and deterministic settings. Bar-Itzhack and Reiner [4], Choukroun et al. [11, 12] consider rotation matrix estimation within a Kalman filtering framework. Firoozi and Namvar [20], Khosravian and Namvar [36] consider rotation matrix estimation of a spacecraft endowed with a rate gyro and one vector measurement, generalizing the SO(3) estimator structure presented in [27, 33, 44, 45]. Similar rotation matrix estimators can be found in Grip et al. [25], Kinsey and Whitcomb [38].

The closest work to our is that of Vandersteen et al. [63] who propose a batch discrete time estimation framework for attitude estimation and spacecraft calibration. They show that batch or moving horizon estimates outperform filters in terms of accuracy and convergence rate. However, in this formulation, the number of states is tied to the number of measurement times. This limits the approach to low-rate sensors.

III. Theory

This section will present theoretical background and contributions. We will assume that the reader is familiar with the basics of B-splines and the very basics of Lie groups. For an thorough introduction to B-splines, see Bartels et al. [6]. For a brief overview with a focus on estimation, see Furgale et al. [24]. For an thorough introduction to Lie groups, see Kirillov et al. [39] and for a more applied approach, see Hall [26].

A. Construction of a Lie group valued B-spline

In this section, we derive a method to construct a B-spline curve on any arbitrary finite-dimensional Lie group. Our derivation is based on the unit-length quaternion B-spline curve from Kim et al. [37]. They define a curve based on *cumulative* B-spline basis functions. Cumulative basis functions represent an alternative but equivalent method of constructing a B-spline function on $\mathbb{R}$-vector spaces.

We will quickly motivate that alternative form. Let **f** be a B-spline function of order O defined on a increasing sequence of knots, $(t_i)_{i=1}^{I}$ with $I > O$. At time $t \in [t_i, t_{i+1})$, **f**, when $i \geq O$ and $i < I - O$, may be written as

$$\mathbf{f}(t) = \sum_{j=s(i)}^{i} b_j(t)\mathbf{c}_j, \tag{1}$$

where $s(i) := i - (O-1)$, $\mathbf{c}_j \in \mathcal{V}$ denotes the j-th control vertex in an $\mathbb{R}$-vector space $\mathcal{V}$, and b_j denotes the j-th B-spline basis function. The B-spline order, O, is precisely the spline's polynomial degree plus one. For $t < t_O$ and $t \geq t_{I-O+1}$, there are fewer than O basis functions nonzero and we treat the B-spline as undefined. This expression may be rearranged into the cumulative form as follows

$$\mathbf{f}(t) = \sum_{j=s(i)}^{i} b_j(t)\mathbf{c}_j = \mathbf{c}_{s(i)} + \sum_{j=s(i)+1}^{i} \left(\sum_{k=j}^{i} b_k(t) \right)(\mathbf{c}_j - \mathbf{c}_{j-1}) = \mathbf{c}_{s(i)} + \sum_{j=1}^{O-1} \beta_{i,j}(t)(\mathbf{c}_{s(i)+j} - \mathbf{c}_{s(i)+j-1}), \tag{2}$$

defining the cumulative basis functions $\beta_{i,j}(t) := \sum_{k=s(i)+j}^{i} b_k(t)$ for $1 \leq j \leq O-1$.

We will now generalize this form (2) to the Lie group, $\mathcal{G}$, just as Kim et al. [37] do it for unit-length quaternions. Let $(\mathcal{G}, \cdot)$ be a connected finite dimensional Lie group, with Lie algebra $\mathfrak{g}$. Let $\boldsymbol{\iota} \in \mathcal{G}$ denote its identity element. Instead of vectors, $\mathbf{c}_i$, a Lie-group-valued curve is defined by a tuple of Lie group elements as control vertices, $\mathbf{q}_i \in \mathcal{G}$. The sum of vectors is naturally identified with the corresponding sequence of Lie group operations (written here as multiplication) in the order given by the indices. The vector difference $(\mathbf{c}_k - \mathbf{c}_{k-1})$, where $k = s(i)+j$, is generalized to the left fraction $(\mathbf{q}_{k-1}^{-1}\mathbf{q}_k)$. The scaling of these "differences" (by the $\beta_{i,j}(t)$) is done "in the Lie algebra" by exploiting the exponential and logarithm maps of $\mathcal{G}$. This way one gets the generalized form of (2) as

$$\mathbf{q}(t) := \mathbf{q}_{s(i)} \prod_{j=1}^{O-1} \mathbf{r}_{i,j}(t), \tag{3}$$

where

$$\mathbf{r}_{i,j}(t) := \exp\left(\beta_{i,j}(t) \boldsymbol{\varphi}_{s(i)+j} \right), \tag{4}$$

and

$$\boldsymbol{\varphi}_k := \log\left(\mathbf{q}_{k-1}^{-1}\mathbf{q}_k \right). \tag{5}$$

This construction requires that the $\log$ function is defined on $\mathbf{q}_{k-1}^{-1}\mathbf{q}_k$. To guarantee this, one has to make sure that there are enough intermediate points to have all $\mathbf{q}_{k-1}^{-1}\mathbf{q}_k$ close enough to identity. In a connected Lie group, this should always be possible with finite many $\mathbf{q}_i$. Note that the vector space construction, (2), is equivalent to the Lie group approach, (3), when $(\mathcal{G}, \cdot)$ is chosen as the additive group $(\mathcal{V}, +)$ of the vector space $\mathcal{V}$, which is always a commutative Lie group. In that sense, these Lie group valued B-splines are a true generalization of the traditional vector-space-valued B-splines.

B. Theoretical preparations

To facilitate the analysis of Lie group B-splines we will start with some theoretical preparations.

1. The ambient algebra assumption

First we assume the Lie group to be embedded in an ambient associative algebra.

Assumption 1. *The Lie group $(\mathcal{G}, \cdot)$ is assumed to be a multiplicative sub monoid and embedded differential manifold of a unital associative $\mathbb{R}$-algebra $(\mathcal{A}, +, \cdot)$.*

Informally, $\mathcal{A}$ is an $\mathbb{R}$-vector space equipped with an $\mathbb{R}$-bi-linear associative vector multiplication and a multiplicative identity element. The Lie group, $\mathcal{G}$, is a subset that inherits its (semi) group and differential structure from $\mathcal{A}$. All real matrix Lie groups, the unit-length quaternion Lie group, the unit-length dual quaternions Lie group, and the unit complex number Lie group all naturally fulfill this assumption. Those who feel uncomfortable with the abstract concept of an $\mathbb{R}$-algebra are encouraged to think of the unit-length quaternions as $\mathcal{G}$ and the quaternions as $\mathcal{A}$, or the group SO(3) as $\mathcal{G}$ and $\mathbb{R}^{3\times 3}$ as $\mathcal{A}$.

The Assumption 1 allows us to take derivatives of curves through $\mathcal{G}$ in the ambient algebra $\mathcal{A}$, where we can easily use the product rule. The resulting formulas will be expressions in this algebra (with a global tangent space) and thus simpler to calculate than doing the same intrinsically with respect to a basis for each point.

In this setting, it is useful to have special operators to retrieve the matrix the left and right multiplication by a given vector $\mathbf{a} \in \mathcal{A}$ representing (with respect to a fixed default basis for $\mathcal{A}$). This is always uniquely possible because both multiplications are linear maps. We will denote them with $\mathbf{a}^L$ and $\mathbf{a}^R$ respectively. This implies

$$\Psi(\mathbf{ab}) = \mathbf{a}^L \Psi(\mathbf{b}) = \mathbf{b}^R \Psi(\mathbf{a}), \tag{6}$$

where $\mathbf{a}, \mathbf{b} \in \mathcal{A}$ and $\Psi : \mathcal{A} \to \mathbb{R}^{\dim \mathcal{A}}$ maps the vectors to their coordinate tuples (with respect to the default basis). In the following we will implicitly identify vectors with their coordinates. These operators, restricted to $\mathcal{G}$, are Lie group-monomorphism, $(\cdot)^L$, and antimonomorphism, $(\cdot)^R$, from $\mathcal{G}$ into GL$(\mathcal{A})$. This means they are (anti)isomorphisms to their images. And the anti prefix refers to the antihomomorphy identity, which one gets from the homomorphy identity by swapping the arguments of one of the binary operations. These (anti)monomorphisms give us the opportunity to reduce computational complexity of Jacobian matrix evaluation by choosing to apply the multiplication to elements of $\mathcal{G}$ rather than $\mathcal{G}^L$ or $\mathcal{G}^R$. The mapping $(\cdot)^L$ also leads to the following theorem.

Theorem 1. *$\mathcal{A}^L$ is a sub $\mathbb{R}$-algebra of $\mathbb{R}^{\dim \mathcal{A} \times \dim \mathcal{A}}$, $\mathcal{G}^L$ is a matrix Lie group embedded in $\mathcal{A}^L$, and this pair is fully isomorphic to the pair of $\mathcal{G}$ and $\mathcal{A}$.*

We will use this theorem to import knowledge about matrix Lie groups to our general setting. It also tells us that—up to isomorphism—Assumption 1 restricts us exactly to real matrix Lie groups because, for any matrix Lie group, $\mathcal{G}$, there is an $\mathcal{A}$ such that the pair, $(\mathcal{A}, \mathcal{G})$, fulfills this assumption. This implies that we could have restricted these derivations to the $\mathbb{R}$-matrix Lie groups in the first place. However, it is still of practical use to know that all derivations are correct in the nonmatrix examples, such as the unit-length quaternions. A proof is provided in the appendix, proof of Theorem 1.

2. *The exponential maps*

In both $\mathcal{A}$ and $\mathcal{G}$ there is a canonical definition of an exponential map. For the algebra, $\mathcal{A}$, the exponential map, $\exp_{\mathcal{A}}$, is defined by the usual formal power series. Applied to matrices, this yields the matrix exponential map. For the Lie group, $\mathcal{G}$, the exponential map, $\exp_{\mathcal{G}}$, is required to be smooth and, for all $\mathbf{v} \in \mathfrak{g}$, the directional exponential map $\exp_{\mathcal{G}}^{\mathbf{v}} : \mathbb{R} \to \mathcal{G}, t \mapsto \exp_{\mathcal{G}}(t\mathbf{v})$ must fulfill two identities: the classical exponentiation identity,

$$\exp_{\mathcal{G}}^{\mathbf{v}}(\alpha + \beta) = \exp_{\mathcal{G}}^{\mathbf{v}}(\alpha) \exp_{\mathcal{G}}^{\mathbf{v}}(\beta), \tag{7}$$

for all $\alpha, \beta \in \mathbb{R}$, and the differential identity,

$$\frac{\mathrm{d}}{\mathrm{d}t} \exp_{\mathcal{G}}^{\mathbf{v}}(t)|_{t=0} = \mathbf{v}. \tag{8}$$

Theorem 2. *In our setting $\exp_{\mathcal{G}}$ is a restriction of $\exp_{\mathcal{A}}$ to $\mathfrak{g} \subset \mathcal{A}$, employing the natural identification of $\mathcal{G}$'s tangent spaces with sub vector spaces of $\mathcal{A}$.*

We exploit this property to simplify Jacobian calculation by interpreting our B-spline construction (3) as construction in $\mathcal{A}$. A proof is provided in the appendix, proof of Theorem 2. From now on we won't distinguish anymore formally between both exponential maps and write just $\exp$.

C. Bi-equivariance of this B-spline construction

In this section we will prove that this B-spline construction has an attractive property for a Lie group valued parameterized curve, called “bi-invariance” by [55] for the special case of SO(3). From our point of view, the term bi-equivariance is a better fit because it requires the curve’s value to transform the same way as its control vertices when transformed by left or right multiplication with a group element. The term “invariant” would indicate no change in the curve’s value for the same operations, but that is neither the case nor intended for our splines or for the curves presented in [55]. In contrast equivariance of a map with respect to a group action is exactly that: the function’s value changes according to the same group action as applied on its argument. We will denote the set of all multiplicative invertible elements of $\mathcal{A}$ (usually called units) with $\mathcal{A}_{\text{inv}}$.

First we consider left-equivariance: equivariance with respect to left multiplication by any $\mathbf{u} \in \mathcal{A}_{\text{inv}}$, $L_{\mathbf{u}} : \mathcal{A} \to \mathcal{A}, \mathbf{a} \mapsto \mathbf{ua}$. The expression for $\mathbf{r}_{i,j}$ in (4) only depends on the control vertices through expressions like $\mathbf{q}_{k-1}^{-1}\mathbf{q}_k$. Such expressions are invariant with respect to $L_{\mathbf{u}}$, rendering all the $\mathbf{r}$ invariant, too. It follows that

$$\mathbf{q}((L_{\mathbf{u}}(\mathbf{q}_k))_1^K, t) = L_{\mathbf{u}}(\mathbf{q}_{s(i)}) \prod_{j=1}^{O-1} \mathbf{r}_{i,j}(t) = L_{\mathbf{u}}(\mathbf{q}((L_{\mathbf{u}}(\mathbf{q}_k))_1^K, t)) \tag{9}$$

Next we consider the right equivariance proof. For $\mathbf{u} \in \mathcal{A}_{\text{inv}}$, the *conjugation* by $\mathbf{u}$, $\pi_{\mathbf{u}} : \mathcal{A} \to \mathcal{A}, \mathbf{a} \mapsto \mathbf{uau}^{-1}$ is an $\mathbb{R}$-algebra automorphism. Therefore any (partial) function of type $\mathcal{A}^k \to \mathcal{A}$ for some $k \in \mathbb{N}$ is automatically equivariant with respect to conjugation by $\mathbf{u}$ if the function is *$\mathcal{A}$-intrinsically defined*. A function from $\mathcal{A}^k \to \mathcal{A}$ is called $\mathcal{A}$-intrinsically defined if and only if its definition is purely based on the structure of $\mathcal{A}$; informally this means that its value and its domain can be defined by only referring to its arguments, real numbers, $\mathbf{0}, \mathbf{1} \in \mathcal{A}$, the operations $+$, $*$, the limit in $\mathcal{A}$, the identity relation, set or logical operators, or other $\mathcal{A}$-intrinsically defined functions. This implies that the $\exp$ and $\log$ mappings on $\mathcal{A}$, and ultimately our whole B-spline construction (3), are equivariant with respect to conjugation by $\mathbf{u}$ when the time parameter is considered a constant real number. The B-spline’s right-equivariance immediately follows from the simple fact that the right multiplication, $R_{\mathbf{u}} : \mathcal{A} \to \mathcal{A}, \mathbf{a} \mapsto \mathbf{au}$, is equal to a left multiplication after a conjugation:

$$R_{\mathbf{u}} = L_{\mathbf{u}} \circ \pi_{\mathbf{u}^{-1}}$$

These equivariances hold for any $\mathbf{u} \in \mathcal{A}_{\text{inv}}$, but the transformed spline will remain $\mathcal{G}$-valued if and only if $\mathbf{u} \in \mathcal{G}$. Therefore, in practice, only the equivariances with respect to left or right multiplication by $\mathbf{u} \in \mathcal{G}$ will be of interest.

D. Derivatives

1. Time derivatives

We can calculate the B-splines derivatives with respect to t using the usual product rule supported by Assumption 1, which allows us to interpret the whole B-spline construction (3) as defined more generally in $\mathcal{A}$.

$$\frac{\mathrm{d}^k}{\mathrm{d}t^k}\mathbf{q}(t) = \frac{\mathrm{d}^k}{\mathrm{d}t^k}\mathbf{q}_s \prod_{j=1}^{O-1} \mathbf{r}_{i,j}(t) = \mathbf{q}_{s(i)} \sum_{\substack{\boldsymbol{\alpha}\in\mathbb{N}^{O-1},\\ \sum\boldsymbol{\alpha}=k}} \prod_{j=1}^{O-1} \frac{k!}{\boldsymbol{\alpha}_j!}\frac{\mathrm{d}^{\boldsymbol{\alpha}_j}}{\mathrm{d}t^{\boldsymbol{\alpha}_j}}\mathbf{r}_{i,j}(t) \tag{10}$$

This can not be further simplified by summing only over ordered $\boldsymbol{\alpha}$ because the multiplication in $\mathcal{A}$ may not be commutative. To go further requires the time derivatives of $\mathbf{r}_{i,j}$ up to order k. For that step we use the directional exponential,

$$\exp^{\mathbf{a}}(t) := \exp(t\mathbf{a}), \tag{11}$$

for any $\mathbf{a} \in \mathcal{A}$ as function of $t \in \mathbb{R}$. For its derivative we can benefit from the well known property of the matrix exponential map on $\mathbb{R}^{\dim\mathcal{A}\times\dim\mathcal{A}}$, which also holds in $\mathcal{A}$ guaranteed by Theorem 1,

$$\frac{\mathrm{d}}{\mathrm{d}t}\exp^{\mathbf{a}}(t) = \mathbf{a}\exp^{\mathbf{a}}(t). \tag{12}$$

Using the directional exponential map we can easily derive expressions for the derivatives of the $\mathbf{r}_{i,j}(t)$ expressions making use of Theorem 2 to identify the exp and log in (4) and (5) with their extensions to $\mathcal{A}$, with $k := s(i) + j$:

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{r}_{i,j}(t) = \frac{\mathrm{d}}{\mathrm{d}t}\exp^{\boldsymbol{\varphi}_k}(\beta_{i,j}(t)) = \beta'_{i,j}(t)\boldsymbol{\varphi}_k\exp^{\boldsymbol{\varphi}_k}(\beta_{i,j}(t)) \tag{13}$$

$$= \beta'_{i,j}(t)\boldsymbol{\varphi}_k\mathbf{r}_{i,j}(t) \tag{14}$$

$$\frac{\mathrm{d}^2}{\mathrm{d}t^2}\mathbf{r}_{i,j}(t) = (\beta'_{i,j}(t)^2\boldsymbol{\varphi}_k + \beta''_{i,j}(t)\boldsymbol{\iota})\boldsymbol{\varphi}_k\mathbf{r}_{i,j}(t) \tag{15}$$

$$\frac{\mathrm{d}^3}{\mathrm{d}t^3}\mathbf{r}_{i,j}(t) = \left(\beta'_{i,j}(t)\boldsymbol{\varphi}_k(\beta'_{i,j}(t)^2\boldsymbol{\varphi}_k + \beta''_{i,j}(t)) + 2\beta'_{i,j}(t)\beta''_{i,j}(t)\boldsymbol{\varphi}_k + \beta'''_{i,j}(t)\boldsymbol{\iota}\right)\boldsymbol{\varphi}_k\mathbf{r}_{i,j}(t) \tag{16}$$

We will not need higher order derivatives for our experiments.

2. *Jacobians with respect to changes in the control vertices*

To implement the Gauss-Newton or Levenberg-Marquardt optimization algorithms analytically, we must have access to an expression for the Jacobian (with respect to small changes in the state variables) for any nonlinear function that appears in our cost functions. This involves the general problem of providing Jacobians with respect to to variables constrained to lie on a differentiable manifold. Here we assume that we represent the control vertices as vectors in the n-dimensional vector space, $\mathcal{A}$, constrained to lie in a m-dimensional Lie group, $\mathcal{G} \subset \mathcal{A}$. Typically, $n > m$. Rather than performing constrained optimization, it is usual to use a *minimal perturbation* in the tangent space mapped to the Lie group via its exponential map. In this paper, we choose to multiply the exponential map on the left,

$$\mathbf{q}(\delta\boldsymbol{\phi}) = \exp(\delta\boldsymbol{\phi})\overline{\mathbf{q}}, \tag{17}$$

where $\overline{\mathbf{q}}$ is our current guess, and $\delta\boldsymbol{\phi}$ is a minimal (m-dimensional), perturbation. When an iteration of Levenberg-Marquardt produces an answer for $\delta\boldsymbol{\phi}$, the update is applied as

$$\overline{\mathbf{q}} \leftarrow \exp(\delta\boldsymbol{\phi})\overline{\mathbf{q}}. \tag{18}$$

This update is *constraint sensitive* in that the updated $\overline{\mathbf{q}}$ is still in the Lie group $\mathcal{G}$. In other words, given a basis, $\mathcal{B}$, for the Lie algebra, $\mathfrak{g}$, each iteration of the optimization algorithm is executed in the exponential chart centered at the current guess and determined by the right multiplied basis, $\mathcal{B} \cdot \overline{\mathbf{q}}$ (element-wise). The concept can be transferred from Lie groups to Riemannian manifolds in a straightforward way (then using the Riemannian exponential map). Both are instances of the usual concept for differentiable manifolds, for which there is no such canonical choice for the charts. We will denote this chart map at $\overline{\mathbf{q}}$ with

$$\Phi_{\overline{\mathbf{q}}} : \mathbb{R}^m \to \mathcal{G}, \boldsymbol{\phi} \mapsto \exp(\mathcal{B}(\boldsymbol{\phi}))\overline{\mathbf{q}},$$

where $\mathcal{B}(\boldsymbol{\phi}) := \sum_{i=1}^{m} \boldsymbol{\phi}_i \mathbf{b}_i$ denotes the vector in $\mathfrak{g}$ corresponding to the coordinates, $\boldsymbol{\phi}$, with respect to $\mathcal{B}$. To use this method to perform unconstrained optimization using the Lie group-valued B-splines, we must derive the Jacobian of (3) with respect to small changes in the individual control vertices. Let us derive the partial Jacobians of the whole B-spline with respect to the small perturbations, $\boldsymbol{\phi}_l$, while considering the control vertices, $\mathbf{q}_l(\boldsymbol{\phi}_l) = \exp(\mathcal{B}\boldsymbol{\phi})\overline{\mathbf{q}}_l = \Phi_{\overline{\mathbf{q}}_l}(\boldsymbol{\phi})$, as functions of $\boldsymbol{\phi}_l$. Any partial derivative will thus be implicitly at $\mathbf{0} \in \mathbb{R}^m$ and all elements of $\mathcal{G} \subset \mathcal{A}$ implicitly identified with their coordinates with respect to $\mathcal{A}$'s default basis. Omitting the time parameter we get:

$$\frac{\partial \mathbf{q}}{\partial \boldsymbol{\phi}_l} = \frac{\partial}{\partial \boldsymbol{\phi}_l}\mathbf{q}_{s(i)}\prod_{j=1}^{O-1}\mathbf{r}_{i,j} = \left(\prod_{j=1}^{O-1}\mathbf{r}_{i,j}\right)^R \frac{\partial \mathbf{q}_{s(i)}}{\partial \boldsymbol{\phi}_l} + {\mathbf{q}_{s(i)}}^L \sum_{\hat{j}=1}^{O-1}\left(\prod_{j=1}^{\hat{j}-1}\mathbf{r}_{i,j}\right)^L \left(\prod_{j=\hat{j}+1}^{O-1}\mathbf{r}_{i,j}\right)^R \frac{\partial \mathbf{r}_{i,\hat{j}}}{\partial \boldsymbol{\phi}_l} \tag{19}$$

For the first summand we get zero for $l \neq s(i)$ and for $l = s(i)$,

$$\frac{\partial \mathbf{q}_{s(i)}}{\partial \boldsymbol{\phi}_l} = \exp(\mathcal{B}\boldsymbol{\phi}_l)\overline{\mathbf{q}}_{s(i)} = {\overline{\mathbf{q}}_{s(i)}}^R\mathbf{V}, \tag{20}$$

where $\mathbf{V}$ denotes the Jacobian matrix of $\mathcal{G}$'s exponential map in coordinates, $\Psi \circ \exp \circ \mathcal{B}$, at $\mathbf{0}$. Note that, because the derivative of the $\exp$ map in $\mathcal{A}$ at zero is the identity map, this $\mathbf{V}$-matrix is also the Jacobian of the implicit identification $\mathfrak{g}$ into $\mathcal{A}$. We can use $\mathbf{V}$ to convert coordinates with respect to $\mathcal{B}$ to coordinates in $\mathcal{A}$ with respect to the default basis.

We will derive the Jacobian of $\mathbf{r}_{i,j} = \frac{\partial}{\partial \boldsymbol{\phi}_l}\exp(\beta_{i,j}\boldsymbol{\varphi}_k)$, for any $1 \leq j < O$ and with $k := s(i) + j$, by preparing Jacobians for the involved functions from inner to outer.

For $\boldsymbol{\varphi}_k = \log(\mathbf{d})$, with $\mathbf{d} := \mathbf{q}_{k-1}^{-1}\mathbf{q}_k$ we denote the Jacobian of the $\log$ at $\overline{\mathbf{d}}$ with $\mathbf{L}(\overline{\mathbf{d}})$ and assume it to be given in the basis $\mathcal{B}$ for its codomain, $\mathfrak{g}$. We use the chart, $\Phi_{\overline{\mathbf{d}}}$, for its domain in the neighborhood of $\overline{\mathbf{d}}$ because it is more compact to calculate and closed form solutions are available for our target Lie groups. This matrix is precisely the first order approximation with respect to perturbations of the form in (17),

$$\log\big(\exp(\boldsymbol{\phi})\overline{\mathbf{d}}\big) \approx \log(\overline{\mathbf{d}}) + \mathbf{L}(\overline{\mathbf{d}})\boldsymbol{\phi}. \tag{21}$$

To use this matrix, we decompose $\boldsymbol{\varphi}_k = \log(\mathbf{d})$ into $(\log \circ \Phi_{\overline{\mathbf{d}}} \circ \Phi_{\overline{\mathbf{d}}}^{-1})(\mathbf{d})$ because $\mathbf{L}(\overline{\mathbf{d}})$ was defined to be a Jacobian of $\log \circ \Phi_{\overline{\mathbf{d}}}$. So we are only lacking the following Jacobian:

$$\frac{\partial}{\partial \boldsymbol{\phi}_l}\Phi_{\overline{\mathbf{d}}}^{-1}(\mathbf{d}) = \frac{\partial}{\partial \boldsymbol{\phi}_l}\log(\mathbf{d}\overline{\mathbf{d}}^{-1}) = \mathbf{W}\frac{\partial}{\partial \boldsymbol{\phi}_l}(\mathbf{d}\overline{\mathbf{d}}^{-1}) \tag{22a}$$

$$= \mathbf{W}\frac{\partial}{\partial \boldsymbol{\phi}_l}(\overline{\mathbf{q}}_{k-1}^{-1}\exp(-\phi_{k-1})\exp(\phi_k)\overline{\mathbf{q}}_k(\overline{\mathbf{q}}_{k-1}^{-1}\overline{\mathbf{q}}_k)^{-1}) \tag{22b}$$

$$= \mathbf{W}\big(\overline{\mathbf{q}}_{k-1}^{-1}\big)^L(\overline{\mathbf{q}}_{k-1})^R\frac{\partial}{\partial \boldsymbol{\phi}_l}\exp(-\phi_{k-1})\exp(\phi_k) \tag{22c}$$

$$= \underbrace{\mathbf{W}\big(\overline{\mathbf{q}}_{k-1}^{-1}\big)^L(\overline{\mathbf{q}}_{k-1})^R\mathbf{V}}_{=:\mathbf{C}(\overline{\mathbf{q}}_{k-1}^{-1})}\begin{cases} -\mathbf{1} & , l = k-1 \\ \mathbf{1} & , l = k, \\ \mathbf{0} & , \text{otherwise} \end{cases} \tag{22d}$$

We used $\mathbf{W}$ to denote the Jacobian of the $\log$ at $\boldsymbol{\iota}$ with respect to the default basis for $\mathcal{A}$ and $\mathcal{B}$ for $\mathfrak{g}$. The introduced $\mathbf{C}(\mathbf{q})$ is precisely the Jacobian of the *adjoint operation* of an element, $\mathbf{q} \in \mathcal{G}$, on $\mathfrak{g}$, which implies that $\mathbf{C}(\mathbf{q}^{-1}) = (\mathbf{C}(\mathbf{q}))^{-1}$.

The last missing piece is $\mathbf{E}(\overline{\mathbf{v}})$, the Jacobian of $\mathcal{G}$'s exponential map at $\overline{\mathbf{v}} := \beta_{i,j}\overline{\boldsymbol{\varphi}}_k$ with respect to the default basis on both sides, which fulfills the first-order approximation,

$$\exp\left(\overline{\mathbf{v}} + \delta\mathbf{v}\right) \approx \exp\left(\overline{\mathbf{v}}\right) + \mathbf{E}(\overline{\mathbf{v}})\delta\mathbf{v}. \tag{23}$$

If the exponential map's Jacobian is instead available in the exponential chart, $\Phi_{\exp(\mathbf{v})}$, here denoted with $\mathbf{S}(\overline{\mathbf{v}})$, one can use the following identity to retrieve $\mathbf{E}$,

$$\mathbf{E}(\overline{\mathbf{v}}) = \exp(\overline{\mathbf{v}})^R\mathbf{V}\mathbf{S}(\overline{\mathbf{v}}), \tag{24}$$

which holds because $\exp(\overline{\mathbf{v}})^R\mathbf{V}$ is the Jacobian of $\Phi_{\exp(\overline{\mathbf{v}})}$ at zero with respect to the default basis for $\mathcal{A}$ as codomain. This $\mathbf{S}$-matrix is well known for attitude representations as the matrix that relates parameter rates to angular velocity (c.f. [30]). Altogether we finally get:

$$\frac{\partial\boldsymbol{\varphi}_k}{\partial\boldsymbol{\phi}_k} = \mathbf{L}\left(\overline{\mathbf{q}}_{k-1}^{-1}\overline{\mathbf{q}}_k\right)\mathbf{C}(\overline{\mathbf{q}}_{k-1}^{-1}) \tag{25a}$$

$$\frac{\partial\boldsymbol{\varphi}_k}{\partial\boldsymbol{\phi}_{k-1}} = -\frac{\partial\boldsymbol{\varphi}_k}{\partial\boldsymbol{\phi}_k} \text{ and } \frac{\partial\boldsymbol{\varphi}_k}{\partial\boldsymbol{\phi}_l} = \mathbf{0} \text{ for } l \notin \{k-1, k\} \tag{25b}$$

$$\frac{\partial\mathbf{r}_{i,j}}{\partial\boldsymbol{\phi}_k} = \frac{\partial}{\partial\boldsymbol{\phi}_k}\exp\left(\beta_{i,j}\boldsymbol{\varphi}_k\right) = \mathbf{E}(\beta_{i,j}\overline{\boldsymbol{\varphi}}_k)\beta_{i,j}\frac{\partial\boldsymbol{\varphi}_k}{\partial\boldsymbol{\phi}_k} \tag{25c}$$

$$= \overline{\mathbf{r}}_{i,j}{}^R\mathbf{V}\mathbf{S}(\beta_{i,j}\overline{\boldsymbol{\varphi}}_k)\beta_{i,j}\frac{\partial\boldsymbol{\varphi}_k}{\partial\boldsymbol{\phi}_k} \tag{25d}$$

We will also need the corresponding Jacobians for the time derivatives of the B-spline. Considering (19) and (25), we have to derive the following (not expanding the derivatives of $\boldsymbol{\varphi}_k$ for the sake of readability):

$$\frac{\partial}{\partial\boldsymbol{\phi}_k}\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{r}_{i,j}(t) = \beta'_{i,j}(t)\frac{\partial}{\partial\boldsymbol{\phi}_k}(\mathbf{V}\boldsymbol{\varphi}_k)\mathbf{r}_{i,j}(t) = \beta'_{i,j}(t)\left(\mathbf{r}_{i,j}(t)^R\mathbf{V}\frac{\partial\boldsymbol{\varphi}_k}{\partial\boldsymbol{\phi}_k} + (\mathbf{V}\boldsymbol{\varphi}_k)^L\frac{\partial\mathbf{r}_{i,j}(t)}{\partial\boldsymbol{\phi}_k}\right) \tag{26}$$

$$\begin{aligned}\frac{\partial}{\partial\boldsymbol{\phi}_k}\frac{\mathrm{d}^2}{\mathrm{d}t^2}\mathbf{r}_{i,j}(t) &= \frac{\partial}{\partial\boldsymbol{\phi}_k}\left(\beta'_{i,j}(t)^2\mathbf{V}\boldsymbol{\varphi}_k + \beta''_{i,j}(t)\boldsymbol{\iota}\right)(\mathbf{V}\boldsymbol{\varphi}_k)\mathbf{r}_{i,j}(t)\\ &= \left(\beta'_{i,j}(t)^2\mathbf{V}\boldsymbol{\varphi}_k + \beta''_{i,j}(t)\boldsymbol{\iota}\right)^L\left(\mathbf{r}_{i,j}(t)^R\mathbf{V}\frac{\partial\boldsymbol{\varphi}_k}{\partial\boldsymbol{\phi}_k} + (\mathbf{V}\boldsymbol{\varphi}_k)^L\frac{\partial\mathbf{r}_{i,j}(t)}{\partial\boldsymbol{\phi}_k}\right)\\ &\quad + ((\mathbf{V}\boldsymbol{\varphi}_k)\mathbf{r}_{i,j}(t))^R(\beta'_{i,j}(t)^2\mathbf{V}\frac{\partial\boldsymbol{\varphi}_k}{\partial\boldsymbol{\phi}_k})\end{aligned} \tag{27}$$

E. Rough B-spline fitting for initialization

To initialize the nonlinear optimization of a $\mathcal{G}$-valued curve we will need a strategy to come to a suitable initial guess. Formally, this is the problem of fitting a curve defined on [0, T] to a time series in $\mathcal{G}$, $\alpha := (\tau_k, g_k)_{k=1}^K \in [0, T] \times \mathcal{G}$, while somehow penalizing high acceleration, and preventing overfitting. For our experiments we used the following strategy to get an rough fit of our B-spline by making use of Assumption 1.

As we have $\mathbf{q}(t, (\mathbf{q}_k)_{k=1}^{I}) \in \mathcal{G} \subset \mathcal{A}_{\text{inv}} \subset \mathcal{A}$ for all times $t \in [0, T]$ and control vertices $\mathbf{q}_k \in \mathcal{G}$ we can consider the whole $\mathcal{G}$-valued B-spline construct as a Lie group B-spline in the Lie group $(\mathcal{A}_{\text{inv}}, \cdot)$, i.e. based on the multiplication in $\mathcal{A}$. In $\mathcal{A}$ we also have the $\mathbb{R}$-vector space structure and we can define for each multiplicative $\mathbf{q}(t, (\mathbf{q}_k)_{k=1}^{I})$ the corresponding additive B-spline $\mathbf{q}^{\text{add}}(t, (\mathbf{q}_k)_{i=1}^{I})$ as defined in the additive Lie group, $(\mathcal{A}, +)$. These are now traditional vector-space B-splines and we know how to efficiently fit them to the time series α. This is a linear problem, even with the acceleration penalty as regularization (see the solution of Schoenberg and Reinsch as described in Chapter XIV of [15]). After the linear fit, we may needs to modify the control vertices to make them elements in $\mathcal{G}$. As long as they are not too far off this can be done uniquely and for some common examples also efficiently. For example in the case of unit-length quaternions this just means normalizing the resulting vectors in $\mathbb{R}^4$ to unit-length.

To summarize, we get our initial guess for optimization using the following steps:

1. acquire control vertices $\mathbf{q}_k \in \mathcal{A}$ by doing a least squares fit of $\mathbf{q}^{\text{add}}$ to α while keeping the integral over the splines acceleration low (by introducing an appropriate, linear cost),
2. project the $\mathbf{q}_k$s into $\mathcal{G}$, and
3. use the modified $\mathbf{q}_k$ as control vertices for a multiplicative spline in $\mathcal{G}$.

Despite being very simple, this method worked extremely well in all of our experiments.

But still the experiments can show if this strategy works for a specific example. And in fact it worked for all our experiments very well up to B-spline order 12, where the initialization started to lead into wrong local likelihood minima for knot spacing above a certain duration with our default acceleration penalty. This could be solved by rising it from a factor of $1e4$ to $1e6$. How to choose this tuningparamer automatically would need further investiagion similiar to what was done in [54] but is out scope for this paper.

Note that it is not equally good to just use a subset of the values in α as control vertices, even though this would also result in a somewhat close to curve. Basically because we would have no means to penalize high acceleration, which would allow outliers to badly affect the so fitted B-spline.

F. Integrals along curves

To deal with cost function (summands) that involve integrals along our B-splines (e.g. to impose dynamic system models) we used the following approach. Given a time interval $[0, T]$ and a cost functional J defined on continuous curves through $\mathcal{G}$, $\gamma : [0, T] \to \mathcal{G}$ such that

$$J(\gamma) = \int_0^T f(\gamma(t))^T W(t) f(\gamma(t))\,\mathrm{d}t,$$

where $l \in \mathbb{N}$, $W : [0, T] \to \mathbb{R}^{l\times l}$ and $f : \mathcal{G} \to \mathbb{R}^l$, each also continuous, we apply a nonadaptive numeric integration scheme $(w, \xi) : \{1 \dots k\} \to \mathbb{R} \times [0, T]$ (e.g. Simpson's, Trapezoidal rule, etc.) such that

$$\bigforall_{g\in\mathcal{C}([0,T],\mathbb{R})} \sum_{i=1}^{k} w_i g(\xi_i) \approx \int_0^T g(t)\,\mathrm{d}t$$

to convert J into a sequence of quadratic error terms (to interface a Gauss-Newton optimization package), this becomes

$$J(\gamma) \approx \sum_{i=1}^{k} w_i f(\gamma(\xi_i))^T W(\xi_i) f(\gamma(\xi_i)).$$

G. The unit-length quaternions Lie group

In this section we will focus on aspects special for the Lie group of unit-length quaternions when used as representative for SO(3). To be prepared for that we will shortly define some notation. First, let $\mathbb{H}$ denote the skew field and $\mathbb{R}$-algebra of quaternions and

$$S^0(\mathbb{H}) := \{\mathbf{q} \in \mathbb{H} | |\mathbf{q}| = 1\}$$

denote the Lie group of unit-length quaternions, the zero-dimensional sphere in $\mathbb{H}$. We can make use of the theory in section III B using the following identification. In this setting, $\mathbb{H}$ is the the ambient $\mathbb{R}$-algebra ($\mathcal{A}$) and $S^0(\mathbb{H})$ is the embedded Lie group ($\mathcal{G}$).

We will rely on the notation presented in Barfoot et al. [5]. Let $\boldsymbol{\iota}$ denote the *identity element* of $S^0(\mathbb{H})$, which is equal to the multiplicative identity element in $\mathbb{H}$. By using $(\mathbf{i}, \mathbf{j}, \mathbf{k}, \boldsymbol{\iota})$ as a basis for $\mathbb{H}$, we can write the coordinates of a quaternion, $\mathbf{q}$, as

$$\mathbf{q} =: \begin{bmatrix} \boldsymbol{\epsilon} \\ \eta \end{bmatrix}, \tag{28}$$

where $\boldsymbol{\epsilon}$ is 3×1, η is a scalar. We will denote the product of two quaternions $\mathbf{p}, \mathbf{q} \in \mathbb{H}$ with $\mathbf{p} \cdot \mathbf{q}$ or just $\mathbf{pq}$ and assume Hamilton's multiplication order, i.e. $\mathbf{ij} = \mathbf{k}$ holds. The left and right multiplication matrices, $(\cdot)^L$ and $(\cdot)^R$, are

$$\mathbf{q}^L = \begin{bmatrix} \eta\mathbf{1} + \boldsymbol{\epsilon}^\times & \boldsymbol{\epsilon} \\ -\boldsymbol{\epsilon}^T & \eta \end{bmatrix} \text{ and } \mathbf{q}^R = \begin{bmatrix} \eta\mathbf{1} - \boldsymbol{\epsilon}^\times & \boldsymbol{\epsilon} \\ -\boldsymbol{\epsilon}^T & \eta \end{bmatrix}, \tag{29}$$

where $\mathbf{1}$ is the 3×3 identity matrix. The quaternion *compound* operators, $(\cdot)^\oplus$, $(\cdot)^+$ of Barfoot et al. [5], are precisely $(\cdot)^L$ and $(\cdot)^R$. Given a unit-length quaternion, $\mathbf{q} \in S^0(\mathbb{H})$, the *complex conjugate* operator coincides with multiplicative inverse operator,

$$\mathbf{q}^{-1} = \begin{bmatrix} -\boldsymbol{\epsilon} \\ \eta \end{bmatrix}. \tag{30}$$

The Lie algebra of $S^0(\mathbb{H})$ consists exactly of the pure imaginary quaternions. We pick the usual basis, $\mathcal{B} := (\mathbf{i}, \mathbf{j}, \mathbf{k})$ and represent this Lie algebra's vectors with coordinate vectors, $\boldsymbol{\phi} \in \mathbb{R}^3$. To convert from three to four vectors and back we use the matrices

$$\mathbf{V} = \begin{bmatrix} \mathbf{1} \\ \mathbf{0}^T \end{bmatrix}, \text{ implying } \mathbf{V}\boldsymbol{\phi} = \begin{bmatrix} \boldsymbol{\phi} \\ 0 \end{bmatrix}, \text{ and } \mathbf{W} = \mathbf{V}^T, \text{ implying } \mathbf{WV} = \mathbf{1}, \tag{31}$$

which correspond to the matrices introduced after (20) and (22d) respectively as the Jacobians of the exponential map, at zero, and logarithm map, at identity.

We will identify a unit-quaternion $\mathbf{q}$ with the formal rotation

$$\mathrm{Rot}_{\mathbf{q}} : \mathbb{R}^3 \to \mathbb{R}^3, \mathbf{x} \mapsto \mathbf{W}(\mathbf{q}(\mathbf{Vx})\mathbf{q}^{-1}). \tag{32}$$

The proper orthogonal matrix, $\mathbf{C} \in$ SO(3), associated with the same rotation by $\forall_{\mathbf{x} \in \mathbb{R}^3} \mathbf{Cx} = \mathrm{Rot}_{\mathbf{q}}(\mathbf{x})$ may be computed using

$$\mathbf{q}^L \mathbf{q}^{-1^R} = \begin{bmatrix} \mathbf{C} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix}. \tag{33}$$

This is precisely the matrix $\mathbf{C}$ defined in (22d).

This Lie group's $\log$ and $\exp$ functions in coordinates on respectively into $\mathbb{R}^3$ can be calculated the following way

$$\exp(\boldsymbol{\phi}) = \begin{cases} \boldsymbol{\iota} & , \boldsymbol{\phi} = \mathbf{0} \\ \begin{bmatrix} \sin(\phi)\mathbf{a} \\ \cos\phi \end{bmatrix} & , \boldsymbol{\phi} \neq \mathbf{0} \end{cases}, \quad \log(\boldsymbol{q}) = \begin{cases} \mathbf{0} & , \boldsymbol{q} = \boldsymbol{\iota} \\ \frac{\arccos\eta}{\sqrt{1-\eta^2}}\boldsymbol{\epsilon} & , \boldsymbol{q} \neq \pm\boldsymbol{\iota} \\ \text{undefined} & , \boldsymbol{q} = -\boldsymbol{\iota} \end{cases}, \tag{34}$$

where $\boldsymbol{\phi}$ is 3×1, $\phi := \sqrt{\boldsymbol{\phi}^T\boldsymbol{\phi}}$, and $\mathbf{a} := \boldsymbol{\phi}/\phi$.

H. Derivatives of quaternion B-splines

In this section we give the time derivative and Jacobian formulas specific to our formulation for curves over $S^0(\mathbb{H})$. The B-spline time derivatives we calculated for the general case in (13) apply immediately to a unit-length quaternion curve, $\mathbf{q}(t) = [\boldsymbol{\epsilon}(t)\,\eta(t)]^T$, giving us directly quantities like $\dot{\boldsymbol{\epsilon}}(t)$, $\ddot{\boldsymbol{\epsilon}}(t)$, $\dot{\eta}(t)$, $\ddot{\eta}(t)$, etc. However, when we use such a curve to represent rotations as part of a physical model, we also need expressions for the angular velocity, $\boldsymbol{\omega}$, and angular acceleration, $\boldsymbol{\alpha}$, of a rotating frame.

To derive formulas for $\boldsymbol{\omega}$ or $\boldsymbol{\alpha}$, as physical notions, we need to specify how we interpret the formal rotation (32) in a physical context. We will identify the rotation from frame $\underset{\rightarrow}{\mathcal{F}}_a$, an inertial frame, to $\underset{\rightarrow}{\mathcal{F}}_b$, one attached to a rigid body, by the unit-length quaternion, $\mathbf{q}_{ba}$, such that

$$\forall_{\mathbf{v}\in\mathbb{E}^3} \mathrm{Rot}_{\mathbf{q}_{ba}}(\mathbf{v}_a) = \mathbf{v}_b, \tag{35}$$

where $\mathbf{v}_a$ and $\mathbf{v}_b$ denote the coordinates of a vector $\mathbf{v}$ in Euclidean space ($\mathbb{E}$) with respect to $\underset{\rightarrow}{\mathcal{F}}_a$ and $\underset{\rightarrow}{\mathcal{F}}_b$ respectively. The quaternion, $\mathbf{q}_{ba}$, is only uniquely defined up to negation and thus all physical equations need to be invariant with respect to negation of this variable.

1. *Angular velocity*

For the angular velocity, $\boldsymbol{\omega}$, of $\underset{\rightarrow}{\mathcal{F}}_b$ with respect to $\underset{\rightarrow}{\mathcal{F}}_a$ given by a quaternion-valued curve through time $\mathbf{q}_{ba}(t) = \begin{bmatrix}\boldsymbol{\epsilon}(t) & \eta(t)\end{bmatrix}^T$ we get (omitting the time parameter for clarity):

$$\boldsymbol{\omega}_a = -2\mathbf{W}(\mathbf{q}_{ba}^{-1}\dot{\mathbf{q}}_{ba}) = -2((\eta\dot{\boldsymbol{\epsilon}} - \dot{\eta}\boldsymbol{\epsilon}) - \boldsymbol{\epsilon}^{\times}\dot{\boldsymbol{\epsilon}}), \tag{36}$$

when expressed in $\underset{\rightarrow}{\mathcal{F}}_a$ and

$$\boldsymbol{\omega}_b = -2\mathbf{W}(\dot{\mathbf{q}}_{ba}\mathbf{q}_{ba}^{-1}) = -2((\eta\dot{\boldsymbol{\epsilon}} - \dot{\eta}\boldsymbol{\epsilon}) - \dot{\boldsymbol{\epsilon}}^{\times}\boldsymbol{\epsilon}), \tag{37}$$

when expressed in $\underset{\rightarrow}{\mathcal{F}}_b$.

2. Angular acceleration

It follows for the angular acceleration $\boldsymbol{\alpha}$, omitting the subscripts for $\mathbf{q}_{ba}$ for brevity:

$$\boldsymbol{\alpha}_b = \dot{\boldsymbol{\omega}}_b = -2\mathbf{W}(\frac{\mathrm{d}}{\mathrm{d}t}(\dot{\mathbf{q}}\mathbf{q}^{-1})) = -2\mathbf{W}(\dot{\overline{\mathbf{q}^{-1}}}\dot{\mathbf{q}} + \ddot{\mathbf{q}}\mathbf{q}^{-1}) = -2\mathbf{W}(\ddot{\mathbf{q}}\mathbf{q}^{-1}) \tag{38}$$

$$= -2((\eta\ddot{\boldsymbol{\epsilon}} - \ddot{\eta}\boldsymbol{\epsilon}) - \ddot{\boldsymbol{\epsilon}}^{\times}\boldsymbol{\epsilon}) \tag{39}$$

3. Jacobians of quaternion B-splines with respect to the control vertices

To use the results from section III D 2 we will need the specific matrices for the unit-length quaternions. First, we provide the Jacobian of the $\log(\cdot)$ function,

$$\mathbf{L}(\mathbf{q}) = \begin{cases} \mathbf{1} & , \mathbf{q} = \boldsymbol{\iota} \\ \mathbf{1} + \boldsymbol{\phi}^{\times} + \left(1 - \frac{\phi}{\tan\phi}\right)\mathbf{a}^{\times}\mathbf{a}^{\times} & , \mathbf{q} \neq \boldsymbol{\iota} \end{cases}, \tag{40}$$

where we have used $\boldsymbol{\phi} := \log(\mathbf{q})$, $\phi := \|\boldsymbol{\phi}\|$, and $\mathbf{a} := \boldsymbol{\phi}/\phi$.

Next, we provide the Jacobian of the exponential map.

$$\mathbf{S}(\boldsymbol{\theta}) := \begin{cases} \mathbf{1} & , \boldsymbol{\theta} = \mathbf{0} \\ \mathbf{1} - \frac{1}{\theta}\sin^2\theta\,\mathbf{a}^{\times} + \frac{1}{\theta}(\theta - \frac{1}{2}\sin 2\theta)\mathbf{a}^{\times}\mathbf{a}^{\times} & , \boldsymbol{\theta} \neq \mathbf{0} \end{cases}. \tag{41}$$

with $\theta := \|\boldsymbol{\theta}\|$ and $\mathbf{a} := \boldsymbol{\theta}/\theta$.

4. Jacobians of angular velocity

The Jacobians of the angular velocity with respect to minimal perturbations of the control vertices are also required. So let again $1 \leq j < O$ and $k := s(i) + j$,

$$\frac{\partial\boldsymbol{\omega}_a}{\partial\boldsymbol{\phi}_k} = -2\left(\frac{\partial\eta}{\partial\boldsymbol{\phi}_k}\dot{\boldsymbol{\epsilon}} + \eta\frac{\partial\dot{\boldsymbol{\epsilon}}}{\partial\boldsymbol{\phi}_k} - \frac{\partial\dot{\eta}}{\partial\boldsymbol{\phi}_k}\boldsymbol{\epsilon} - \dot{\eta}\frac{\partial\boldsymbol{\epsilon}}{\partial\boldsymbol{\phi}_k} + \dot{\boldsymbol{\epsilon}}^{\times}\frac{\partial\boldsymbol{\epsilon}}{\partial\boldsymbol{\phi}_k} - \boldsymbol{\epsilon}^{\times}\frac{\partial\dot{\boldsymbol{\epsilon}}}{\partial\boldsymbol{\phi}_k}\right). \tag{42}$$

5. Jacobians of angular acceleration

For the angular acceleration it yields analogously

$$\frac{\partial\boldsymbol{\alpha}_a}{\partial\boldsymbol{\phi}_k} = -2\left(\frac{\partial\eta}{\partial\boldsymbol{\phi}_k}\ddot{\boldsymbol{\epsilon}} + \eta\frac{\partial\ddot{\boldsymbol{\epsilon}}}{\partial\boldsymbol{\phi}_k} - \frac{\partial\ddot{\eta}}{\partial\boldsymbol{\phi}_k}\boldsymbol{\epsilon} - \ddot{\eta}\frac{\partial\boldsymbol{\epsilon}}{\partial\boldsymbol{\phi}_k} + \ddot{\boldsymbol{\epsilon}}^{\times}\frac{\partial\boldsymbol{\epsilon}}{\partial\boldsymbol{\phi}_k} - \boldsymbol{\epsilon}^{\times}\frac{\partial\ddot{\boldsymbol{\epsilon}}}{\partial\boldsymbol{\phi}_k}\right). \tag{43}$$

I. Associating a unit-quaternion time series to a rotation time series

To fit a unit-quaternion curve to a time series of noisy rotations can be a challenge. For the fitting procedure described in III E one has to first convert it to a unit-quaternion time series. We will refer to this step in the initialization process as *the association step*.

The association of unit-length quaternions to rotations is ambiguous as (32) maps any pair of antipodal unit-quaternions to the same rotation. Choosing the wrong quaternion of the pair during association results in jumps in the unit-length quaternion curve and can cause the optimization to diverge. The obvious solution is to retrieve the minimum-length sequence of corresponding quaternions, measured as sum of distances of all adjacent pairs.

This can be approximated by starting from one side of the time series and sequentially picking one of the two unit-length quaternions with minimal distance. However, this strategy can perform badly in the presence of outliers. Our current implementation iterates over the rotations but chooses the quaternion minimizing the distance to the last three quaternions picked before. This worked well enough in our experiments, except as noted when we tested with high sensor noise (Figure Figure 8 on page 30).

IV. Experiments

In this section we apply the state representation derived above to the spacecraft attitude estimation problem. The goal is to estimate the attitude of a reference frame attached to the vehicle body, $\underrightarrow{\mathcal{F}}_B$, with respect to an inertial frame, $\underrightarrow{\mathcal{F}}_I$, over a time interval of interest, $\mathcal{T} := [t_0, t_0 + T]$. We would like to estimate a unit-length quaternion trajectory describing the rotation from the inertial frame to the body frame, $\mathbf{q}_{BI}(t)$, over $\mathcal{T}$.

The vehicle may have bearing sensors (we will consider one or two). A bearing sensor takes measurements at discrete time instances, $(t_m)_{m=1}^{M} \in \mathcal{T}$. The sensor's frame $\underrightarrow{\mathcal{F}}_S$ is rigidly attached to the vehicle body frame. We assume that the orientation of the sensor frame with respect to the body frame, $\mathbf{C}_{SB}$, is known. Let $\mathbf{b}_I^{m\ell}$ be the known bearing of beacon ℓ (in case of sun sensors or star trackers) or the magnetic field (in case of a magnetometers) at time t_m, expressed in the inertial frame. A measurement of this bearing, written $\mathbf{y}_{m\ell}$, is modelled as

$$\mathbf{y}_{m\ell} = \mathbf{h}\left(\mathbf{C}_{SB}\mathbf{C}_{\mathbf{q}_{BI}(t_m)}\mathbf{b}_I^{m\ell}\right) + \mathbf{n}_{m\ell}, \qquad \mathbf{n}_{m\ell} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{R}_{m\ell}\right), \tag{44}$$

where the bearing vector is first rotated into the sensor frame, $\mathbf{h}(\cdot)$ is a nonlinear observation model, $\mathbf{C}_{\mathbf{q}}$ is the proper orthogonal matrix given by a quaternion, $\mathbf{q}$, as in (33), here acting as the direction cosine matrix from the inertia frame to the body frame, and $\mathbf{n}_{m\ell}$ is zero-mean Gaussian noise with covariance $\mathbf{R}_{m\ell}$.

Following the standard practice of maximum likelihood estimation, we define the error term associated with this measurement to be

$$\mathbf{e}_{m\ell} := \mathbf{y}_{m\ell} - \mathbf{h}\left(\mathbf{C}_{SB}\mathbf{C}_{\mathbf{q}_{BI}(t_m)}\mathbf{b}_I^{m\ell}\right). \tag{45}$$

These errors contribute to the term J_y in our overall objective function,

$$J_y := \frac{1}{2}\sum_{m=1}^{M}\mathbf{e}_{m\ell}^T\mathbf{R}_{m\ell}^{-1}\mathbf{e}_{m\ell}. \tag{46}$$

Below we present simulated experiments using this common setup for the exteroceptive measurements, they differ in terms of bearing sensors and dynamics model used. Based on the latter we distinguish two types of experiments. In the first type of experiments, we use measured angular velocities from a gyroscope. In the second, we use a dynamics model based on Euler's equation and the vehicle inertia matrix.

A. Attitude estimation based on gyroscope measurements

In Experiment 1, we consider an estimation problem that neglects analytical vehicle dynamics in favor of measured dynamics from a three-axis gyroscope. For simplicity, we choose to place our vehicle body frame at the center of our gyroscope measurement frame. The gyroscope measurement model follows the one commonly used in robotics [35, 49],

$$\boldsymbol{\varpi}_g = \boldsymbol{\omega}(t_g) + \mathbf{b}(t_g) + \mathbf{n}_{g\omega}, \qquad \mathbf{n}_{g\omega} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_{g\omega}), \tag{47a}$$

$$\dot{\mathbf{b}}(t) = \mathbf{w}_b(t), \qquad \mathbf{w}_b \sim \mathcal{GP}\left(\mathbf{0}, \delta(t-t')\mathbf{Q}_b\right), \tag{47b}$$

where $\boldsymbol{\varpi}_g$ denotes the angular velocity measurement at t_g and $\boldsymbol{\omega}$ is the angular velocity of the body as seen from the inertial frame but expressed in the body frame. It is related to the quaternion trajectory $\mathbf{q}_{BI}(t)$ via (37). The gyroscope measurements are assumed to be subject to both zero-mean Gaussian measurement noise, $\mathbf{n}_{g\omega}$, and a slowly evolving bias, $\mathbf{b}(t)$[67]. We will model the bias with a traditional B-spline function in $\mathbb{R}^3$ in every experiment with the same order as the attitude unit-quaternion B-spline. The error for a single gyroscope measurement may be written as

$$\mathbf{e}_{g\omega} := \boldsymbol{\varpi}_g - \boldsymbol{\omega}(t_g) - \mathbf{b}(t_g), \tag{48}$$

which becomes a term, J_ω, in our objective function:

$$J_\omega := \frac{1}{2}\sum_{g=1}^{\mathcal{G}}\mathbf{e}_{g\omega}^T\mathbf{R}_{g\omega}^{-1}\mathbf{e}_{g\omega}. \tag{49}$$

The bias motion error may be written as

$$\mathbf{e}_b(t) := \dot{\mathbf{b}}(t). \tag{50}$$

This error contributes to our objective function as

$$J_b := \frac{1}{2}\int_{t_0}^{t_K} \mathbf{e}_b^T(t)\mathbf{Q}_b^{-1}\mathbf{e}_b(t)\,\mathrm{d}t. \tag{51}$$

Together, (46), (49), and (51) define the combined objective function, $J := J_y + J_\omega + J_b$. Let $\boldsymbol{\mathcal{Q}}$ be the stacked matrix of quaternion control vertices and $\mathbf{c}_b$ the stacked vector of bias spline control vertices. The control vertices for the maximum likelihood estimate of the trajectories $\mathbf{q}_{BI}(t)$ and $\mathbf{b}(t)$, $\boldsymbol{\mathcal{Q}}^\star, {\mathbf{c}_b}^\star$, can then approximately be found by minimizing $J(\boldsymbol{\mathcal{Q}}, \mathbf{c}_b)$,

$$(\boldsymbol{\mathcal{Q}}^\star, {\mathbf{c}_b}^\star) = \operatorname*{argmin}_{\boldsymbol{\mathcal{Q}}, \mathbf{c}_b} J(\boldsymbol{\mathcal{Q}}, \mathbf{c}_b)\,. \tag{52}$$

B. Attitude estimation based on Euler's equation

In this type of experiments, we consider a spacecraft attitude estimation problem that uses a vehicle dynamics model based on Euler's equation. The dynamics model is, expressed in the body frame,

$$\mathbf{I}\dot{\boldsymbol{\omega}}(t) + \boldsymbol{\omega}^\times(t)\mathbf{I}\boldsymbol{\omega}(t) = \mathbf{u}(t) + \mathbf{w}(t), \tag{53}$$

where t is time, $\mathbf{I}$ is the vehicle inertia matrix, $\boldsymbol{\omega}(t)$ is again the angular velocity of the body as seen from the inertial frame, $\mathbf{u}(t)$ is a known control input, and $\mathbf{w}$ is a zero-mean white Gaussian process with covariance $\mathbf{Q}_d$,

$$\mathbf{w} \sim \mathcal{GP}\left(\mathbf{0}, \delta(t-t')\mathbf{Q}_d\right), \tag{54}$$

where $\delta(\cdot)$ is Dirac's delta function. Following Furgale et al. [24], we define the error for the dynamics model at time t to be

$$\mathbf{e}_d(t) := \mathbf{I}\dot{\boldsymbol{\omega}}(t) + \boldsymbol{\omega}^\times(t)\mathbf{I}\boldsymbol{\omega}(t) - \mathbf{u}(t). \tag{55}$$

The resulting term in our objective function is

$$\mathbf{J}_d := \frac{1}{2}\int_{t_0}^{t_K} \mathbf{e}_d^T(t)\mathbf{Q}_d^{-1}\mathbf{e}_d(t)\,\mathrm{d}t. \tag{56}$$

Together, (46) and (56) define the combined objective function, $J := J_y + J_d$. If $\mathcal{Q}$ is, again, the stacked matrix of quaternion control vertices. The trajectory estimate is then approximated by the J-minimizing quaternion control vertices,

$$\mathcal{Q}^\star = \underset{\mathcal{Q}}{\operatorname{argmin}}\, J(\mathcal{Q}). \tag{57}$$

C. Simulation configurations

We simulated the attitude trajectory of a spacecraft at 350 km altitude, with a 35 degree inclination, and an initial angular velocity of square norm 0.5 deg/s. This configuration is intentionally similar to those used in [63] and [13] so that the result can be directly compared. To stress test the dynamic model based approach we also simulated trajectories with a sinusoidal thruster turned on. The thruster's sinusoidal torque, $\boldsymbol{\tau}$, is simulated at a time t for a given thruster factor, f_τ, as follows,

$$\mathbf{a}_\tau := \begin{bmatrix} 0.001 & 0.0005 & 0.00075 \end{bmatrix}^T \text{Nm}, \tag{58}$$

$$\boldsymbol{\omega}_\tau := \begin{bmatrix} 0.1 & 0.05 & 0.075 \end{bmatrix}^T \frac{\text{rad}}{\text{s}}, \tag{59}$$

$$\boldsymbol{\phi}_\tau := \begin{bmatrix} 0 & \frac{\pi}{2} & -\frac{\pi}{4} \end{bmatrix}^T \text{rad}, \tag{60}$$

$$\boldsymbol{\tau}_0 := \begin{bmatrix} 0.005 & -0.005 & 0.000 \end{bmatrix}^T \text{Nm}, \tag{61}$$

$$\boldsymbol{\tau}(t) := f_\tau \left(\mathbf{a}_\tau \sin(t\boldsymbol{\omega}_\tau + \boldsymbol{\phi}_\tau) + \boldsymbol{\tau}_0\right), \tag{62}$$

where the sinus is applied componentwise. This torque is used as the input $\mathbf{u}$ in (53). The angular acceleration induced by the thruster is determined by the assumed inertia tensor of the spacecraft in the same frame:

$$\mathbf{I} := \operatorname{diag}(\begin{bmatrix} 27 & 17 & 25 \end{bmatrix})\text{kgm}^2 \tag{63}$$

$$\boldsymbol{\alpha}_\tau(t) := \mathbf{I}^{-1}\boldsymbol{\tau}(t) \tag{64}$$

We chose our system process noise, described in (54), to be defined by $\mathbf{Q}_d = (1\mu\text{Nm})^2\frac{1}{\text{s}}\mathbf{1}$. We simulated a three-axis magnetometer (TAM), a sun sensor, and a rate gyroscope. They were all sampled at 1Hz, as in [63] ([13] use 10Hz). We used the following parameters for the sensors:

- Gyroscope : $\mathbf{R}_{g\boldsymbol{\omega}} = (0.3\mu\frac{\text{rad}}{\text{s}})^2\mathbf{1}$, $\mathbf{Q}_b = (3 * 10^4\mu\frac{\text{rad}}{\text{s}^2})^2\frac{1}{\text{s}}\mathbf{1}$ (Section (47))
- Magnetometer : $\mathbf{R}_B = (50\text{n T})^2\mathbf{1}$ (Section (44))

- Sun sensor: $\mathbf{R}_S = (5\text{m rad})^2\mathbf{1}$ (Section (44))

The simulated gyroscope bias was started at $0\text{deg}/\text{h}$ or $1000\text{deg}/\text{h}$ depending on the experiment. To the best of our knowledge, these parameters are identical to [63] and [13], except that they did not simulate a sun sensor. For some tests we introduced a noise factor $F \in [1, 100]$ applied to the standard deviation of the sensor measurements and the gyroscope bias process noise. For every simulation configuration we simulated an ensemble of 100 different seeds for the pseudo random number generators used to simulate process and sensor noise.

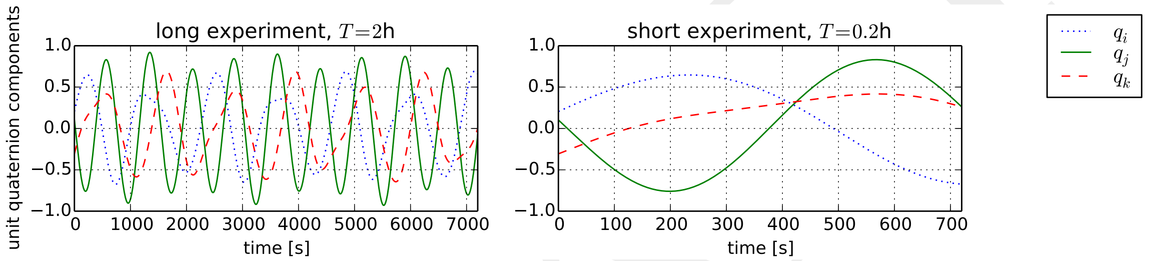


Fig. 1 A typical simulated attitude trajectory plotted as the three imaginary components of $\mathbf{q}_{BI}$ over the longest run time $T = 2$h and a zoomed plot of the first 0.2h.

The typical simulated attitude trajectory is depicted in Figure 1 on page 25. It depicts the three imaginary components of the attitude quaternion, $\mathbf{q}_{BI}$, over our maximal duration $T = 2$h and zooms in to the first $[0, 0.2]$h interval as used in most of the short experiments.

D. Estimators

For our comparison we have implemented the proposed spline estimators for continuous-time estimation capable of integrating all sensor output, including the gyroscope, as described in IV A, and the dynamics model, as described in IV B with arbitrary spline order (starting at 2 corresponding to a polynomial order of 1, i.e. a “linear spline”). We will refer to these estimators with (label, marker) on all plots:

- (*splXX*, *XX*) : our quaternion spline implementation where XX is the B-spline order;
- (*MEKF*, *K*) : a multiplicative EKF implementation; and
- (*splVan*, *V*) a modified linear spline that uses the symplectic integration scheme described in [63] to calculate its angular velocity. Although our optimization algorithm will certainly use a different path

through the parameter space of the spline the result should be (up to the solver precision) the same as of a full implementation of the estimator described in [63].

E. Estimator configurations

As estimator configurations we consider a specification of

- the prior attitude distribution as normal distribution on SO(3), $\mathcal{N}(\boldsymbol{\mu}_A, \mathbf{P}_A)$;
- the prior bias distribution as normal distribution on $\mathbb{R}^3$, $\mathcal{N}(\boldsymbol{\mu}_b, \mathbf{P}_b)$;
- the observation duration T;
- the sensor period—for all sensors (adjustable by subsampling the simulated sensor readings);
- the knot spacing for the splines; and
- the set of sensors used (rate gyro, sun, TAM).

The system dynamics model is used if and only if the gyroscope is not used.

To initialize the spline batch estimators we start with calculating a quaternion sequence using different strategies, depending on the sensor configuration:

- If both bearing sensors are used, we use the q-Method [65, pp. 426–428] to get a quaternion for each measurement time.
- Otherwise, if the rate gyroscope is available we integrate its angular velocity measurements starting at the prior's mean. To mitigate the drift of the gyroscope integration in long experiments, we initialize the full spline in subbatches of about 10 minutes. For each subbatch, we integrate the gyroscope, perform a full estimation with all sensor data, then repeat the process for the next subbatch using the final value of the previous subbatch at the start of the integration time.
- If neither are available, we use a constant sequence equal to the prior's mean.

In each case, we roughly fit the quaternion spline to this sequence to initialize the nonlinear optimization (Section III E), then run a full batch optimization.

To employ the system model in the second type of experiments we used Simpson's rule and twice as many evaluations for the numeric integration (Section III F) as knots in the spline.

For the prior attitude we chose the mean, $\boldsymbol{\mu}_A$, to be 120deg off ground truth. and the covariance matrix $\mathbf{P}_A = (180\text{deg})^2\mathbf{1}$ and $\mathbf{P}_b = (0.2\text{deg/h})^2\mathbf{1}$. The bias mean is always $\mathbf{0} : \mathbb{R}^3$. We use $T = 0.2$h as the observation time and $1s$ as the default sensor period (corresponding to the 1Hz simulation frequency). As the default knot spacing we chose the sensor period for the linear splines (B-spline and Vandersteen) and twice the sensor period for all other splines as this gave the best results in each case. All experiment descriptions in the results section will be relative to this default configuration.

V. Results

We defined several experiments, by picking a simulator configuration and an estimator configuration up to one open parameter. This parameter will be varied in a specific interval and make up the abscissa of the 2d-result plots. As ordinates, we plot mean and standard deviation over the simulated ensemble for a temporally averaged angular distance between ground truth and estimated attitude. To allow the different estimators time to converge, this temporal average is taken over the last ΔT seconds of the observation time T. We will denote this measure with MADE(ΔT) for mean angular distance error.

A. Gyroscope based

First the experiments based on gyroscope measurements, corresponding to IV A.

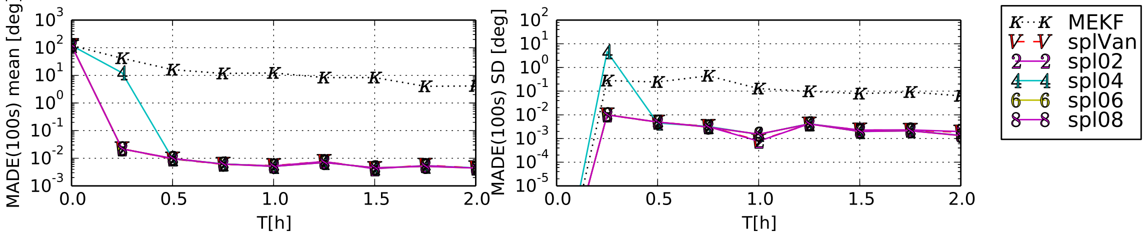


Fig. 2 MADE(100s) over the observation time T given a simulated initial bias of 1000deg/h. $T \in [0, 2]$h. Sensors: rate gyroscope, magnetometer.

Figure 2 on page 27 shows the results for the extreme initial bias test case of [63]. This experiment uses an initial bias of 1000deg/h and only a gyroscope and magnetometer (no sun sensor). It shows how the

MEKF recovers very slowly from the bad prior (120deg away from ground truth) while all spline estimators have good results after 0.5h, at the latest. The 4th-order spline has more problems with the high bias error then the linear ones and the 6th-order. The former is probably mainly an effect of the fact that the linear splines have twice as much knots. Qualitatively these results fit the results of [63] as far as they overlap. However there are two major differences. First, in our experiment the MEKF was able to converge for all random seeds—just very slowly. And second, the limiting accuracy was about $5 * 10^{-3}$deg in our case and 10^{-3}deg in theirs. These differences are most likely caused by differences in the simulated sensor data as we do not know the initial angular velocity or the simulated system noise used in [63].

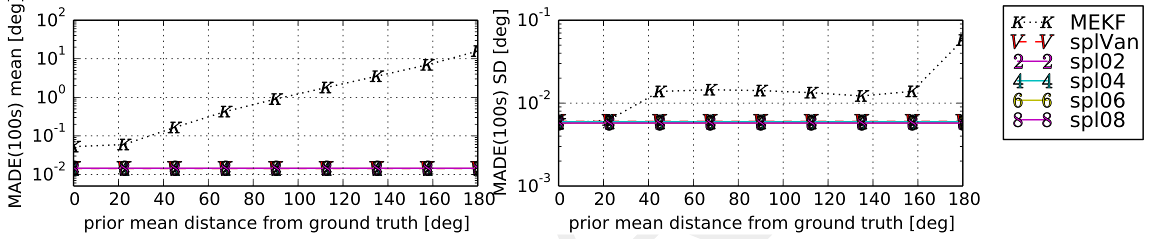


Fig. 3 MADE(100s) over the distance of the prior mean to ground truth. $T = 0.2$**h. Sensors: rate gyroscope, magnetometer.**

In Figure 3 on page 28 we show the response of each estimator to a bad prior mean after $T = 0.2$h of simulation time. Here, all batch estimators converge with essentially the same accuracy, while the Kalman filter has trouble converging as the prior mean failure gets large. The low variance of the MEKF measurements indicates that the MEKF's estimate is not only spoiled by noise but has systematic problems when facing a prior with a mean far off the truth.

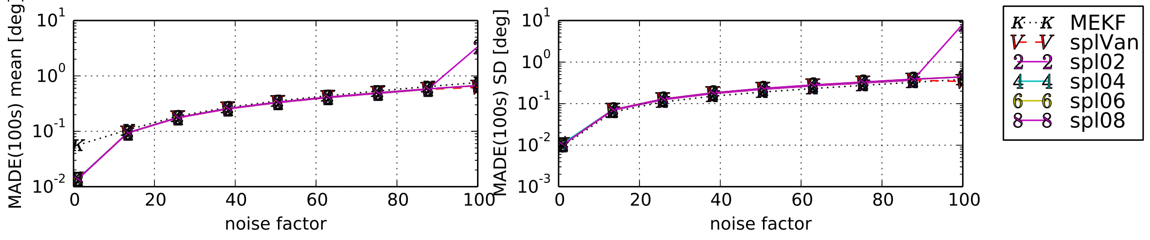


Fig. 4 MADE(100s) over a factor multiplying the standard deviation of all sensor noise. $T = 0.2$**h. Sensors: rate gyroscope, sun sensor, magnetometer.**

In Figure 4 on page 28 we test the effect of sensor noise on each estimator. The splines excel in low noise range but are on par with the MEKF when sensor noise increases. The 8th-order spline's bad performance for the highest noise case is caused by wrongly signed quaternions in the result of the quaternion association step (Section III I). This step gets more and more challenging when the sensor noise increases. The lower order splines share the associations but are better in recovering from these flaws in the rough fitting step (Section III E) and the later nonlinear optimization. The MEKF converges in every case due to the high sensor rate.

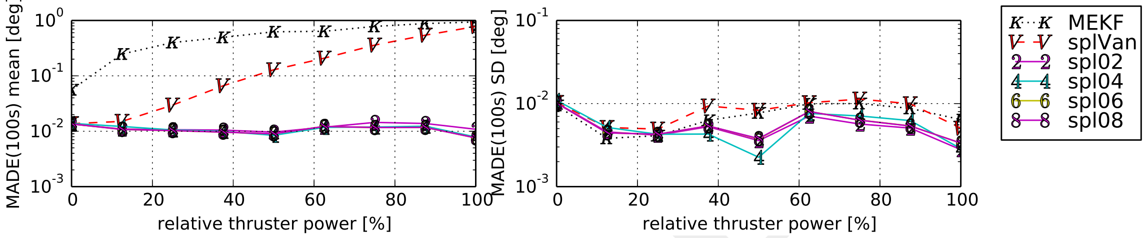


Fig. 5 MADE(100s) over the thruster's power factor in percentage. $T = 0.2$h. Sensors: rate gyroscope, sun sensor, magnetometer.

In Figure 5 on page 29 we tested the effect on higher angular velocity and acceleration due to activated thruster. Only the Kalman filter and the Vandersteen spline are noticeably affected by this addition. In the case of the Vandersteen spline, we suspect that this is due to the use of the symplectic integration scheme to approximate the angular velocity. In contrast, state derivatives are exact and analytical for all B-spline solutions.

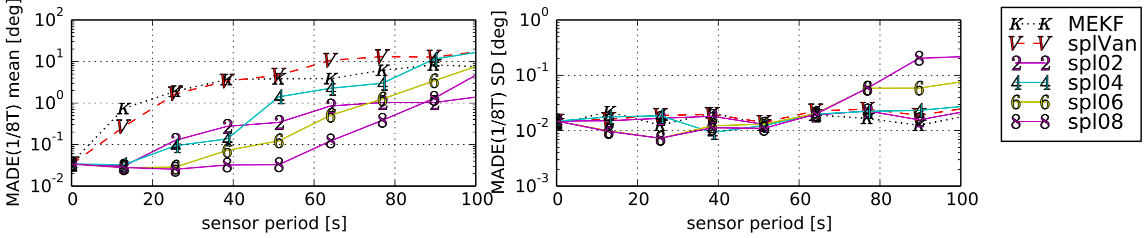


Fig. 6 MADE($\frac{1}{8}T$) over the sensors' period. T is determined by a fixed amount of 64 measurements per sensor. Sensors: rate gyroscope, sun sensor, magnetometer.

In Figure 6 on page 29 we examined the effect of the sensor rate (defined by its period). Again the Kalman filter and the Vandersteen spline are much more negatively affected. The similarity between Figure 6 on page 29 and Figure 5 on page 29 (especially in the beginning) is because the accuracy of the simulated sensor is not affected by the spacecraft's velocity; increasing the sensor period has virtually the same effect as speeding up the whole motion and leaving the period fixed. The effect of the thruster at 100% power roughly corresponds to the sensor period of 20 s. The 4th and 6th-order spline become worse than the linear spline (spline order 2) because they have only half the number of knots, and in this experiment they all reach their bounds of their *expressiveness* (the frequency of motion they can represent), because they have to model a much longer trajectory with a fixed number of knots.

B. Euler's equation based

Next we present the experiments based on Euler's equation. Here we cannot use the two linear spline versions we used in former section because their angular acceleration is zero or undefined which does not allow a straightforward application of the system model. In this section, we skip the first convergence experiment (corresponding to Figure 2 on page 27 in the last section), as we do not have a initialization method reliable enough for the extreme initial bias case. Apart from that, we first follow roughly the same program as for the gyroscope-based experiments.

In Figure 7 on page 30 we show the response of each estimator to a bad prior mean after $T = 0.2$h of simulation time.

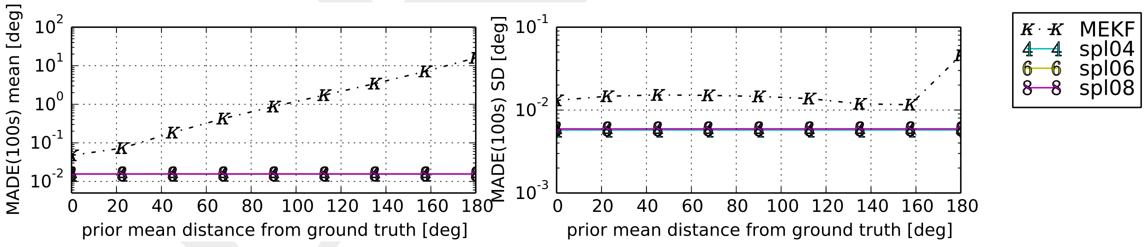


Fig. 7 MADE(100s) over the angular distance error of the prior. $T = 0.2$**h. Sensors: magnetometer.**

The Figure 7 on page 30 shows again how all splines perform superior to the MEKF.

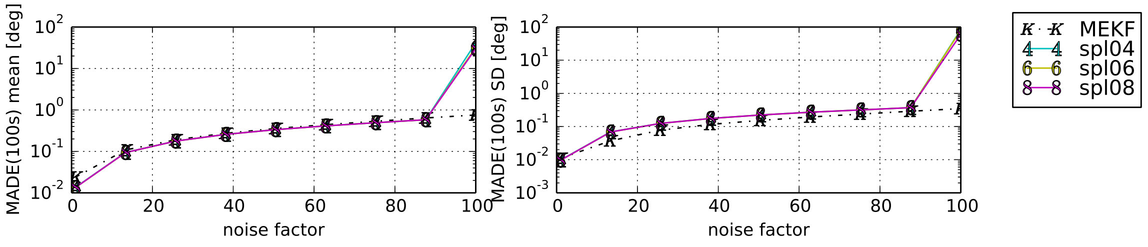


Fig. 8 MADE(100s) over a factor multiplying the standard deviation of all sensor noise. $T = 0.2$**h. Sensors: sun sensor, magnetometer.**

In Figure 8 on page 30 we test the effect of sensor noise on each estimator. The system-model-based spline estimators excel in the low noise range. Their bad performance for the highest noise case is caused by wrongly signed quaternions in the result of the quaternion association step (Section III I). In the case of such a bad initialization the system model pulls strongly towards an incorrect solution. When we use gyroscope measurements instead of the system model (Figure 4 on page 28), the estimator is able to find its way back to the correct solution.

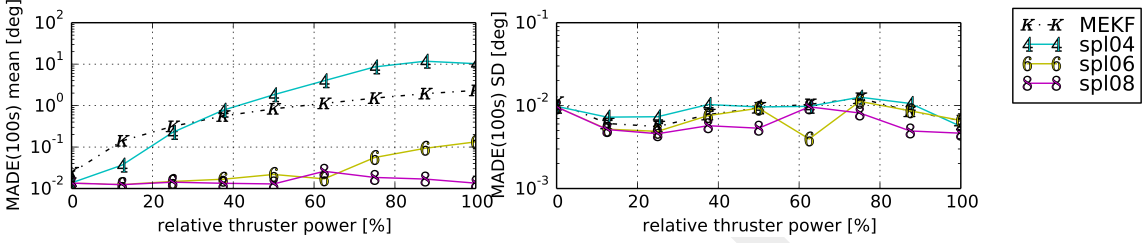


Fig. 9 MADE(100s) over the thruster's power factor. $T = 0.2$**h. Sensors: sun sensor, magnetometer.**

In Figure 9 on page 31 we compare the performance of the estimators with while varying the intensity of the thruster. Only the Kalman filter and the 4th-order spline are noticeably affected. Here the high order splines clearly excel. The poor performance of the 4th-order spline is caused by its bad compatibility with the ODE given by the system model, especially when the thruster is activated. This incompatibility pulls the estimator away from the good solutions as the low noise on the dynamics makes the estimator trust the system model more than the sensors. When we increase the noise $\mathbf{w}$ in the simulated system (53) this difference continuously decreases. We will analyze that effect further in Figure 14 on page 36 and Figure 11 on page 33.

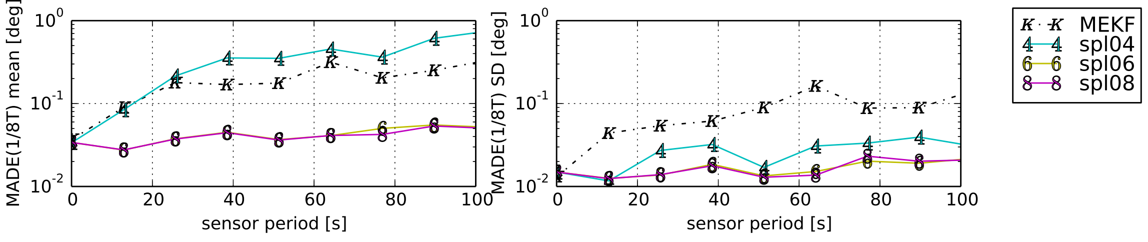


Fig. 10 MADE($\frac{1}{8}T$**) over the sensors' period.** T **is determined by a fixed amount of 64 measurements per sensor. Sensors: sun sensor, magnetometer.**

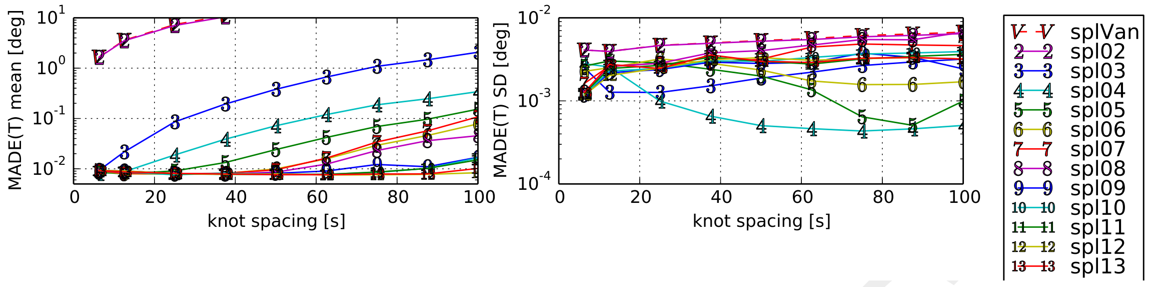
In Figure 10 on page 31 we tested the effect of the sensor rate (as in Figure 6 on page 29). Again, the

Kalman filter is much more negatively affected than the high order splines. Figure 9 on page 31 is again quite similar to the first 20 seconds. The bad effect on the 4th-order spline is more severe than in the gyroscope experiment. We believe that this is again caused by the incompatibility between cubic B-splines and the system model. This difference becomes small when we increase the system noise.

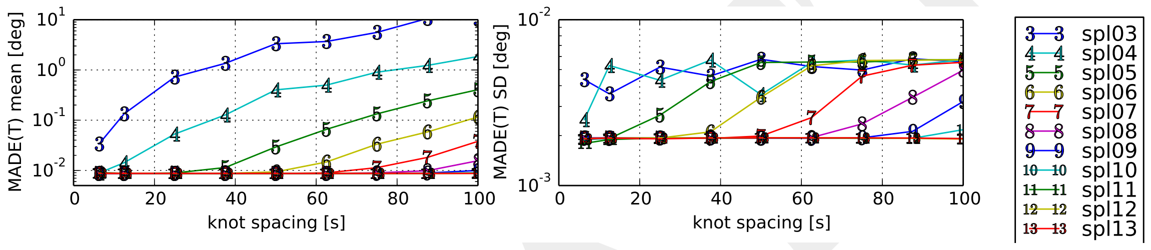
C. Discussion

The most surprising discovery we had while evaluating these experiments was the strong influence of the spline order on the estimation results. The numbers of parameters for a B-spline with N usable segments (between the knots t_O and t_{O+N}) and spline order O is proportional to $O + N$. These parameters, N and O, are the control values through which one may influence the *expressiveness* of a B-spline curve. Because the number of parameters increases linearly with $O + N$, we could be led to believe that the assumption that the expressiveness is also proportional to that. However, for a given system (represented mathematically by a stochastic ODE), the leap from cubic to quintic polynomials (requiring only two more control vertices for any length curve) may result in a much better fit to the actual motions produced by the system. B-splines are C^{O-1} continuous at the knots and infinitely differentiable everywhere else. This means that increasing the number of knots, N, creates more times at which the curve is not infinitely differentiable, whereas leaving N fixed and increasing the spline order, O, increases how often the curves are differentiable everywhere. For our spacecraft dynamics, the 4th-order spline is a measurably worse fit than a 6th-order spline.

To further investigate the role of spline order, we came up with the following series of experiments.



(a) with gyroscope



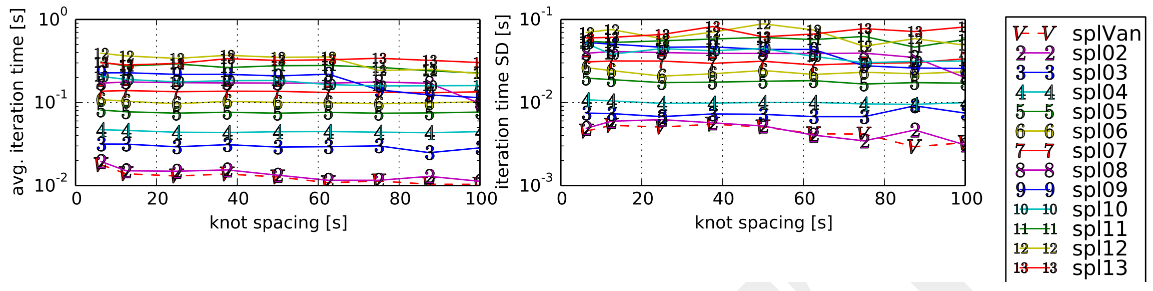
(b) with system model

Fig. 11 MADE(T) over the time between knots. $T = 0.2$h. Sensors: sun sensor, magnetometer.

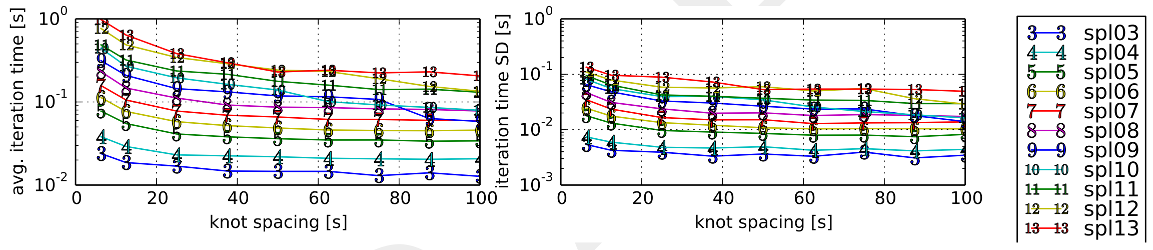
In Figure 11 on page 33 we assessed the estimation accuracy while changing the uniform knot spacing, Δt_{knots}, for different spline orders up to 13. This implies dividing the run time, $T = 0.2h$, into different numbers of segments, $N = \text{ceil}(720s/\Delta t_{\text{knots}})$, ranging from 115 to 8. The higher order splines (≥ 12) were more difficult to initialize for high knot spacing (above 80s) as mentioned in III E and required to rise the acceleration penalty. The drastic effect of the order catches the eye. In the system model experiment, increasing the order from 4 to 5 (adding one quaternion more to the parameters) corresponds roughly to reducing the number of segments from 20 to 10 yielding a reduction of parameters from 24 to 15 quaternions. In the gyroscope experiment the effect is lower, but still significant; the step from 4 to 5 corresponds to a decrease in segments from 20 to 12.

In the high-accuracy range the result is even more extreme. For example, in the gyroscope-based experiment, a 4th-order spline requires at least 8 times more segments to achieve the same accuracy as a 12th-order spline. In the system model experiment it is 16 times more.

In these plots, the linear splines seem even worse than in the other experiments because they are using the same knot spacing as the other splines. Recall that in the other experiments, they used twice as many knots.



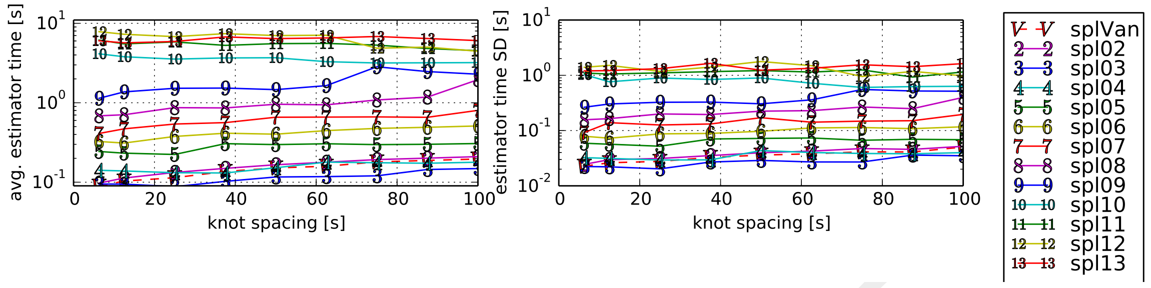
(a) with gyroscope



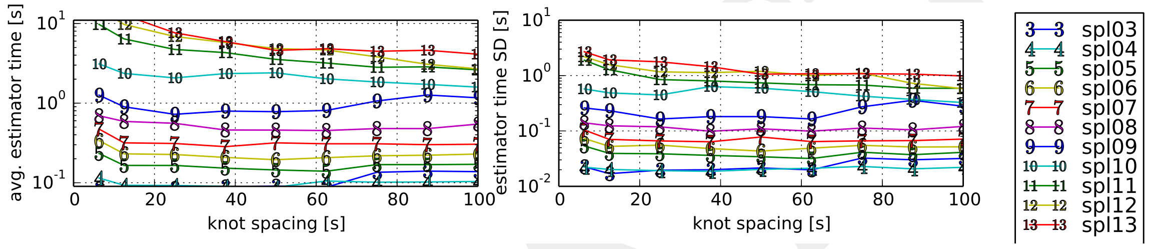
(b) with system model

Fig. 12 Time for one iteration over knot spacing on a single Intel(R) Xeon(R) E5 core at 3.60GHz with 2 * 12800 MB/s peak memory transfer rate. $T = 0.2$**h, yielding** 720 **readings per sensor. Sensors: sun sensor, magnetometer.**

Of course this needs to be compared with the corresponding computational cost. As the Jacobian evaluation has a big impact it is hard to theoretically derive the computational complexity. Thus we only provide single core CPU time measurements on a workstation in Figure 12 on page 34. It shows surprisingly little effect of the number of segments on a single iteration. The majority of time is spent on the evaluation of the Jacobians, which only depends on the number of measurements (fixed here) and spline order, O.



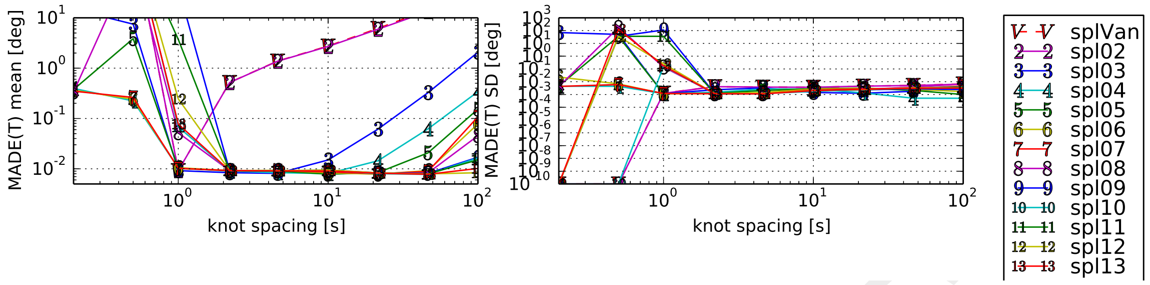
(a) with gyroscope



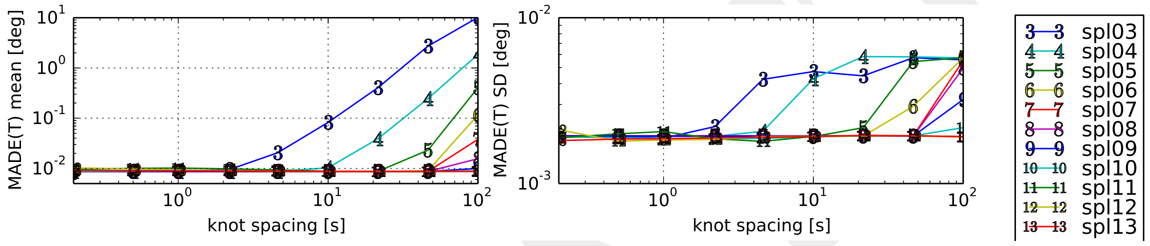
(b) with system model

Fig. 13 Estimation time over knot spacing on a single Intel(R) Xeon(R) E5 core at 3.60GHz with 2 * 12800 MB/s peak memory transfer rate. $T = 0.2$**h, yielding** 720 **readings per sensor. Sensors: sun sensor, magnetometer.**

When aiming at estimation accuracy, it clearly would be more important to compare the time over all iterations of the nonlinear solving till convergence of the cost because the different spline orders affect the convergence speed in terms of required iterations. Therefore we plot this total time in Figure 13 on page 35.



(a) with gyroscope



(b) with system model

Fig. 14 MADE(T) over the time between knots on a logarithmic scale. The sensor rate is a $1\text{s} = 10^0\text{s}$. $T = 0.2$h. Sensors: sun sensor, magnetometer.

To demonstrate how well the system model can prevent the splines from overfitting we run another very similar experiment by starting the knot spacing below the sensor period (1s), and exponentially increasing the number of knots. The result is depicted in Figure 14 on page 36. Obviously the gyroscope-based estimation becomes inaccurate and unstable when the knot rate reaches the sensor rate (10^0s) while the system model based estimation keeps maximal accuracy until the expressiveness of the spline is not able to represent the trajectory.

VI. Conclusion and Future work

In this paper we extended the quaternion B-spline formulation of [37] to any order and all Lie groups and derived all necessary Jacobian formulas needed for batch continuous-time state estimation on Lie groups both in the general case, and in the specific case for attitude estimation using unit-length quaternion splines, a singularity-free continuous-time attitude representation. We evaluated the performance of these curves for estimating spacecraft attitude against the state-of-the-art approach recently published by [63]. We showed that B-splines have equal or superior performance over all test cases and provide two key tuning parameters—the number of knots and the spline order—that an engineer can use to trade off accuracy and computational efficiency when choosing a spline representation for a given estimation problem.

We believe that we have just scratched the surface of what this method is capable of. The rest of this section outlines promising directions for future work.

A. Lie group B-splines

Most of the empirical results of the experiment section IV should be possible to derive theoretically. To do that could also answer the design questions (which estimator should I use for my problem) much more reliably.

Another valuable future work could be to evaluate the performance of different integration schemes and different policies to choose where to evaluate the integrand with respect to the B-spline's knots.

Our ad hoc method of initialization proved to be very useful, but, for mission-critical applications, it will be necessary to have analytical guarantees on its reliability. We strongly believe that, for many important Lie groups, this method is reliable within appropriate bounds.

When integrating velocity measurements for initialization it is important to recognize instances where the rotation vector corresponding to a time step leaves the injectivity area of the exponential map. One could simply chop the offending vector into smaller chunks. Another way could be to append the B-spline control vertices with vectors that replace the $\log(\mathbf{q}_{i-1}^{-1}\mathbf{q}_i)$ expression. This would render the parameters more redundant but would allow segments longer than the $\log$ supports and increase the performance of spline evaluation by saving the computational cost of the logarithm. The tradeoff would be that updating the B-splines parameters would become much more involved. Still, this approach could be an interesting extension, especially to sparsely model high velocity but rather low acceleration movements.

Finally, we believe that, for spline-based estimation to be successful, we need reliable methods to choose the number of knots required based on the data as it is received. There is some preliminary work in this area [2, 54], but we believe there is much more effort needed on this topic.

B. Attitude estimator evaluation

To reduce the burden on the experimental comparison of attitude estimation a very valuable contribution would be to provide a diversity of benchmark problems in form of publicly available data. That would greatly improve the comparability of experimental results. We spent a lot of effort trying to replicate the experiments of [63], but we were unable to reproduce the results exactly (see V A). We can only guess that the difference is in the simulated data. Having a benchmarking suite with data and standard performance measures would make these cross-paper comparisons much easier.

1. Proof of Theorem 1

Theorem 1. *$\mathcal{A}^L$ is a sub $\mathbb{R}$-algebra of $\mathbb{R}^{\dim \mathcal{A} \times \dim \mathcal{A}}$, $\mathcal{G}^L$ is a matrix Lie group embedded in $\mathcal{A}^L$, and this pair is fully isomorphic to the pair of $\mathcal{G}$ and $\mathcal{A}$.*

Proof. It is enough to show that $\cdot^L$ is an algebra-monomorphism into $\mathbb{R}^{\dim \mathcal{A} \times \dim \mathcal{A}}$ because its image, $\mathcal{A}^L$, is then a sub algebra and the mapping an algebra-isomorphism onto it. The existence of this isomorphism yields already that $\mathcal{A}$ is isomorphic to $\mathcal{A}^L$, that $\mathcal{G}^L$ is a multiplicative subgroup in it and its restriction to $\mathcal{G}$, $\cdot^L|_{\mathcal{G}}$, must be a group isomorphism. As any vector space isomorphism is also smooth with respect to the canonical differential structure of a finite dimensional vector space we also have isomorphic Lie groups as we assumed the differential structure on $\mathcal{G}$ to be identical with the one induced from $\mathcal{A}$.

Let $\mathbf{a}, \mathbf{b} \in \mathcal{A}$ be arbitrary vectors, $\mathbf{1} \in \mathcal{A}$ denote the multiplicative identity element and Ψ again the map $\mathcal{A} \to \mathbb{R}^{\dim \mathcal{A}}$ the coordinate map with respect to some basis for $\mathcal{A}$. We will now proof the injectivity and homomorphy of $\cdot^L$, which together implies it to be an monomorphism.

1. The *injectivity* basically follows from $\mathcal{A}$ having a one and Ψ being injective:

$$\mathbf{a}^L = \mathbf{b}^L \Rightarrow \mathbf{a}^L \Psi(\mathbf{1}) = \mathbf{b}^L \Psi(\mathbf{1}) \underset{\text{def}}{\Rightarrow} \Psi(\mathbf{a1}) = \Psi(\mathbf{b1}) \Rightarrow \mathbf{a1} = \mathbf{b1} \Rightarrow \mathbf{a} = \mathbf{b}$$

2. *Algebra-homomorphy*:

   Let $\alpha \in \mathbb{R}$ be an arbitrary real number and $\mathbf{v} \in \mathcal{A}$ be a further arbitrary vector.

   From the fact that $\mathcal{A}$'s vector multiplication, the matrix multiplication and Ψ are (bi)linear follows then

$$(\mathbf{a}+\mathbf{b})^L \Psi(\mathbf{v}) \underset{\text{def}}{=} \Psi((\mathbf{a}+\mathbf{b})\mathbf{v}) = \Psi(\mathbf{av}+\mathbf{bv}) \underset{\text{def}}{=} \mathbf{a}^L\Psi(\mathbf{v}) + \mathbf{b}^L\Psi(\mathbf{v}) = (\mathbf{a}^L + \mathbf{b}^L)\Psi(\mathbf{v}) \qquad (65)$$

and

$$(\alpha\mathbf{a})^L \Psi(\mathbf{v}) \underset{\text{def}}{=} \Psi((\alpha\mathbf{a})\mathbf{v}) = \Psi(\mathbf{a}(\alpha\mathbf{v})) \underset{\text{def}}{=} \mathbf{a}^L \Psi(\alpha\mathbf{v}) = \mathbf{a}^L(\alpha\Psi(\mathbf{v})) = (\alpha\mathbf{a}^L)\Psi(\mathbf{v}). \tag{66}$$

From the associativity of both the $\mathcal{A}$-vector multiplication and the matrix product it follows that

$$(\mathbf{ab})^L \Psi(\mathbf{v}) \underset{\text{def}}{=} \Psi((\mathbf{ab})\mathbf{v}) = \Psi(\mathbf{a}(\mathbf{bv})) \underset{\text{def}}{=} \mathbf{a}^L(\mathbf{b}^L\Psi(\mathbf{v})) = (\mathbf{a}^L\mathbf{b}^L)\Psi(\mathbf{v}). \tag{67}$$

As $\mathbf{v}$ and with it $\Psi(\mathbf{v})$ was arbitrary it follows from the last three equations by the the fact that two square matrices are identical if they always yield the same when multiplied with arbitrary vectors that

$$(\mathbf{a}+\mathbf{b})^L = \mathbf{a}^L + \mathbf{b}^L,\ (\alpha\mathbf{a})^L = \alpha\mathbf{a}^L \text{ and } (\mathbf{ab})^L = \mathbf{a}^L\mathbf{b}^L.$$

As $\mathbf{a}, \mathbf{b}$ were arbitrary it follows the homomorphy with respect to the three algebra operations, the addition the scalar and vectorial multiplications.

□

2. Proof of Theorem 2

Theorem 2. *In our setting* $\exp_{\mathcal{G}}$ *is a restriction of* $\exp_{\mathcal{A}}$ *to* $\mathfrak{g} \subset \mathcal{A}$*, employing the natural identification of* $\mathcal{G}$*'s tangent spaces with sub vector spaces of* $\mathcal{A}$*.*

Proof. It is a well known fact for matrix Lie groups that Theorem 2 holds for the pair $(\mathcal{G}^L, \mathbb{R}^{\dim\mathcal{A}\times\dim\mathcal{A}})$. The exponential map on $\mathcal{A}^L$ is obviously a restriction of the matrix exponential on $\mathbb{R}^{\dim\mathcal{A}\times\dim\mathcal{A}}$ to $\mathcal{A}^L$. Restricting $\exp_{\mathbb{R}^{\dim\mathcal{A}\times\dim\mathcal{A}}}$ first to $\mathcal{A}^L$ and then to $\mathcal{G}^L$ yields the same as restricting directly to $\mathcal{G}^L$, and therefore the statement follows for the pair $(\mathcal{G}^L, \mathcal{A}^L)$. Finally, due to Theorem 1, the statement also follows for the isomorphic pair $(\mathcal{G}, \mathcal{A})$ because both notions of exponential maps are intrinsically defined. □

APPENDIX A: ACKNOWLEDGEMENT

This project has received funding from the European UnionâĂŹs Seventh Framework Programme for research, technological development and demonstration under grant agreement no 610603, EUROPA2.

[1] Anderson, S. and Barfoot, T. D. (2013). Towards relative continuous-time slam. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1033–1040, Karlsruhe, Germany.

[2] Anderson, S., Dellaert, F., and Barfoot, T. D. (2014). A hierarchical wavelet decomposition for continuous-time slam. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 373–380, Hong Kong, China.

[3] Bar-Itzhack, I. Y. and Oshman, Y. (1985). Attitude determination from vector observations: Quaternion estimation. *IEEE Transactions on Aerospace and Electronic Systems*, AES-21:128–136.

[4] Bar-Itzhack, I. Y. and Reiner, J. (1984). Recursive attitude determination from vector observations: Direction cosine matrix identification. *Journal of Guidance, Control, and Dynamics*, 7(1):51–56.

[5] Barfoot, T. D., Forbes, J. R., and Furgale, P. T. (2011). Pose estimation using linearized rotations and quaternion algebra. *Acta Astronautica*, 68(1-2):101–112.

[6] Bartels, R. H., Beatty, J. C., and Barsky, B. A. (1987). *An Introduction to Splines for use in Computer Graphics and Geometric Modeling*. Morgan Kaufmann Publishers Inc., Los Altos, California, USA.

[7] Bauchau, O. and Trainelli, L. (2003). The vectorial parameterization of rotation. *Nonlinear dynamics*, 32(1):71–92.

[8] Bosse, M., Zlot, R., and Flick, P. (2012). Zebedee: Design of a spring-mounted 3-d range sensor with application to mobile mapping. *Robotics, IEEE Transactions on*, 28(5):1104–1119.

[9] Chee, S. A. and Forbes, J. R. (2014). Norm-constrained consider kalman filtering. *Journal of Guidance, Control, and Dynamics*. To appear.

[10] Choukroun, D., Bar-Itzhack, I. Y., and Oshman, Y. (2006a). A novel quaternion kalman filter. *IEEE Transactions on Aerospace and Electronic Systems*, 42(1):174–190.

[11] Choukroun, D., Weiss, H., Bar-Itzhack, I. Y., and Oshman, Y. (2006b). Kalman filtering for matrix estimation. *IEEE Transactions on Aerospace and Electronic Systems*, 42(1):147–159.

[12] Choukroun, D., Weiss, H., Bar-Itzhack, I. Y., and Oshman, Y. (2010). Direction cosine matrix estimation from vector observations using a matrix kalman filter. *IEEE Transactions on Aerospace and Electronic Systems*, 46(1):61–79.

[13] Crassidis, J. L. and Markley, F. L. (2003). Unscented filtering for spacecraft attitude estimation. *Journal of Guidance, Control, and Dynamics*, 26(4):536–542.

[14] Crassidis, J. L., Markley, F. L., and Cheng, Y. (2007). A survey of nonlinear attitude estimation methods. *Journal of Guidance, Control, and Dynamics*, 30(1):12–28.

[15] de Boor, C. (2001). *A practical guide to splines*. Springer Verlag, New York, USA.

[16] de Ruiter, A. H. J. and Forbes, J. R. (2014). On the solution of wahbaï£¡s problem on $SO(n)$. *Journal of the Astronautical Sciences*. To appear.

[17] Dong, H. J. and Barfoot, T. D. (2012). Lighting-invariant visual odometry using lidar intensity imagery and pose interpolation. In *Proceedings of the International Conference on Field and Service Robotics (FSR)*, Matsushima, Japan.

[18] Farrel, J. L., Stuelpnagel, J. C., Wessner, R. H., Velman, J. R., and Brook, J. E. (1966). A least-squares estimate of satellite attitude. *SIAM Review*, 8(3):384–386.

[19] Farrell, J. L. (1970). Attitude determination by kalman filtering. *Automatica*, 6:419–430.

[20] Firoozi, D. and Namvar, M. (2012). Analysis of gyro noise in non-linear attitude estimation using a single vector measurement. *IET Control Theory and Applications*, 6(14):2226–2234.

[21] Forbes, J. R. and de Ruiter, A. H. J. (2014). LMI-based solution to wahba's problem. *Journal of Guidance, Control, and Dynamics*. To appear.

[22] Forbes, J. R., de Ruiter, A. H. J., and Zlotnik, D. E. (2014). Continuous-time norm-constrained kalman filtering. *Automatica*. To appear.

[23] Furgale, P., Rehder, J., and Siegwart, R. (2013). Unified temporal and spatial calibration for multi-sensor systems. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1280–1286, Tokyo, Japan.

[24] Furgale, P. T., Barfoot, T. D., and Sibley, G. (2012). Continuous-time batch estimation using temporal basis functions. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2088–2095, St. Paul, MN.

[25] Grip, H. F., Fossen, T., Johansen, T. A., and Saberi, A. (2012). Attitude estimation using biased gyro and vector measurements with time-varying reference vectors. *IEEE Transactions on Automatic Control*, 57(5):1332–1338.

[26] Hall, B. (2003). *Lie groups, Lie algebras, and representations: an elementary introduction*, volume 222. Springer.

[27] Hamel, T. and Mahony, R. (2006). Attitude estimation on $SO(3)$ based on direct inertial measurements. *IEEE Conference on Robotics and Automation, Orlando, FL, May*, pages 2170–2175.

[28] Hedborg, J., Forssen, P.-E., Felsberg, M., and Ringaby, E. (2012). Rolling shutter bundle adjustment. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1434 –1441.

[29] Howard, T. and Kelly, A. (2007). Optimal rough terrain trajectory generation for wheeled mobile robots. *International Journal of Robotics Research*, 26(2):141–166.

[30] Hughes, P. C. (1986). *Spacecraft Attitude Dynamics*. John Wiley & Sons, New York.

[31] IEEE Aerospace and Electronic Systems Society. Gyro and Accelerometer Panel and Institute of Electrical and Electronics Engineers (1998). *IEEE Standard Specification Format Guide and Test Procedure for Single-axis Interferometric Fiber Optic Gyros*. IEEE (std.). IEEE.

[32] Jazwinski, A. H. (1970). *Stochastic Processes and Filtering Theory*. Academic Press, Inc, New York, NY, USA.

[33] Jensen, K. J. (2011). Generalized nonlinear complementary attitude filter. *Journal of Guidance, Control, and Dynamics*, 34(5):1588–1592.

[34] Kaess, M., Johannsson, H., Roberts, R., Ila, V., Leonard, J., and Dellaert, F. (2012). iSAM2: Incremental smoothing

and mapping using the Bayes tree. *Intl. J. of Robotics Research, IJRR*, 31(2):217–236.

[35] Kelly, J. and Sukhatme, G. S. (2011). Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration. *The International Journal of Robotics Research*, 30(1):56–79.

[36] Khosravian, A. and Namvar, M. (2012). Rigid body attitude control using a single vector measurement and gyro. *IEEE Transactions on Automatic Control*, 57(5):1273–1279.

[37] Kim, M.-J., Kim, M.-S., and Shin, S. Y. (1995). A general construction scheme for unit quaternion curves with simple high order derivatives. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '95, pages 369–376, New York, NY, USA. ACM.

[38] Kinsey, J. C. and Whitcomb, L. L. (2007). Adaptive identification on the group of rigid-body rotations and its application to underwater vehicle navigation. *IEEE Transactions on Robotics*, 23(1):124–136.

[39] Kirillov, A. A., Kirillov, A. A., and Kirillov, A. A. (2008). *An introduction to Lie groups and Lie algebras*. Number 113. Cambridge University Press Cambridge.

[40] Lefferts, E. J., Markley, F. L., and Shuster, M. D. (1982). Kalman filtering for spacecraft attitude estimation. *Journal of Guidance, Control, and Dynamics*, 5(5):417–429.

[41] Leutenegger, S., Furgale, P., Rabaud, V., Chli, M., Konolige, K., and Siegwart, R. (2013). Keyframe-based visual-inertial slam using nonlinear optimization. In *Proceedings of Robotics: Science and Systems*, Berlin, Germany.

[42] Li, M. and Mourikis, A. I. (2013). High-precision, consistent EKF-based visual-inertial odometry. *International Journal of Robotics Research*, 32(6):690–711.

[43] Lovegrove, S., Patron-Perez, A., and Sibley, G. (2013). Spline fusion: A continuous-time representation for visual-inertial fusion with application to rolling shutter cameras. In *Proceedings of the British Machine Vision Conference*. BMVA Press.

[44] Mahony, R., Hamel, T., and Pflimlin, J.-M. (2005). Complementary filter design on the special orthogonal group $SO(3)$. 44^{th} *IEEE Conference on Decision and Control and the European Control Conference, Seville, Spain, December 12-15*, pages 1477–1484.

[45] Mahony, R., Hamel, T., and Pflimlin, J.-M. (2008). Nonlinear complementary filters on the special orthogonal group. *IEEE Transactions on Automatic Control*, 53(5):1203–1218.

[46] Markley, F. L. (1988). Attitude determination using vector observations and the singular value decomposition. *Journal of the Astronautical Sciences*, 36(3):245–258.

[47] Markley, F. L. (2003). Attitude error representations for kalman filtering. *Journal of Guidance, Control, and Dynamics*, 26(2):311–317.

[48] Markley, F. L. (2004). Multiplicative vs. additive filtering for spacecraft attitude determination. In *Proceedings of the 6th Conference on Dynamics and Control of Systems and Structures in Space (DCSSS)*, volume D22, Riomag-

giore, Italy.

[49] Mirzaei, F. and Roumeliotis, S. (2008). A Kalman filter-based algorithm for IMU-camera calibration: Observability analysis and performance evaluation. *Robotics, IEEE Transactions on*, 24(5):1143–1156.

[50] Moore, J. B. and Tam, P. K. (1973). Fixed-lag smoothing for nonlinear systems with discrete measurements. *Information Sciences*, 6:151–160.

[51] Mortari, D. (1997). Esoq: A closed-form solution to the wahba problem. *Journal of the Astronautical Sciences*, 45(2):195–204.

[52] Mortari, D. (2000). Second estimator of the optimal quaternion. *Journal of Guidance, Control and Dynamics*, 23(5):885–888.

[53] Mourikis, A. I., Trawny, N., Roumeliotis, S. I., Johnson, A. E., Ansar, A., and Matthies, L. (2009). Vision-aided inertial navigation for spacecraft entry, descent, and landing. *IEEE Transactions on Robotics*, 25(2):264–280.

[54] Oth, L., Furgale, P., Kneip, L., and Siegwart, R. (2013). Rolling shutter camera calibration. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 1360–1367.

[55] Park, F. C. and Ravani, B. (1997). Smooth invariant interpolation of rotations. *ACM Trans. Graph.*, 16:277–295.

[56] Poli, D. and Toutin, T. (2012). Review of developments in geometric modelling for high resolution satellite pushbroom sensors. *The Photogrammetric Record*, 27(137):58–73.

[57] Ringaby, E. and Forssén, P.-E. (2011). Efficient video rectification and stabilisation for cell-phones. *International Journal of Computer Vision*, pages 1–18. 10.1007/s11263-011-0465-8.

[58] Shuster, M. D. (1989). A simple kalman filter and smoother for spacecraft attitude. *Journal of the Astronautical Sciences*, 37(1):89–106.

[59] Shuster, M. D. and Oh, S. D. (1981). Three-axis attitude determination from vector observations. *Journal of Guidance, Control, and Dynamics*, 4(1):70–77.

[60] Strasdat, H., Montiel, J., and Davison, A. J. (2012). Visual slam: Why filter? *Image and Vision Computing*, 30(2):65–77.

[61] Tong, C. H. and Barfoot, T. D. (2013). Gaussian process gauss-newton for 3d laser-based visual odometry. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 5204–5211, Karlsruhe, Germany.

[62] Tong, C. H., Furgale, P. T., and Barfoot, T. D. (2013). Gaussian process Gauss-Newton for non-parametric simultaneous localization and mapping. *International Journal of Robotics Research*, 32(5):507–525.

[63] Vandersteen, J., Diehl, M., Aerts, C., and Swevers, J. (2013). Spacecraft attitude estimation and sensor calibration using moving horizon estimation. *Journal of Guidance, Control, and Dynamics*, 36(3):734–742.

[64] Wahba, G. (1965). Problem 65-1, a least-squares estimate of satellite attitude. *SIAM Review*, 7(3):409.

[65] Wertz, J. R. (1978). *Spacecraft Attitude Determination and Control*. Kluwer Academic Publishers, Dordrecht, The Netherlands.

[66] Zanetti, R., Majji, M., Bishop, R., and Mortari, D. (2009). Norm-constrained kalman filtering. *Journal of Guidance, Control, and Dynamics*, 32(5):1458–1465.

[67] The noise characteristics of a particular gyroscope may be identified using the method described in IEEE Aerospace and Electronic Systems Society. Gyro and Accelerometer Panel and Institute of Electrical and Electronics Engineers [31]

Online Self-Calibration for Robotic Systems

Journal:	*International Journal of Robotics Research*
Manuscript ID:	IJR-14-2147
Manuscript Type:	Original Article
Date Submitted by the Author:	26-Aug-2014
Complete List of Authors:	Maye, Jérôme; ETH Zurich, MAVT Furgale, Paul; ETH Zurich, MAVT Siegwart, Roland; ETH Zurich, MAVT
Keyword:	Learning and Adaptive Systems < Cognitive Robotics, Sensor Fusion < Sensing and Perception Computer Vision, Mapping < Mobile and Distributed Robotics SLAM, Localization < Mobile and Distributed Robotics SLAM, Calibration and Identification < Mechanics, Design and Control
Abstract:	We present a generic algorithm for self calibration of robotic systems that utilizes two key innovations. First, it uses an information-theoretic measure to automatically identify and store novel measurement sequences. This keeps the computation tractable by discarding redundant information and allows the system to build a sparse but complete calibration dataset from data collected at different times. Second, as the full observability of the calibration parameters may not be guaranteed for an arbitrary measurement sequence, the algorithm detects and locks unobservable directions in parameter space using a combination of rank-revealing QR and singular value decompositions of the Fisher information matrix. The result is an algorithm that listens to an incoming sensor stream, builds a minimal set of data for estimating the calibration parameters, and updates parameters as they become observable, leaving the others locked at their initial guess. We validate our approach through an extensive set of simulated and real-world experiments.

Online Self-Calibration for Robotic Systems

Jérôme Maye, Roland Siegwart, and Paul Furgale
Autonomous Systems Lab, ETH Zurich, Switzerland
email: {`jerome.maye, roland.siegwart, paul.furgale`}`@mavt.ethz.ch`

Abstract

We present a generic algorithm for self calibration of robotic systems that utilizes two key innovations. First, it uses an information-theoretic measure to automatically identify and store novel measurement sequences. This keeps the computation tractable by discarding redundant information and allows the system to build a sparse but complete calibration dataset from data collected at different times. Second, as the full observability of the calibration parameters may not be guaranteed for an arbitrary measurement sequence, the algorithm detects and locks unobservable directions in parameter space using a combination of rank-revealing QR and singular value decompositions of the Fisher information matrix. The result is an algorithm that listens to an incoming sensor stream, builds a minimal set of data for estimating the calibration parameters, and updates parameters as they become observable, leaving the others locked at their initial guess. We validate our approach through an extensive set of simulated and real-world experiments.

1 Introduction

Every robotic system has some set of parameters—scale factors, sensor locations, link lengths, etc.—that are needed for state estimation, planning, and control. Despite best efforts during construction, some parameters will change over the lifetime of a robot due to normal wear and tear. In the best case, incorrect parameter values degrade performance. In the worst case, they cause critical safety issues. We are interested in developing automated systems that are capable of robust and long-term deployment in the hands of non-experts, so the automatic identification and update of these parameter values is highly important.

As an example, consider a camera-based collision avoidance system intended for deployment in a consumer automobile. For the vehicle to be able to avoid collisions, the pose of each camera with respect to the vehicle coordinate system must be known precisely so that obstacle positions can be transformed from camera coordinates into vehicle coordinates. However, a consumer vehicle will have no access to special calibration hardware or expert data analysis. In such a scenario, the vehicle must be capable of self recalibration. This problem is inherently difficult for a number of reasons that we will briefly discuss here.

1) Parameters change over time—although the vehicle may be factory calibrated, the parameters can change slowly over time due to vibration, thermal expansion, loose parts, or any number of other common problems that follow from normal usage.

2) Parameters must be inferred from the data—as the cameras may be installed in different places on the vehicle, their pose cannot be measured directly. Instead, the parameters must be inferred from the data produced by the full system. However, this is only possible if the motion of the vehicle renders the parameters *observable*[1].

3) Normal operation may result in unobservable directions in parameter space—unfortunately, for this and many other practical problems normal operation may not render all directions in parameters space observable. In this example, when two cameras do not share an overlapping field of view, planar motion renders the calibration problem degenerate[2]; the transformation between cameras only becomes observable under general 3D motion.

4) Unobservable directions in parameter space may appear observable in the presence of noise—even if our hypothetical car is piloted only on a plane (a degenerate case), noise in the measurements can make unobservable parameters appear observable. We call these parameters *numerically unobservable* to mirror the related concept of numerically rank-deficient matrices.

Existing algorithms for self calibration generally handle issues (1) and (2). Issue (3) is dealt with by designing experiments that guarantee all parameters become observable. Issue (4) has been largely ignored. Therefore, unless we plan to require all vehicle owners to outfit their parking places with calibration patterns or regularly drive off-road to make the parameters observable, new advances for online system calibration are required. To the best of our knowledge, our previous conference paper (that this paper extends) [34] is the first published self-calibration algorithm able to cope with issue (4).

In this paper, we propose an algorithm to deal with *all* of the aforementioned difficulties. Our approach exploits the algebraic links between the Gauss-Newton algorithm, the Fisher information matrix, and nonlinear observability analysis to automatically detect directions in parameter space that are numerically unobservable and avoid updating our parameters in these directions; at any given time, directions in the parameter space that are observable will be updated based on the latest information, while unobservable directions will remain at their initial guess. Novel sets of measurements are detected using an information gain test and added to a working set that is used to estimate the parameters. The result is an algorithm that listens to an incoming stream of data, automatically accumulating a batch of data that may be used to calibrate a full robot system. The only requirements are that (i) the parameters are theoretically observable given some ideal set of data, (ii) it is possible to implement

[1]Informally, observability means that the parameters can be inferred unambiguously from some local batch of measurement data [22].

[2]See [28] for a handy table of degenerate cases or [25] for a more theoretical analysis.

a batch Gauss-Newton estimator for the system state and calibration parameters based on a set of measurements, and (iii) we have some reasonable initial guess for the calibration parameters (*e.g.* from factory calibration or manual measurements).

This paper follows on from our previous conference paper [34]. The method and the experiments have both been extended significantly. The remainder of the paper is structured as follows. Sec. 2 will give a brief overview over related approaches. Sec. 3 is dedicated to the mathematical groundings of our method. Sec. 4 demonstrates the validity of our approach through extensive experiments and evaluation. Sec. 5 will conclude the paper.

2 Related Work

The problem of sensor calibration has been a recurring one in the history of robotics and computer vision. Thereby, it has been addressed using a variety of sensor setups and algorithms. A calibration process may involve recovering *intrinsic*, *e.g.*, focal length for a camera, and *extrinsic* parameters, *i.e.*, the rigid transformation between the sensor's coordinate system and a reference coordinate system. For the former, one could devise a naive approach where the parameters are accurately determined during the manufacturing process. Similarly, the transformation could be retrieved by means of some measuring instrument. However, the disadvantages of such purely engineered methods are manifold. Apart from their impracticality, it can be nearly impossible to reach a satisfying level of accuracy and thus hinder the proper use of the sensor in a robotic system. Furthermore, external factors such as temperature variations or mechanical shocks may seriously bias a factory calibration. Therefore, much efforts have been dedicated over the years to develop algorithms for calibration of systems in the field.

At a first level, sensor calibration methods can be classified into online or offline algorithms. While the latter are more popular in the computer vision community, the former are widespread in robotic applications. Online methods usually augment the state vector of a Bayes filter with calibration parameters and offline approaches rely on the minimization of a cost function. At a second level, calibration algorithms can be distinguished in how they address observability or singularity issues. While some publications present an informal or formal offline analysis, some others silently assume that the dataset contains enough information to calibrate the sensor parameters. Finally, the complexity of the setup and the amount of supervision are substantial criteria to assess the practicality of a calibration method.

Due to its simplicity, the use of a planar calibration pattern such as a checkerboard, coupled with nonlinear regression, has become the most popular method in computer vision during the last decade for calibrating intrinsic camera parameters [47, 42]. Given a set of point correspondences between the model planes and their images, the algorithm attempts to minimize a nonlinear cost function consisting in the sum of squared reprojection errors. While [47] remains rather vague about degenerate configurations, a compendium of singularities is proposed in [42]. The planar calibration pattern is also used in [46] to estimate the relative pose between a camera and a laser rangefinder, with no particular attention to singularities in the camera views. While being relatively efficient and inexpensive, this procedure still requires expert knowledge to reach a good level of accuracy. Even though point correspondences can be automatically identified by corner detectors [19, 39], it is still up to the user to execute proper motion of the calibration pattern in front of the camera.

A key limitation of the plane-based calibration technique for intrinsic camera parameters is that it remains essentially offline. Whenever a change of calibration occurs, *e.g.*, an adjustment of the focal length, the vision task has to be interrupted for recalibration. In order to allow for online operation, Maybank and Faugeras have pioneered in [33, 9] a method for self-calibration that does not require any pattern with known 3D geometry. Given a couple of images from the same scene under different viewpoints and point correspondences, epipolar transformations are computed and a system of equations is solved to yield the camera intrinsic parameters. Due to the lack of probabilistic model, the method is inherently sensitive to noise. Furthermore, a proper calibration cannot be guaranteed under general camera motion.

In the context of mobile robotics, several authors have included the calibration problem in a state estimation framework, either with filtering [32, 36, 12, 24] or nonlinear least-squares [5, 26, 20] techniques. Filtering techniques based on the Kalman filter are appealing due to their inherently online nature. However, in case of nonlinear systems, least-squares techniques based on iterative optimization are usually superior in terms of accuracy. In [32], Martinelli *et al.* perform a nonlinear observability analysis using Lie derivatives and overcome the singularities by choosing an alternative parameterization for the robot's state. An identical method for observability analysis has also been employed in [36, 24].

More recently, Brookshire and Teller [2, 3] have carried out a formal observability analysis in order to identify degenerate paths of the calibration dataset. In contrast to the aforementioned methods, they inspect the rank of the Fisher information matrix to determine the observability of the calibration parameters. However, their approach still expects that the robot travels along non-degenerate paths during the calibration run. A standard nonlinear least-squares method is subsequently used for the estimation.

Another class of methods seeks to maximize the concordance of the sensor data with the environment [45, 29] or a quality measure [40, 31, 41]. While Underwood *et al.* [45] use a vertical pole with retro reflective tape, Levinson and Thrun [29] propose an unsupervised approach relying on the assumption that neighboring points from laser beams lie on a contiguous surface. The method presented in [40, 31, 41] estimates relative poses of laser rangefinders by minimizing the entropy of an underlying generative model for the location of laser points in space, hence maximizing the sharpness of the distribution and the quality of the resulting point cloud.

Closely related to our approach, Menq *et al.* [35] have also adopted matrix factorization to identify observable and unobservable subspaces in the context of calibration of industrial robots. However, the authors do not explicitly deal with the notion of numerical rank deficiency, which, in our opinion, is essential to the proper operation of a calibration algorithm.

To the best of our knowledge, little research has been devoted to account for degenerate cases frequently occurring during calibration runs. Some authors, notably in the robotics community, have proposed an offline observability analysis to either state that the system is observable or to identify potential singularities. As will be demonstrated in this paper, traditional observability analysis is helpless when it comes to noisy input data on real physical systems. It is indeed purely algebraic, performed a priori, and hence omits the numerical difficulties associated with sensory data. In general, apart from [35], the authors assume that their estimation framework is fed with well-behaved data. In contrast, our approach relaxes this assumption and is therefore suitable in the hands of non-expert users for online and long-term operations on various platforms and sensors.

3 Method

In this section, we shall first state a probabilistic model for the calibration problem and present an in-depth description of our inference method. In a second step, we shall propose an online extension to our approach. Practical algorithmic considerations will conclude the presentation.

3.1 Probabilistic Model

The first step of our calibration method is to identify the different variables in the system and establish a probabilistic model through which they interact. These variables shall be classified in the following three categories:

1. The data variable denoted by $\mathbf{X} = (X_1, \ldots, X_D)^T$ and samples thereof as $\mathbf{X}_i$. A realization of the observable random variable, $\mathbf{X}_i$, corresponds to the measurement of a sensor, $\mathcal{S}$.

2. The calibration variable denoted by $\mathbf{\Theta} = (\Theta_1, \ldots, \Theta_K)^T$. This latent random variable is the primary object of interest and comprises all the intrinsic and extrinsic calibration parameters of the sensor $\mathcal{S}$.

3. The nuisance variable represented by $\mathbf{\Psi} = (\Psi_1, \ldots, \Psi_L)^T$. This latent random variable is not of direct interest, but the data distribution depends on it (*e.g.*, poses of landmarks).

For the sake of clarity, we restrict our formulation to a single sensor $\mathcal{S}$. Nevertheless, our approach smoothly generalizes to multiple sensors. We denote random variates of $\mathbf{X}$ by $\mathbf{x}$, of $\mathbf{\Theta}$ by $\boldsymbol{\theta}$, and of $\mathbf{\Psi}$ by $\boldsymbol{\psi}$.

Assuming a realization $\mathbf{x}_1, \dots, \mathbf{x}_N$ of an *i.i.d.* data sample, henceforth denoted by $\mathbf{x}_{1:N}$, the posterior joint probability density function over the latent variables is

$$f_{\boldsymbol{\Psi},\boldsymbol{\Theta}}\left(\boldsymbol{\psi},\boldsymbol{\theta} \mid \mathbf{x}_{1:N}\right) = \frac{\prod_{i=1}^{N} f_{\mathbf{X}}\left(\mathbf{x}_i \mid \boldsymbol{\psi},\boldsymbol{\theta}\right) f_{\boldsymbol{\Psi},\boldsymbol{\Theta}}\left(\boldsymbol{\psi},\boldsymbol{\theta}\right)}{\prod_{i=1}^{N} f_{\mathbf{X}}\left(\mathbf{x}_i\right)}. \tag{1}$$

As our final goal is to perform inference on the calibration variable, $\boldsymbol{\Theta}$, we shall marginalize over the nuisance variable, $\boldsymbol{\Psi}$, to get the marginal posterior density function

$$f_{\boldsymbol{\Theta}}\left(\boldsymbol{\theta} \mid \mathbf{x}_{1:N}\right) = \int f_{\boldsymbol{\Psi},\boldsymbol{\Theta}}\left(\boldsymbol{\psi},\boldsymbol{\theta} \mid \mathbf{x}_{1:N}\right) \, \mathrm{d}\boldsymbol{\psi}. \tag{2}$$

From the large sample theory, the posterior density in Equ. 1 can be approximated through Laplace's method [13] by the normal distribution,

$$f_{\boldsymbol{\Psi},\boldsymbol{\Theta}}\left(\boldsymbol{\psi},\boldsymbol{\theta} \mid \mathbf{x}_{1:N}\right) \approx \mathcal{N}\left(\widehat{\boldsymbol{\psi\theta}}, \mathcal{I}^{-1}\left(\widehat{\boldsymbol{\psi\theta}}\right)\right), \tag{3}$$

where $\widehat{\boldsymbol{\psi\theta}}$ is the joint posterior mode and $\mathcal{I}(\widehat{\boldsymbol{\psi\theta}})$ the observed Fisher information matrix, *i.e.*, the negative of the Hessian matrix of the log-likelihood function evaluated at the joint posterior mode.

Using the properties of the normal distribution [1], we can derive a closed-form expression for the posterior marginal density in Equ. 2 as

$$f_{\boldsymbol{\Theta}}\left(\boldsymbol{\theta} \mid \mathbf{x}_{1:N}\right) \approx \mathcal{N}\left(\boldsymbol{\mu}_{\boldsymbol{\Theta}}, \boldsymbol{\Sigma}_{\boldsymbol{\Theta}}\right), \tag{4}$$

where $\boldsymbol{\mu}_{\boldsymbol{\Theta}}$ and $\boldsymbol{\Sigma}_{\boldsymbol{\Theta}}$ represent partitions of the joint mean and covariance matrix in Equ. 3 respectively.

To conclude this probabilistic model, we shall state the assumed form of the data distribution $f_{\mathbf{X}}$ in Equ. 1 as

$$\mathbf{X} \sim \mathcal{N}\left(g_{\mathcal{S}}\left(\boldsymbol{\psi},\boldsymbol{\theta}\right), \boldsymbol{\Sigma}_{\mathbf{X}}\right), \tag{5}$$

where $g_{\mathcal{S}}(\cdot)$ is an arbitrary twice-differentiable function, henceforth assumed to be nonlinear, and $\boldsymbol{\Sigma}_{\mathbf{X}}$ the deterministic covariance matrix of the measurements. The choice of a normally distributed measurement model is standard in robotics. Borrowing from the inverse problem theory [43], we refer to the function $g_{\mathcal{S}}(\cdot)$ as the forward model. It expresses the mathematical model of the physical system under study and predicts the outcome of $\mathbf{X}$ for given model parameters $\boldsymbol{\psi}$ and $\boldsymbol{\theta}$.

3.2 Nonlinear Least-Squares Problem

Under the assumptions of Sec. 3.1, the joint posterior mode in Equ. 3 coincides with the solution to the least-squares problem

$$\widehat{\boldsymbol{\psi\theta}} = \underset{\boldsymbol{\psi},\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{i=1}^{N} \mathbf{e}_i^T \boldsymbol{\Sigma}_{\mathbf{X}}^{-1} \mathbf{e}_i, \tag{6}$$

where $\mathbf{e}_i$ is an error term defined as

$$\mathbf{e}_i = \mathbf{x}_i - g_{\mathcal{S}}(\boldsymbol{\psi}, \boldsymbol{\theta}). \tag{7}$$

We adopt an objective Bayesian viewpoint by setting an uninformative prior density $f_{\boldsymbol{\Psi},\boldsymbol{\Theta}}(\boldsymbol{\psi}, \boldsymbol{\theta}) \propto 1$.

For the general case of a nonlinear function, $g_{\mathcal{S}}(\cdot)$, Equ. 6 has no closed-form solution and we have to resort to numerical optimization techniques [23]. In this paper, we adopt the Gauss-Newton algorithm that iteratively refines an initial guess, $\boldsymbol{\psi\theta}_{(0)}$, towards a local minimum,

$$\boldsymbol{\psi\theta}_{(t+1)} \leftarrow \boldsymbol{\psi\theta}_{(t)} + \Delta\boldsymbol{\psi\theta}_{(t)}. \tag{8}$$

The shift vector, $\Delta\boldsymbol{\psi\theta}_{(t)}$, is the solution to a linear least-squares problem constructed from a first-order linear approximation at the current estimate, $\boldsymbol{\psi\theta}_{(t)}$. The least-squares estimate for the shift vector is thus the solution to the normal equations,

$$\mathbf{J}_{(t)}^T \boldsymbol{\Sigma}^{-1} \mathbf{J}_{(t)} \Delta\boldsymbol{\psi\theta}_{(t)} = -\mathbf{J}_{(t)}^T \boldsymbol{\Sigma}^{-1} \Delta\mathbf{e}_{(t)}, \tag{9}$$

where $\mathbf{J}_{(t)}$ is the Jacobian matrix of the stacked error terms evaluated at the current estimate, $\boldsymbol{\Sigma}$ the covariance matrix of the whole data sample, and $\Delta\mathbf{e}_{(t)}$ the stacked vector of residuals at the current estimate.

Using the Cholesky decomposition of the positive-semidefinite inverse covariance matrix $\boldsymbol{\Sigma}^{-1}$, we can rewrite Equ. 9 as

$$\left(\boldsymbol{\Sigma}^{-\frac{T}{2}} \mathbf{J}_{(t)}\right)^T \left(\boldsymbol{\Sigma}^{-\frac{T}{2}} \mathbf{J}_{(t)}\right) \Delta\boldsymbol{\psi\theta}_{(t)} = -\left(\boldsymbol{\Sigma}^{-\frac{T}{2}} \mathbf{J}_{(t)}\right)^T \left(\boldsymbol{\Sigma}^{-\frac{T}{2}} \Delta\mathbf{e}_{(t)}\right), \tag{10}$$

which, from linear algebra theory [15], can be recognized as the normal equations of the overdetermined system of linear equations,

$$\left(\boldsymbol{\Sigma}^{-\frac{T}{2}} \mathbf{J}_{(t)}\right) \Delta\boldsymbol{\psi\theta}_{(t)} = -\boldsymbol{\Sigma}^{-\frac{T}{2}} \Delta\mathbf{e}_{(t)}. \tag{11}$$

At convergence of the Gauss-Newton algorithm after I iterations, the parameter estimate becomes

$$\widehat{\boldsymbol{\psi\theta}} = \boldsymbol{\psi\theta}_{(I)}, \tag{12}$$

and the observed Fisher information, constituting the inverse covariance matrix in Equ. 3, is given by

$$\mathcal{I}(\widehat{\boldsymbol{\psi\theta}}) = \mathbf{J}_{(I)}^T \boldsymbol{\Sigma}^{-1} \mathbf{J}_{(I)}. \tag{13}$$

Matrix factorization algorithms such as the singular value decomposition (SVD) or rank-revealing QR (RRQR) decomposition [6] provide a unique least-squares solution to Equ. 11 whenever $\boldsymbol{\Sigma}^{-T/2}\mathbf{J}_{(t)}$ has full rank[3]. In case of

[3]Informally, the rank of a matrix corresponds to the maximum number of linearly independent column or row vectors.

rank deficiency, there are an infinite number of least-squares minimizers, the SVD providing the unique minimum-norm solution and RRQR decomposition a basic solution.

As the matrix rank is an algebraic concept, a matrix built from noisy sensor data or resulting from floating-point operations will rarely be truly rank deficient. Our algorithm relies on automated analysis of the rank of the Gauss-Newton system, even in the presence of noisy measurements. Fortunately, the concept of numerical rank [15] provides an elegant workaround to these difficulties. The numerical rank corresponds to the algebraic rank of the unperturbed matrix within a neighborhood defined by a parameter ϵ proportional to the magnitude of the perturbation matrix. When the noise on the matrix entries has the same scale (*e.g.*, by column or row scaling), the numerical rank can be determined by the singular values [18].

3.3 Parameter Inference

This section explains our method of performing iterative Gauss-Newton-style estimation while automatically detecting and locking unobservable directions in the space of the calibration parameters, $\mathbf{\Theta}$. While $\mathbf{\Theta}$ is mostly low dimensional, $\mathbf{\Psi}$ can become high dimensional, especially in a simultaneous localization and mapping (SLAM) scenario, where it encodes parameters for the robot states and the landmarks in the scene. Hence, the matrices involved in the Gauss-Newton system of equations are large, sparse, and may be rank deficient.

There are well-known methods to deal with the problem of rank deficiency in Gauss-Newton. The most widespread is the Levenberg-Marquardt algorithm that adds a scaled identity matrix to the left-hand-side of the Gauss-Newton equations. This mutes the singular values of the Fischer information matrix and makes it possible to solve the linear system. However, this method is not suitable in our case; we are interested in performing automated rank analysis of the reduced system so that we can determine the directions in parameter space that are numerically unobservable. Levenberg-Marquardt changes the linear system and makes this analysis impossible.

The SVD would be the optimal tool for accurate numerical rank detection [7] and matrix analysis. Unfortunately, this method is intractable on large sparse matrices, where we must use an RRQR decomposition. Therefore, we propose a method, inspired by the Schur complement, that exploits the benefits of each approach. The RRQR algorithm (suitable for large, sparse matrices) is used to marginalize out $\mathbf{\Psi}$. The remaining low-dimensional system of equations for $\mathbf{\Theta}$ can then be solved (and analyzed) using the SVD. Finally, back substitution is used to recover the shift vector corresponding to $\mathbf{\Psi}$. The method is unique in its partitioning of the problem and handling of rank deficient matrices.

In the Jacobian matrix of Equ. 10, each column corresponds to one dimension of the parameter space. For our application, we shall thus partition it in the following manner

$$\mathbf{J} = \begin{pmatrix} \mathbf{J}_{\boldsymbol{\psi}} & \mathbf{J}_{\boldsymbol{\theta}} \end{pmatrix}, \tag{14}$$

where $\mathbf{J}_{\boldsymbol{\psi}} \in \mathbb{R}^{M\times L}$ corresponds to the Jacobian matrix with respect to $\boldsymbol{\psi}$, $\mathbf{J}_{\boldsymbol{\theta}} \in \mathbb{R}^{M\times K}$ with respect to $\boldsymbol{\theta}$, and $M = N \times D$. For the sake of clarity, we hence assume that the data covariance matrix, $\boldsymbol{\Sigma}$, has been absorbed in the Jacobian matrix and the right-hand side of the normal equations using a Cholesky decomposition. Furthermore, we drop the minus sign and the iteration indexes. Given these assumptions, the normal equations in Equ. 10 can be rewritten as

$$\begin{pmatrix} \mathbf{J}_{\boldsymbol{\psi}}^T\mathbf{J}_{\boldsymbol{\psi}} & \mathbf{J}_{\boldsymbol{\psi}}^T\mathbf{J}_{\boldsymbol{\theta}} \\ (\mathbf{J}_{\boldsymbol{\psi}}^T\mathbf{J}_{\boldsymbol{\theta}})^T & \mathbf{J}_{\boldsymbol{\theta}}^T\mathbf{J}_{\boldsymbol{\theta}} \end{pmatrix} \begin{pmatrix} \Delta\boldsymbol{\psi} \\ \Delta\boldsymbol{\theta} \end{pmatrix} = \begin{pmatrix} \mathbf{J}_{\boldsymbol{\psi}}^T\Delta\mathbf{e} \\ \mathbf{J}_{\boldsymbol{\theta}}^T\Delta\mathbf{e} \end{pmatrix}. \quad (15)$$

3.3.1 Calibration Variable Estimation

In order to isolate the subsystem of equations related to $\Delta\boldsymbol{\theta}$ only, we need to cancel out the lower-left block of the Fisher information matrix by left-multiplying it with the full-rank matrix

$$\mathbf{C} = \begin{pmatrix} \mathbf{I}_{\boldsymbol{\psi}} & \mathbf{0} \\ -(\mathbf{J}_{\boldsymbol{\psi}}^T\mathbf{J}_{\boldsymbol{\theta}})^T(\mathbf{J}_{\boldsymbol{\psi}}^T\mathbf{J}_{\boldsymbol{\psi}})^{-1} & \mathbf{I}_{\boldsymbol{\theta}} \end{pmatrix}, \quad (16)$$

where $\mathbf{I}_{\boldsymbol{\psi}} \in \mathbb{R}^{L\times L}$ and $\mathbf{I}_{\boldsymbol{\theta}} \in \mathbb{R}^{K\times K}$ are identity matrices. Left-multiplying the left-hand side of Equ. 15 by $\mathbf{C}$ yields

$$\begin{pmatrix} \mathbf{J}_{\boldsymbol{\psi}}^T\mathbf{J}_{\boldsymbol{\psi}} & \mathbf{J}_{\boldsymbol{\psi}}^T\mathbf{J}_{\boldsymbol{\theta}} \\ \mathbf{0} & \mathbf{J}_{\boldsymbol{\theta}}^T\mathbf{J}_{\boldsymbol{\theta}} - (\mathbf{J}_{\boldsymbol{\psi}}^T\mathbf{J}_{\boldsymbol{\theta}})^T(\mathbf{J}_{\boldsymbol{\psi}}^T\mathbf{J}_{\boldsymbol{\psi}})^{-1}\mathbf{J}_{\boldsymbol{\psi}}^T\mathbf{J}_{\boldsymbol{\theta}} \end{pmatrix}, \quad (17)$$

and the right-hand side gives

$$\begin{pmatrix} \mathbf{J}_{\boldsymbol{\psi}}^T\Delta\mathbf{e} \\ \mathbf{J}_{\boldsymbol{\theta}}^T\Delta\mathbf{e} - (\mathbf{J}_{\boldsymbol{\psi}}^T\mathbf{J}_{\boldsymbol{\theta}})^T(\mathbf{J}_{\boldsymbol{\psi}}^T\mathbf{J}_{\boldsymbol{\psi}})^{-1}\mathbf{J}_{\boldsymbol{\psi}}^T\Delta\mathbf{e} \end{pmatrix}. \quad (18)$$

It follows from Equ. 17 that we can now solve the equations independently for $\Delta\boldsymbol{\theta}$. The solution of the reduced system will be the same as the solution to the original system of equations.

Up to this point, this is a standard application of the Schur complement. However, when $\mathbf{J}_{\boldsymbol{\psi}}^T\mathbf{J}_{\boldsymbol{\psi}}$ is large, sparse, and potentially rank deficient, we need to take care when computing the matrix expression $(\mathbf{J}_{\boldsymbol{\psi}}^T\mathbf{J}_{\boldsymbol{\theta}})^T(\mathbf{J}_{\boldsymbol{\psi}}^T\mathbf{J}_{\boldsymbol{\psi}})^{-1}\mathbf{J}_{\boldsymbol{\psi}}^T$ in the lower parts of Equ. 17 and Equ. 18. To this end, we first apply an RRQR decomposition to $\mathbf{J}_{\boldsymbol{\psi}}$, which leads to

$$\mathbf{J}_{\boldsymbol{\psi}} = \mathbf{Q}\mathbf{R}\boldsymbol{\Pi}^T, \quad (19)$$

where $\boldsymbol{\Pi} \in \mathbb{R}^{L\times L}$ is a permutation matrix whose goal is to move the most linearly independent columns of $\mathbf{J}_{\boldsymbol{\psi}}$ to the front, $\mathbf{R} \in \mathbb{R}^{M\times L}$, and $\mathbf{Q} \in \mathbb{R}^{M\times M}$ an orthogonal matrix. We use Equ. 19 to simplify the matrix expression as follows

$$\begin{aligned}
&(\mathbf{J}_{\boldsymbol{\psi}}^T\mathbf{J}_{\boldsymbol{\theta}})^T(\mathbf{J}_{\boldsymbol{\psi}}^T\mathbf{J}_{\boldsymbol{\psi}})^{-1}\mathbf{J}_{\boldsymbol{\psi}}^T \\
&\quad = \mathbf{J}_{\boldsymbol{\theta}}^T\mathbf{Q}\mathbf{R}\boldsymbol{\Pi}^T(\boldsymbol{\Pi}\mathbf{R}^T\mathbf{Q}^T\mathbf{Q}\mathbf{R}\boldsymbol{\Pi}^T)^{-1}\boldsymbol{\Pi}\mathbf{R}^T\mathbf{Q}^T \\
&\quad = \mathbf{J}_{\boldsymbol{\theta}}^T\mathbf{Q}\mathbf{R}\boldsymbol{\Pi}^T(\boldsymbol{\Pi}\mathbf{R}^T\mathbf{R}\boldsymbol{\Pi}^T)^{-1}\boldsymbol{\Pi}\mathbf{R}^T\mathbf{Q}^T \\
&\quad = \mathbf{J}_{\boldsymbol{\theta}}^T\mathbf{Q}_1\mathbf{R}_1\boldsymbol{\Pi}^T(\boldsymbol{\Pi}\mathbf{R}_1^T\mathbf{R}_1\boldsymbol{\Pi}^T)^{-1}\boldsymbol{\Pi}\mathbf{R}_1^T\mathbf{Q}_1^T \\
&\quad = \mathbf{J}_{\boldsymbol{\theta}}^T\mathbf{Q}_1\mathbf{R}_1\boldsymbol{\Pi}^T\boldsymbol{\Pi}\mathbf{R}_1^{-1}\mathbf{R}_1^{-T}\boldsymbol{\Pi}^T\boldsymbol{\Pi}\mathbf{R}_1^T\mathbf{Q}_1^T \\
&\quad = \mathbf{J}_{\boldsymbol{\theta}}^T\mathbf{Q}_1\mathbf{R}_1\mathbf{R}_1^{-1}\mathbf{R}_1^{-T}\mathbf{R}_1^T\mathbf{Q}_1^T \\
&\quad = \mathbf{J}_{\boldsymbol{\theta}}^T\mathbf{Q}_1\mathbf{Q}_1^T, \qquad (20)
\end{aligned}$$

where $\mathbf{R}_1 \in \mathbb{R}^{L\times L}$ is the upper-triangular part of $\mathbf{R}$ and $\mathbf{Q}_1 \in \mathbb{R}^{M\times L}$ contains the first L Householder vectors of $\mathbf{Q}$. In this derivation, we have first used the orthogonality of $\mathbf{Q}$ to eliminate it from the matrix inversion. In a second step, we have converted the decomposition into a thin QR decomposition. Lastly, the orthogonality of $\boldsymbol{\Pi}$ has allowed us to arrive at the final result. Consequently, we have transformed an involved expression containing a large matrix inversion into a much affordable matrix multiplication. Furthermore, we are not affected by the numerical rank of $\mathbf{J}_{\boldsymbol{\psi}}$ since we do not compute any matrix inverse, and we only need a reduced QR decomposition.

Using Equ. 20, we introduce the following auxiliary variables,

$$\mathbf{A}_{\boldsymbol{\theta}} = \mathbf{J}_{\boldsymbol{\theta}}^T\mathbf{J}_{\boldsymbol{\theta}} - (\mathbf{J}_{\boldsymbol{\theta}}^T\mathbf{Q}_1)(\mathbf{J}_{\boldsymbol{\theta}}^T\mathbf{Q}_1)^T, \qquad (21)$$

$$\mathbf{b}_{\boldsymbol{\theta}} = \mathbf{J}_{\boldsymbol{\theta}}^T\Delta\mathbf{e} - (\mathbf{J}_{\boldsymbol{\theta}}^T\mathbf{Q}_1)(\Delta\mathbf{e}^T\mathbf{Q}_1)^T, \qquad (22)$$

such that the subsystem of equations for $\Delta\boldsymbol{\theta}$ becomes

$$\mathbf{A}_{\boldsymbol{\theta}}\Delta\boldsymbol{\theta} = \mathbf{b}_{\boldsymbol{\theta}}, \qquad (23)$$

where $\mathbf{A}_{\boldsymbol{\theta}} \in \mathbb{R}^{K\times K}$ and $\mathbf{b}_{\boldsymbol{\theta}} \in \mathbb{R}^{K}$. When using the Householder form of $\mathbf{Q}_1$, $\mathbf{A}_{\boldsymbol{\theta}}$ and $\mathbf{b}_{\boldsymbol{\theta}}$ can be computed at low cost. Moreover, even though $\mathbf{A}_{\boldsymbol{\theta}}$ is dense, the small size of K keeps the memory footprint low. Prior to solving these equations, we factorize $\mathbf{A}_{\boldsymbol{\theta}}$ using SVD such that

$$\mathbf{A}_{\boldsymbol{\theta}} = \mathbf{U}\mathbf{S}\mathbf{V}^T, \qquad (24)$$

where $\mathbf{U} = (\mathbf{u}_1, \ldots, \mathbf{u}_K)$ and $\mathbf{V} = (\mathbf{v}_1, \ldots, \mathbf{v}_K)$ are orthogonal matrices, and $\mathbf{S} = \operatorname{diag}[\varsigma_1, \ldots, \varsigma_K]$ is a diagonal matrix of singular values.

We compute the numerical rank $r_{\epsilon_{\boldsymbol{\theta}}}$ of $\mathbf{A}_{\boldsymbol{\theta}}$ from its singular values using a threshold $\epsilon_{\boldsymbol{\theta}}$. This threshold is proportional to the magnitude of the expected noise in $\mathbf{A}_{\boldsymbol{\theta}}$ [17] and we shall further investigate its determination later in this paper. From the SVD and the numerical rank, we can readily solve Equ. 23 using a truncated SVD (TSVD) [16] as

$$\Delta\boldsymbol{\theta} = \sum_{i=1}^{r_{\epsilon_{\boldsymbol{\theta}}}} \frac{\mathbf{u}_i^T\mathbf{b}_{\boldsymbol{\theta}}}{\varsigma_i}\mathbf{v}_i. \qquad (25)$$

The computation of the full covariance matrix involves the inverse of the Fisher information matrix. However, for a 2×2 block matrix as in the left-hand side of Equ. 15, we can advantageously perform blockwise inversions. Due to the marginalization property of the normal distribution, we only require the lower-right block given by the pseudoinverse $\mathbf{A}_{\boldsymbol{\theta}}^{\dagger}$ [4], which yields the marginal covariance matrix

$$\boldsymbol{\Sigma}_{\boldsymbol{\Theta}} = \sum_{i=1}^{r_{\epsilon_{\boldsymbol{\theta}}}} \frac{\mathbf{v}_i \mathbf{u}_i^T}{\varsigma_i}. \tag{26}$$

Finally, we have a direct access to the numerical nullspace of $\mathbf{A}_{\boldsymbol{\theta}}$ as

$$\text{null}\left[\mathbf{A}_{\boldsymbol{\theta}}, \epsilon_{\boldsymbol{\theta}}\right] = \text{span}\left[\{\mathbf{v}_{r_{\epsilon_{\boldsymbol{\theta}}}+1}, \ldots, \mathbf{v}_K\}\right]. \tag{27}$$

In [21], Jauffret clearly articulates the link between the singularity of the Fisher information matrix and observability of the parameters being estimated. In our context, the basis vectors from the numerical nullspace correspond to directions in the parameter space which are numerically unobservable. While the first $r_{\epsilon_{\boldsymbol{\theta}}}$ column vectors of $\mathbf{V}$ form an orthonormal basis for the row space of $\mathbf{A}_{\boldsymbol{\theta}}$ or the observable subspace, the basis vectors from the nullspace form an orthonormal basis for the unobservable subspace.

Stacking the basis vectors of the nullspace into a matrix $\mathbf{N}_{\mathbf{A}_{\boldsymbol{\theta}}} \in \mathbb{R}^{K \times (K - r_{\epsilon_{\boldsymbol{\theta}}})}$, we reveal the local observability structure of the parameter. Each row i of the nullspace matrix represents the observability of the dimension i of the parameter vector. A row of zeros implies that the corresponding dimension is fully observable. A row containing a single one signifies that the corresponding dimension is unobservable. Any other rows correspond to dimensions in the parameter space that are only observable as a linear combination of other dimensions. We can interpret the TSVD solution in Equ. 25 as the unique least-squares solution in the observable subspace. In other words, the shift vector will only update the observable part of the parameter vector.

3.3.2 Nuisance Variable Estimation

Plugging $\Delta\boldsymbol{\theta}$ into Equ. 15, $\Delta\boldsymbol{\psi}$ becomes the solution to the following system of equations

$$(\mathbf{J}_{\boldsymbol{\psi}}^T \mathbf{J}_{\boldsymbol{\psi}})\Delta\boldsymbol{\psi} = \mathbf{J}_{\boldsymbol{\psi}}^T(\Delta\mathbf{e} - \mathbf{J}_{\boldsymbol{\theta}}\Delta\boldsymbol{\theta}), \tag{28}$$

which can be recognized as the normal equations of the least-squares problem

$$\min_{\Delta\boldsymbol{\psi} \in \mathbb{R}^L} \left\|\mathbf{J}_{\boldsymbol{\psi}}\Delta\boldsymbol{\psi} - (\Delta\mathbf{e} - \mathbf{J}_{\boldsymbol{\theta}}\Delta\boldsymbol{\theta})\right\|_2. \tag{29}$$

Given the potential large size of $\mathbf{J}_{\boldsymbol{\psi}}$, we solve Equ. 29 with an RRQR decomposition. In most applications, this is indeed the only possibility. Furthermore, this decomposition is already available from Equ. 19 and we directly compute a basic solution as

$$\Delta\boldsymbol{\psi} = \boldsymbol{\Pi}\begin{pmatrix}\mathbf{R}_{11}^{-1}\mathbf{Q}_{11}^{T}(\Delta\mathbf{e}-\mathbf{J}_{\boldsymbol{\theta}}\Delta\boldsymbol{\theta})\\ \mathbf{0}\end{pmatrix}, \tag{30}$$

where $\mathbf{Q}_{11} \in \mathbb{R}^{M\times r_{\epsilon_{\boldsymbol{\psi}}}}$ contains the first $r_{\epsilon_{\boldsymbol{\psi}}}$ Householder vectors of $\mathbf{Q}$, $\mathbf{R}_{11} \in \mathbb{R}^{r_{\epsilon_{\boldsymbol{\psi}}}\times r_{\epsilon_{\boldsymbol{\psi}}}}$ is the upper-left triangular part of $\mathbf{R}$, and $r_{\epsilon_{\boldsymbol{\psi}}}$ the estimated numerical rank of $\mathbf{J}_{\boldsymbol{\psi}}$ with tolerance $\epsilon_{\boldsymbol{\psi}}$. Instead of inverting $\mathbf{R}_{11}$, we solve the system by back substitution. The basic solution has at most $r_{\epsilon_{\boldsymbol{\psi}}}$ nonzero components. If $\mathbf{J}_{\boldsymbol{\psi}}$ has full rank, the lower null vector disappears and Equ. 30 yields the unique least-squares solution to Equ. 29. In this paper, we have chosen the basic solution for its computational simplicity. The least-squares solution would have required a complete orthogonal decomposition. As our primary interest lies in estimating $\boldsymbol{\theta}$, we claim that the basic solution is suitable for $\boldsymbol{\psi}$.

3.3.3 Improper Posterior Inference

Using the shift vectors computed in the previous sections, we can iterate to convergence without taking steps in the unobservable directions in parameter space. After convergence, we would like to recover the posterior marginal density. If the calibration parameter is partially observable or unobservable, the posterior marginal density in Equ. 4 is improper. The covariance matrix, $\boldsymbol{\Sigma}_{\boldsymbol{\Theta}}$, is singular and the posterior is a degenerate normal distribution. However, we can still define proper density functions in the observable and unobservable subspaces. To this end, we shall first apply the following affine transformation to $\boldsymbol{\Theta}$

$$\widetilde{\boldsymbol{\Theta}} = \mathbf{V}^{T}\boldsymbol{\Theta}, \tag{31}$$

where $\mathbf{V}$ is the orthogonal matrix from the SVD in Equ. 24. Due to the affine transformation property of the normal distribution, the random vector $\widetilde{\boldsymbol{\Theta}}$ is normally distributed as

$$\widetilde{\boldsymbol{\Theta}} \mid \mathbf{x}_{1:N} \sim \mathcal{N}\left(\mathbf{V}^{T}\boldsymbol{\mu}_{\boldsymbol{\Theta}}, \mathbf{V}^{T}\boldsymbol{\Sigma}_{\boldsymbol{\Theta}}\mathbf{V}\right). \tag{32}$$

We can further partition $\widetilde{\boldsymbol{\Theta}}$ into an observable and unobservable part $\widetilde{\boldsymbol{\Theta}} = (\widetilde{\boldsymbol{\Theta}}_{obs}, \widetilde{\boldsymbol{\Theta}}_{nobs})^{T}$, where

$$\begin{aligned}\widetilde{\boldsymbol{\Theta}}_{obs} &= (\widetilde{\Theta}_{1}, \dots, \widetilde{\Theta}_{r_{\epsilon_{\boldsymbol{\theta}}}})^{T},\\ \widetilde{\boldsymbol{\Theta}}_{nobs} &= (\widetilde{\Theta}_{r_{\epsilon_{\boldsymbol{\theta}}}+1}, \dots, \widetilde{\Theta}_{K})^{T}.\end{aligned} \tag{33}$$

We adopt the same partition for the mean vector of $\widetilde{\boldsymbol{\Theta}}$, *i.e.*,

$$\boldsymbol{\mu}_{\widetilde{\boldsymbol{\Theta}}} = \mathbf{V}^{T}\boldsymbol{\mu}_{\boldsymbol{\Theta}} = (\boldsymbol{\mu}_{\widetilde{\boldsymbol{\Theta}}_{obs}}, \boldsymbol{\mu}_{\widetilde{\boldsymbol{\Theta}}_{nobs}})^{T}. \tag{34}$$

Finally, we can use the orthogonality of $\mathbf{V}$ to simplify the covariance matrix of $\widetilde{\mathbf{\Theta}}$ into

$$\mathbf{\Sigma}_{\widetilde{\mathbf{\Theta}}} = \mathbf{V}^T \mathbf{\Sigma}_{\mathbf{\Theta}} \mathbf{V} = \mathbf{S}^{\dagger}, \tag{35}$$

where $\mathbf{S}$ is the diagonal matrix of the singular values of $\mathbf{A}_{\boldsymbol{\theta}}$ defined in Equ. 24 and its pseudoinverse is the $K \times K$ matrix

$$\mathbf{S}^{\dagger} = \operatorname{diag}\left[\varsigma_1^{-1}, \ldots, \varsigma_{r_{\epsilon_{\boldsymbol{\theta}}}}^{-1}\right]. \tag{36}$$

Using the marginalization property of the normal distribution, the observable part is normally distributed as

$$\widetilde{\mathbf{\Theta}}_{obs} \mid \mathbf{x}_{1:N} \sim \mathcal{N}\left(\boldsymbol{\mu}_{\widetilde{\mathbf{\Theta}}_{obs}}, \mathbf{S}^{\dagger}_{r_{\epsilon_{\boldsymbol{\theta}}}}\right), \tag{37}$$

where $\mathbf{S}^{\dagger}_{r_{\epsilon_{\boldsymbol{\theta}}}} \in \mathbb{R}^{r_{\epsilon_{\boldsymbol{\theta}}} \times r_{\epsilon_{\boldsymbol{\theta}}}}$ is the upper-left nonzero block of $\mathbf{S}^{\dagger}$. Lastly, the unobservable part is distributed as

$$\widetilde{\mathbf{\Theta}}_{nobs} \mid \mathbf{x}_{1:N} \sim \mathcal{N}\left(\boldsymbol{\mu}_{\widetilde{\mathbf{\Theta}}_{nobs}}, \mathbf{0}\right), \tag{38}$$

which can be interpreted as a Dirac distribution [37] with density given by

$$\begin{aligned} f_{\widetilde{\mathbf{\Theta}}_{nobs}}\left(\widetilde{\boldsymbol{\theta}}_{nobs} \mid \boldsymbol{\mu}_{\widetilde{\mathbf{\Theta}}_{nobs}}\right) &= \delta(\widetilde{\boldsymbol{\theta}}_{nobs} - \boldsymbol{\mu}_{\widetilde{\mathbf{\Theta}}_{nobs}}), \\ \delta(\widetilde{\boldsymbol{\theta}}_{nobs} - \boldsymbol{\mu}_{\widetilde{\mathbf{\Theta}}_{nobs}}) &= \begin{cases} +\infty & \text{for } \widetilde{\boldsymbol{\theta}}_{nobs} = \boldsymbol{\mu}_{\widetilde{\mathbf{\Theta}}_{nobs}} \\ 0 & \text{for } \widetilde{\boldsymbol{\theta}}_{nobs} \neq \boldsymbol{\mu}_{\widetilde{\mathbf{\Theta}}_{nobs}}. \end{cases} \end{aligned} \tag{39}$$

3.4 Online Inference

The method presented thus far is inherently offline, *i.e.*, it requires an entire calibration dataset for inferring parameter values. For nonlinear least-squares problems, full batch methods are more accurate and stable as they iteratively refine the linearization point as compared to recursive estimation techniques such as the extended Kalman filter (EKF). In this paper, we want to benefit from the robustness of offline methods, along with the possibility to deploy our algorithm in an online and long-term setting. To reach this goal, we process small batches of data sequentially and decide to keep them based on their utility for the calibration. Each new batch is merged to the old ones to refine our knowledge about the calibration parameters until we reach a satisfactory level of confidence.

To start with the mathematical formulation of our online method, we define the current set of stored data samples as $\mathcal{D}^{info} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, which has led to the posterior marginal density in Equ. 37, associated with the random variable $\widetilde{\mathbf{\Theta}}_{obs} \mid \mathcal{D}^{info}$. The set, $\mathcal{D}^{info}$, contains the current informative measurements for the calibration variable. Our sensor, $\mathcal{S}$, continuously streams new data that we accumulate in another batch of size ΔN denoted by

$\mathcal{D}^{new} = \{\mathbf{x}_{N+1}, \ldots, \mathbf{x}_{N+\Delta N}\}$. Intuitively, if the measurements in $\mathcal{D}^{new}$ are similar to those in $\mathcal{D}^{info}$, we are not really improving our knowledge about $\widetilde{\boldsymbol{\Theta}}_{obs}$ and we can safely discard $\mathcal{D}^{new}$ to keep the computation tractable.

Information theory provides a principled way to evaluate the usefulness of $\mathcal{D}^{new}$. Shortly after its introduction in the signal processing community, information theory has been adopted by statisticians, notably by Bayesian proponents such as Lindley. In [30], he defines the amount of information in a random variable as the negative Shannon entropy. A random variable with high entropy, *e.g.*, uniformly distributed, contains little information. Conversely, a sharply-peaked distribution with low entropy conveys high information. The amount of information provided by an experiment may therefore be defined as the difference between the posterior and prior information. This can be interpreted as a measure of the utility or information gain of the experiment. In our setup, we can consider the prior to be $\widetilde{\boldsymbol{\Theta}}_{obs} \mid \mathcal{D}^{info}$, the posterior $\widetilde{\boldsymbol{\Theta}}_{obs} \mid \mathcal{D}^{info}, \mathcal{D}^{new}$, and define the utility to be

$$\mathcal{U}((\widetilde{\boldsymbol{\Theta}}_{obs} \mid \mathcal{D}^{info}), \mathcal{D}^{new}) = h(\widetilde{\boldsymbol{\Theta}}_{obs} \mid \mathcal{D}^{info}) - h(\widetilde{\boldsymbol{\Theta}}_{obs} \mid \mathcal{D}^{info}, \mathcal{D}^{new}), \tag{40}$$

where $h(\cdot)$ denotes the differential entropy. It follows from Equ. 40 that if $\mathcal{D}^{new}$ is useful for estimating $\widetilde{\boldsymbol{\Theta}}_{obs}$, it reduces its entropy and increases its information. In other words, the utility tells us whether it is worth adding $\mathcal{D}^{new}$ to our set of informative measurements. If the utility falls under a prescribed threshold $\delta_{\mathcal{U}}$, $\mathcal{D}^{new}$ can be discarded since it does not bring a significant information gain. Otherwise, it can be added to the informative set which becomes $\mathcal{D}^{info} \cup \mathcal{D}^{new}$.

We shall now examine the derivation of the utility function in our context where the random variables are normally distributed. The differential entropy of a normally distributed random variable $\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{X}}, \boldsymbol{\Sigma}_{\mathbf{X}})$ of dimension K is given by

$$h(\mathbf{X}) = \frac{K}{2}(1 + \log_e(2\pi)) + \frac{1}{2}\log_e |\boldsymbol{\Sigma}_{\mathbf{X}}|, \tag{41}$$

where $|\cdot|$ denotes the matrix determinant and $\log_e$ the natural logarithm. This latter is chosen for mathematical convenience, given the exponential form of the normal distribution. We can retrieve an entropy in bits by dividing Equ. 41 by $\log_e(2)$. If we now state the distribution of the prior as $\widetilde{\boldsymbol{\Theta}}_{obs} \mid \mathcal{D}^{info} \sim \mathcal{N}(\boldsymbol{\mu}^{prior}, \boldsymbol{\Sigma}^{prior})$ and of the posterior as $\widetilde{\boldsymbol{\Theta}}_{obs} \mid \mathcal{D}^{info}, \mathcal{D}^{new} \sim \mathcal{N}(\boldsymbol{\mu}^{post}, \boldsymbol{\Sigma}^{post})$, the information gain in Equ. 40 becomes

$$\mathcal{U}((\widetilde{\boldsymbol{\Theta}}_{obs} \mid \mathcal{D}^{info}), \mathcal{D}^{new}) = \frac{1}{2}\log_e \frac{|\boldsymbol{\Sigma}^{prior}|}{|\boldsymbol{\Sigma}^{post}|}. \tag{42}$$

As $\boldsymbol{\Sigma}^{post}$ and $\boldsymbol{\Sigma}^{prior}$ are diagonal matrices, their determinant is the product of the respective inverse singular values. Finally, when using the property of the logarithm, Equ. 42 becomes

$$\mathcal{U}((\widetilde{\boldsymbol{\Theta}}_{obs} \mid \mathcal{D}^{info}), \mathcal{D}^{new}) = \frac{1}{2}\left(\sum_{i=1}^{r_{\epsilon_{\boldsymbol{\theta}}}^{post}} \log_e \varsigma_i^{post} - \sum_{i=1}^{r_{\epsilon_{\boldsymbol{\theta}}}^{prior}} \log_e \varsigma_i^{prior}\right), \tag{43}$$

where $r^{post}_{\epsilon_{\boldsymbol{\theta}}}$ and $r^{prior}_{\epsilon_{\boldsymbol{\theta}}}$ are the numerical ranks of $\mathbf{A}^{post}_{\boldsymbol{\theta}}$ and $\mathbf{A}^{prior}_{\boldsymbol{\theta}}$ respectively. In summary, the information gain in Equ. 43 is available at low cost from the already computed SVD. We can simply maintain the sum of the logarithm of the singular values of the current estimate. When $r^{post}_{\epsilon_{\boldsymbol{\theta}}} > r^{prior}_{\epsilon_{\boldsymbol{\theta}}}$, the data has increased the dimensionality of the observable subspace. This corresponds to infinite information gain and the batch is always accepted.

3.5 Algorithmic Details

For a given sensor, $\mathcal{S}$, we require a reasonable initial guess of the calibration ($\boldsymbol{\theta}_{(0)}$) and nuisance ($\boldsymbol{\psi}_{(0)}$) parameters, the forward model, $g_{\mathcal{S}}(\cdot)$, describing the data generation process, the analytical or numerical Jacobian matrix, $\mathbf{J}$, of the error terms, and the data covariance matrix, $\boldsymbol{\Sigma}_{\mathbf{X}}$. This latter is either available from the datasheet of the sensor or should be estimated from a training dataset.

In addition to the model parameters, our method relies on algorithmic parameters that can be classified into the sets $\mathcal{P}_N = \{\epsilon_{\boldsymbol{\psi}}, \epsilon_{\boldsymbol{\theta}}\}$ and $\mathcal{P}_O = \{\delta_{\mathcal{U}}, \Delta N\}$. These parameters are specific to our algorithm and govern its numerical and online behavior respectively. In $\mathcal{P}_N$, the parameters $\epsilon_{\boldsymbol{\psi}}$ and $\epsilon_{\boldsymbol{\theta}}$ are the thresholds for the numerical rank detection of $\mathbf{J}_{\boldsymbol{\psi}}$ and $\mathbf{A}_{\boldsymbol{\theta}}$ defined above. Their value depends on the expected noise in the matrix entries, which is composed of roundoff errors and, in our case, essentially of discrepancies between the current estimate and the true value for the parameters. Except when we have a statistical model for the noise as in [17], the determination of these parameters remains empirical and application specific. In $\mathcal{P}_O$, the parameter ΔN controls the size of the batches evaluated and $\delta_{\mathcal{U}}$ is a threshold for the information provided by a batch with respect to an existing marginal posterior for $\widetilde{\boldsymbol{\Theta}}$. ΔN can represent a certain number of measurements or a time window during which measurements are gathered. A large value for ΔN corresponds in the limit to a full batch method and is therefore not recommended for online operations. A small value for ΔN involves many unnecessary optimization runs and might not provide enough information to render the calibration parameters observable. The selection of ΔN is thus purely empirical, dependent on the vehicle dynamics, and is a trade-off between online and offline behavior. The information threshold $\delta_{\mathcal{U}}$ is given in bits. If it is set to a low value, most of the batches will be accepted and our algorithm will become largely inefficient as compared to a full batch method. On the other hand, when setting $\delta_{\mathcal{U}}$ to a large value, we will discard most of the batches and fail to capture important information. As for ΔN, the choice of $\delta_{\mathcal{U}}$ is empirical and application specific. These two parameters are furthermore related in the sense that larger batches can potentially reduce the uncertainty of the posterior distribution in a more significant way.

As mentioned earlier, robotic applications involving landmark observations or robot's poses will most likely result in a sparse Jacobian matrix. This specific model structure can be exploited by storing it in memory using a sparse matrix data structure, for which there exists an efficient QR decomposition [8]. Consequently, our algorithm has a time complexity in $O(L^{3/2})$ [14] and space complexity in $O(L)$.

4 Experiments

We consider a differential-drive wheeled robot equipped with a laser rangefinder (LRF). While moving in a plane, the LRF measures distances to poles disseminated on the ground. Furthermore, the robot is endowed with odometry sensors, measuring its linear and angular velocity. This scenario is a prototypical SLAM problem. Freely adopting the SLAM jargon, we refer to the poles as landmarks. The positions in an inertial reference frame of a collection of landmarks form a map of the environment. Given an initial inertial reference frame, the SLAM problem consists in estimating the poses of the robot, *i.e.*, its trajectory, and the positions of the landmarks, solely based on its sensors. In order to solve this problem, we require the relative pose between the LRF frame and the robot frame, and the relative poses between the robot and the inertial frame. The LRF measurements shall indeed be transformed in the inertial frame to construct a consistent map. Recalling our presentation in Sec. 3, the relative pose between the LRF frame and the robot frame corresponds to the calibration variable, and the relative poses between the robot and the inertial frame, along with the positions of the landmarks, to the nuisance variable.

This scenario was first showcased in [34] with a preliminary version of our algorithm. In the rest of this section, we shall phrase the underlying mathematical model and explore the problem to its full extent. Due to its simplicity, we can thoroughly analyze the behavior of our algorithm with extensive simulations and real-world experiments.

4.1 Problem Formulation

Before starting, we shall first identify the different coordinate frames in the problem. The inertial reference frame or world frame is represented by $\mathcal{F}_W$ with origin w. As we only compute relative poses to $\mathcal{F}_W$, its absolute pose is arbitrary. The robot frame, $\mathcal{F}_R$, has its origin, r, located at the center of the axle between the left and right wheels. The first axis of $\mathcal{F}_R$ is aligned with the direction of forward motion of the robot and the second axis points to the left. The LRF frame, $\mathcal{F}_L$, has the origin, l, at the point of emission of the laser beams. The first axis of $\mathcal{F}_L$ points along the forward direction of the LRF and the second axis to the left.

The relative pose of $\mathcal{F}_R$ with respect to $\mathcal{F}_W$ is a time-varying transformation denoted by ${}^{w}\mathbf{T}_r(t)$, composed of a rotation, ${}^{w}\mathbf{R}_r(t)$, and a translation, ${}^{w}\mathbf{r}^{wr}(t)$. The relative pose of $\mathcal{F}_L$ with respect to $\mathcal{F}_R$ is a time-invariant transformation denoted by ${}^{r}\mathbf{T}_l$, composed of a rotation, ${}^{r}\mathbf{R}_l$, and a translation, ${}^{r}\mathbf{r}^{rl}$. The motion of the robot being constrained on a plane, we represent a translation, respectively a linear velocity, by two scalar parameters, and a rotation, respectively an angular velocity, by a single angular parameter.

4.1.1 Motion Model

For the purpose of this application, we choose to approximate the motion model in discrete time using a first-order Runge-Kutta method with a small step size, T, as

$$\begin{aligned} {}^{w}\mathbf{r}^{wr}_{k+1} &= {}^{w}\mathbf{r}^{wr}_{k} + T\,{}^{w}\mathbf{R}_{r_k}\,{}^{r}\mathbf{v}^{wr}_{k}, \\ \varphi_{k+1} &= \varphi_k + T\omega_k, \\ 0 &= \begin{pmatrix} 0 & 1 \end{pmatrix} {}^{r}\mathbf{v}^{wr}_{k}, \end{aligned} \tag{44}$$

where k is the discrete-time index, ${}^{r}\mathbf{v}^{wr}_{k}$ the linear velocity of $\mathcal{F}_R$ as seen from $\mathcal{F}_W$ and expressed in $\mathcal{F}_R$, ω_k the angular velocity of $\mathcal{F}_R$ as seen from $\mathcal{F}_W$, and

$${}^{w}\mathbf{R}_{r_k} = \begin{pmatrix} \cos\varphi_k & -\sin\varphi_k \\ \sin\varphi_k & \cos\varphi_k \end{pmatrix}. \tag{45}$$

The last equation in Equ. 44 acts as a non-holonomic constraint disallowing pure lateral motion of the robot.

The robot can measure its linear and angular velocity through rotary encoders attached to its wheels. We adopt here a simplified odometry model that provides at time k a measurement, $\mathbf{v}^{o}_{k} = (v^{o}_{k}, 0, \omega^{o}_{k})^{T}$, composed of a linear velocity, v^{o}_{k}, and an angular velocity, ω^{o}_{k}, in an odometry frame, $\mathcal{F}_O$, collocated with $\mathcal{F}_R$. By adding a pseudo zero measurement for satisfying the non-holonomic constraint, we can isolate ${}^{r}\mathbf{v}^{wr}_{k}$ and ω_k in Equ. 44 to relate it to the odometry measurement.

4.1.2 Observation Model

The environment contains a set of Nl landmarks represented by the points $\{\ell_1, \ldots, \ell_{Nl}\}$. We assume that the LRF returns for each landmark, ℓ_i, a range, r^{ℓ_i}, and a bearing angle, φ^{ℓ_i}. If a landmark is not in the field of view of the LRF, its corresponding range is set to zero and flagged as invalid. Despite being a rather simplified setup, it is perfectly sufficient for demonstrating our approach, while circumventing the problem of landmark detection and association between consecutive laser scans. The range r^{ℓ_i} corresponds to the distance between the origin, l, and the landmark, ℓ_i. The bearing angle, φ^{ℓ_i}, corresponds to the angle between the vector, ${}^{l}\mathbf{r}^{l\ell_i}$, and the first axis of $\mathcal{F}_L$. The ranges and bearing angles can also be interpreted as the polar coordinates of the landmarks with respect to a polar coordinate system $\mathring{\mathcal{F}}_L$, whose pole is located at the point l and polar axis along the first axis of $\mathcal{F}_L$. Therefore, the measurement at time k of a landmark ℓ_i can be represented by the column vector

$${}^{\mathring{l}}\boldsymbol{\ell}_{i_k} = \begin{pmatrix} r^{\ell_i}_{k} \\ \varphi^{\ell_i}_{k} \end{pmatrix} = {}^{\circ}f_{\perp}({}^{l}\mathbf{T}_{r}\,{}^{r}\mathbf{T}_{w_k}\,{}^{w}\boldsymbol{\ell}_i), \tag{46}$$

where the function ${}^{\circ}f_{\perp}$ converts from Cartesian to polar coordinates and ${}^{w}\boldsymbol{\ell}_i$ contains the Cartesian coordinates of ℓ_i with respect to $\mathcal{F}_W$.

4.1.3 Estimation Model

While Equ. 44 provides a forward model $g_o\left({}^w\mathbf{T}_{r_{k+1}},\, {}^w\mathbf{T}_{r_k}\right)$ for an odometry measurement, $\mathbf{v}_k^o$, Equ. 46 defines a forward model $g_{\ell_i}\left({}^w\mathbf{T}_{r_k},\, {}^w\boldsymbol{\ell}_i,\, {}^r\mathbf{T}_l\right)$ for an LRF measurement, ${}^{\mathring{l}}\boldsymbol{\ell}_{i_k}$. We denote the covariance matrix for $\mathbf{v}_k^o$ by $\boldsymbol{\Sigma}_o = \operatorname{diag}\left[\sigma_v^2, \sigma_c^2, \sigma_\omega^2\right]$ and the covariance matrix for ${}^{\mathring{l}}\boldsymbol{\ell}_{i_k}$ by $\boldsymbol{\Sigma}_\ell = \operatorname{diag}\left[\sigma_r^2, \sigma_\varphi^2\right]$. For this application, we assume that we obtain measurements from each sensor synchronously, *i.e.*, for each timestep, k, we observe $\mathbf{v}_k^o$ and ${}^{\mathring{l}}\boldsymbol{\ell}_{1:Nl_k}$.

With regard to Sec. 3, a data sample of size N consists in $\{\mathbf{v}_{1:N}^o,\, {}^{\mathring{l}}\boldsymbol{\ell}_{1:Nl_{1:N}}\}$, the calibration variable in $\boldsymbol{\Theta} = \{{}^r\mathbf{T}_l\}$, and the nuisance variable in $\boldsymbol{\Psi} = \{{}^w\mathbf{T}_{r_{1:N}},\, {}^w\boldsymbol{\ell}_{1:Nl}\}$. Hence, the estimation problem can thus be formulated as the computation of the marginal posterior density $f({}^r\mathbf{T}_l \mid \mathbf{v}_{1:N}^o,\, {}^{\mathring{l}}\boldsymbol{\ell}_{1:Nl_{1:N}})$, from the joint posterior density $f({}^r\mathbf{T}_l,\, {}^w\mathbf{T}_{r_{1:N}},\, {}^w\boldsymbol{\ell}_{1:Nl} \mid \mathbf{v}_{1:N}^o,\, {}^{\mathring{l}}\boldsymbol{\ell}_{1:Nl_{1:N}})$.

We can obtain an initial estimate for ${}^w\mathbf{T}_{r_{2:N}}$ by successively applying the discretized motion model in Equ. 44 with the odometry measurements, $\mathbf{v}_{1:N}^o$, starting from an arbitrary location, ${}^w\mathbf{T}_{r_1}$. The initial guess for ${}^r\mathbf{T}_l$ comes from a measuring tape and an angle finder. For ${}^w\boldsymbol{\ell}_{1:Nl}$, we can use the inverse model of Equ. 46 using the initial guesses for ${}^w\mathbf{T}_{r_{1:N}}$ and ${}^r\mathbf{T}_l$.

4.2 Observability Analysis

Before we delve into experimental evaluation, we shall perform an a priori analysis of the model. After an intuitive observability analysis, we shall validate our thoughts by the examination of the Jacobian matrix structure and the cost function graph. We note that this analysis is not necessary for our algorithm to work properly, but nevertheless provides valuable insights for interpreting the experimental results and for a proper understanding of the behavior of our method.

4.2.1 Forward Model Analysis

We shall first start this analysis by considering a subset of the estimation problem, *i.e.*, the estimation of robot's poses using odometry measurements. This simple example aims at illustrating the procedure that will drive the observability analysis of the full estimation problem.

Extending the discussions in Sec. 3, observability pertains to the injectivity of the forward model, *i.e.*, whether it is possible to uniquely determine an inverse model. Assuming a sample of odometry data, $\mathbf{v}_{1:N}^o$, we inspect the joint forward model, $\mathbf{v}_{1:N}^o = G_o\left({}^w\mathbf{T}_{r_{1:N}}\right)$, that generates it. In $G_o\left(\cdot\right)$, ${}^w\mathbf{T}_{r_{1:N}}$ forms the domain of the function. If distinct instances of ${}^w\mathbf{T}_{r_{1:N}}$ map to the same outputs $\mathbf{v}_{1:N}^o$, the parameters are not observable. In other words, the inverse function $G_o^{-1}\left(\cdot\right)$ is not uniquely determined by $G_o\left(\cdot\right)$.

For the sake of the example, if we withdraw the non-holonomic constraint on the lateral linear velocity of the robot, there are an infinite number of robot's poses, ${}^w\mathbf{T}_{r_{1:N}}$, that generate the same odometry outputs. For instance, the set of robot's poses with identical orientations along the second axis of the robot's frame will always be reflected by a constant zero output on the odometry sensors. Consequently, robot's poses are not observable. On the other hand, the zero velocity constraint on the second component of the linear velocity in the robot frame disallows these configurations and results in full observability. Our informal explanation is in line with the analysis conducted in [32], which suggests a pose parameterization in polar coordinates to solve the problem.

Along this line of thoughts, we can proceed to the full joint forward model including the landmark positions and calibration parameters,

$$\begin{pmatrix} \mathbf{v}^o_{1:N} \\ {}^{\mathring{l}}\boldsymbol{\ell}_{1:Nl_{1:N}} \end{pmatrix} = G\left({}^w\mathbf{T}_{r_{1:N}}, {}^w\boldsymbol{\ell}_{1:Nl}, {}^r\mathbf{T}_l\right). \tag{47}$$

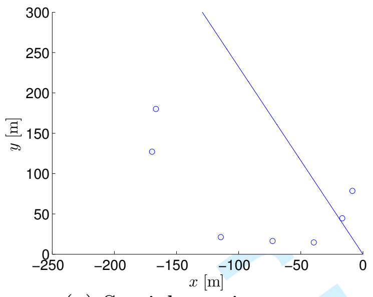
Because the model is under-constrained, there are an infinite number of elements of the domain of $G(\cdot)$ that map to the same element in its codomain. For instance, any affine transformation applied simultaneously to ${}^w\mathbf{T}_{r_{1:N}}$, ${}^w\boldsymbol{\ell}_{1:Nl}$, and ${}^r\mathbf{T}_l$, gives rise to identical outputs. If the relative poses, ${}^w\mathbf{T}_{r_{1:N}}$, have the same orientation, an arbitrary translation applied to ${}^r\mathbf{T}_l$ and to ${}^w\boldsymbol{\ell}_{1:Nl}$ with respect to $\mathcal{F}_W$ has no influence on the outputs. This situation corresponds to the robot driving in a straight line or standing still, with the LRF coordinate system translated along with the landmark positions. Although this rough analysis does not assert an exact figure, the identification of these unobservable directions can guide our experimental evaluation.

In the physical world, the inputs and outputs of $G(\cdot)$ are random quantities associated with a continuous probability density function. Although the probability for the realization of any of these degenerate cases is zero by definition, we might be arbitrarily close to one of them and our estimation problem will appear to be ill-conditioned. For this reason, it only makes sense to speak about numerical observability when a real physical system is involved.

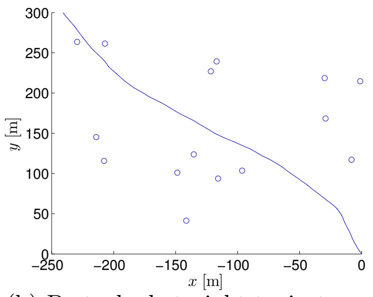
4.2.2 Jacobian Matrix Structure

Following on this informal analysis, we shall now examine if the structure of the Jacobian matrix supports our claims regarding the identified unobservable cases. As can be understood from the model in Sec. 4.1, no data appears in the Jacobian matrix. We can thus evaluate it for any realization of the latent variables given an unobserved data sample.

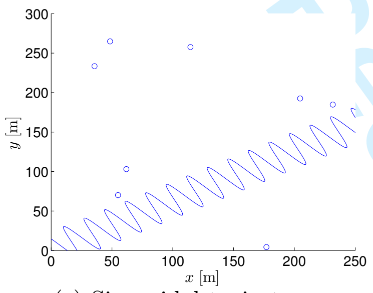
For this analysis, we concentrate on the cases depicted in Fig. 1. The landmark positions, ${}^w\boldsymbol{\ell}_{1:Nl}$, are sampled uniformly within a region around the robot's trajectory, ${}^w\mathbf{T}_{r_{1:N}}$, and the calibration variable, ${}^r\mathbf{T}_l$, is chosen arbitrarily. The trajectory is perfectly straight in Fig. 1a, but in Fig. 1b we slightly perturbed the straight path to resemble noisy odometry received from a robot driving in a straight line. From a linear algebra perspective, the evaluated Jacobian matrix from Fig. 1b is a perturbed version of the one from Fig. 1a. We shall finally focus on the situation sketched in Fig. 1c where the robot's trajectory, ${}^w\mathbf{T}_{r_{1:N}}$, is modeled as a sinusoidal path and hence contains multiple changes of orientation.



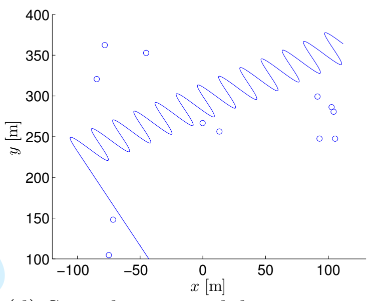
(a) Straight trajectory.



(b) Perturbed straight trajectory.



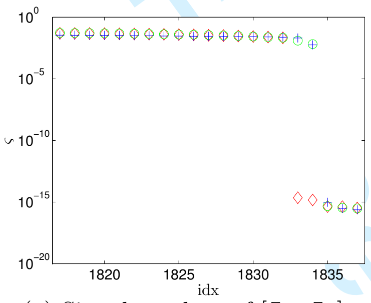
(c) Sinusoidal trajectory.



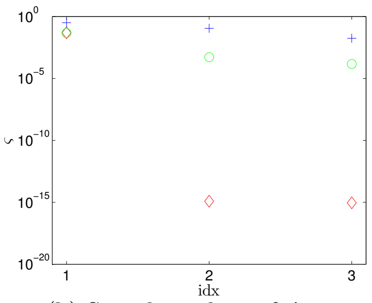
(d) Straight-sinusoidal trajectory.

Figure 1: Examples of robot trajectories used in our simulation experiments. The blue line corresponds to the robot's path, ${}^{w}\mathbf{T}_{r_{1:N}}$, and the blue circles to the landmark positions, ${}^{w}\boldsymbol{\ell}_{1:Nl}$. In Fig. 1a, only the rotational part of ${}^{r}\mathbf{T}_{l}$ is observable. In Fig. 1b, corresponding to noisy odometry when the robot is actually driving straight, the translational part of ${}^{r}\mathbf{T}_{l}$ appears observable due to noise in the Jacobian matrix. Fig. 1c illustrates the fully observable case when the robot undergoes general motion, and Fig. 1d shows the combination of a partially and fully observable cases.

By selecting a reasonable data sample size, we can analyze the structure of the full scaled Jacobian matrix, $[\mathbf{J}_{\psi}, \mathbf{J}_{\theta}]$, by SVD. The computed singular values for the three cases are shown in Fig. 2a. For all cases, the three singular values clamped at zero correspond to the degrees of freedom (two for translation and one for orientation) induced by the lack of global anchor of the model. In the straight trajectory case, two singular values associated with the translation of ${}^{r}\mathbf{T}_{l}$ and ${}^{w}\boldsymbol{\ell}_{1:Nl}$ remain at zero. In the perturbed straight trajectory, these two singular values depart from zero, which shall lead to an incorrect rank determination. The sinusoidal path being well-behaved, the singular values are legitimately above zero. After the SVD analysis of the full Jacobian matrix, we display in Fig. 2b the singular values of the marginal Jacobian matrix, $\mathbf{A}_{\theta}$. This plot conveys essentially the same information and thus validates our intuition.



(a) Singular values of $[\mathbf{J}_{\psi}, \mathbf{J}_{\theta}]$.

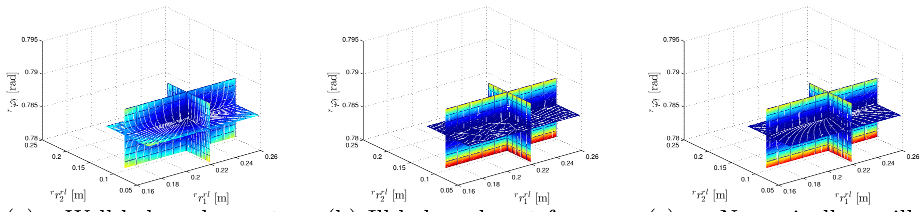


(b) Singular values of $\mathbf{A}_{\theta}$.

Figure 2: The 20 smallest singular values of $[\mathbf{J}_{\psi}, \mathbf{J}_{\theta}]$ (Fig. 2a) and of $\mathbf{A}_{\theta}$ (Fig. 2b) for the straight (red diamonds), perturbed straight (green circles), and sinusoidal (blue pluses) trajectory. In the perturbed straight scenario, the noisy matrix entries lead to two singular values clearly departing from zero, incorrectly rendering the translation parameters of the LRF observable.

4.2.3 Cost Function Visualization

To conclude this a priori analysis, it is worth picturing the cost function defined by Equ. 6 as a function of the calibration parameters, ${}^{r}\mathbf{T}_{l}$. In Fig. 3a, we have fixed the nuisance variables, ${}^{w}\mathbf{T}_{r_{1:N}}$ and ${}^{w}\boldsymbol{\ell}_{1:Nl}$, corresponding to the sinusoidal path in Fig. 1c and evaluated the cost function at different values of the calibration variable, ${}^{r}\mathbf{T}_{l}$. In this fully observable situation, we notice that the cost function is well-behaved, *i.e.*, the gradient lines are convergent. To illustrate the behavior of the cost function in a partially observable setting, we display in Fig. 3b the cost function corresponding to the straight trajectory in Fig. 1a. In this plot, we have applied the same translation to ${}^{r}\mathbf{T}_{l}$ and ${}^{w}\boldsymbol{\ell}_{1:Nl}$, resulting in identical costs for the translation parameters of ${}^{r}\mathbf{T}_{l}$. Consequently, the gradient lines in this plane are divergent and a nonlinear optimization algorithm without regularization will most likely fail. Finally, Fig. 3c demonstrates the numerically unobservable situation depicted in Fig. 1b. Although the input data contains just a slight deviation of the previous case, we witness here that the cost function appears well-behaved.



(a) Well-behaved cost function. (b) Ill-behaved cost function. (c) Numerically ill-behaved cost function.

Figure 3: Cost function for the fully observable (Fig. 3a), partially observable (Fig. 3b), and numerically unobservable (Fig. 3c) scenario. The planes converge at the ground truth value for the calibration parameters, low cost values are blue, higher cost values move through cyan, yellow, and red, and the gradient of the cost function is represented by the white arrows. Although Fig. 3c is a noisy version of Fig. 3b, it incorrectly appears well-behaved.

4.3 Simulation Experiments

As we have an analytical formulation of the forward model for each sensor, we can select any configuration of the latent variables and generate measurements using their data distribution. For generating robot's poses, ${}^{w}\mathbf{T}_{r_{1:N}}$, we have chosen a sinusoid function, parameterized by an amplitude, A, and a frequency, f, providing at will straight to highly curved trajectories. This allows us to explore the sliding scale of numerical observability, *i.e.*, the completely unobservable case when $A = 0$, weakly observable cases when A is small, and fully observable cases when A is large.

4.3.1 Numerical Rank and Observability Detection

In a first experiment, we report in Fig. 4 the evolution of the singular values of the scaled marginal Jacobian matrix, $\mathbf{A}_{\boldsymbol{\theta}}$, as a function of the odometry noise in the straight trajectory example depicted in Fig. 1a. For each noise level, we have averaged 10 simulations runs of 500 seconds. The odometry variances, σ_v^2 and σ_ω^2, range from 0 to 0.2. In the three plots, we observe that the singular values evolve linearly with the noise. For ς_1, the noise induces a slight decrease, which shall play no role in the numerical rank determination. Since we know from the analysis in Sec. 4.2 that ς_2 and ς_3 should be zero, an appropriate threshold for $r_{\epsilon_{\boldsymbol{\theta}}}$ will be in this case $\epsilon_{\boldsymbol{\theta}} \approx \sigma_v^2/30$.

To conclude this part on the numerical rank and observability, we display in Fig. 5 the evolution of the singular values of $\mathbf{A}_{\boldsymbol{\theta}}$ as a function of the amplitude, A, of the sinusoidal trajectory for a fixed sample size $N = 5000$, $\sigma_v^2 = 10^{-4}$, and $\sigma_\omega^2 = 10^{-4}$. We have selected here a conservative numerical rank threshold,

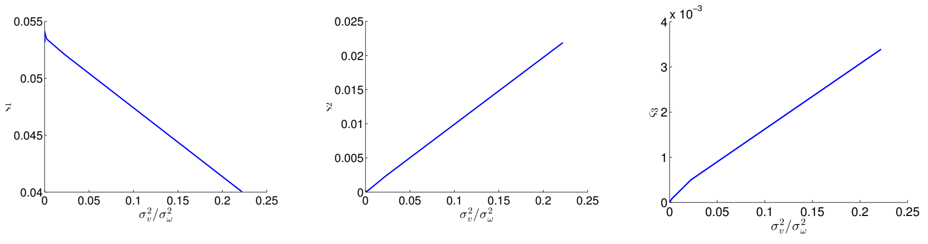


Figure 4: Evolution of the singular values, $\varsigma_{1:3}$, of $\mathbf{A}_{\boldsymbol{\theta}}$ as a function of odometry noise on a straight trajectory. In this scenario, ς_2 and ς_3 should be theoretically clamped at zero.

$\epsilon_{\boldsymbol{\theta}} = 10^{-5}$, such that the calibration parameters become fully observable for an amplitude of approximatively $A = 0.4$ [m].

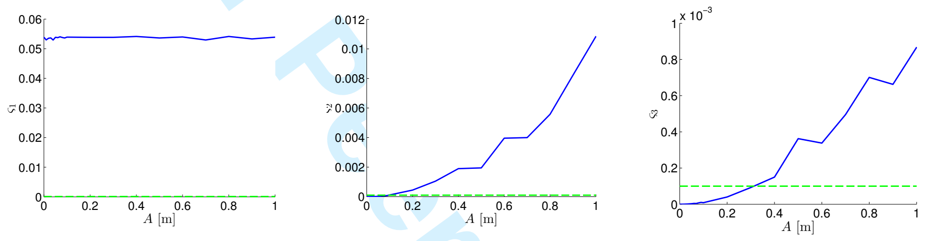


Figure 5: Evolution of the singular values (blue line), $\varsigma_{1:3}$, of $\mathbf{A}_{\boldsymbol{\theta}}$ as a function of the amplitude, A, of a sinusoidal path. With the selected numerical rank threshold (green line), $\epsilon_{\boldsymbol{\theta}} = 10^{-5}$, the calibration parameters become fully observable at $A \approx 0.4$ [m].

In all our experiments, the numerical rank, $r_{\epsilon_{\boldsymbol{\psi}}}$, of $\mathbf{J}_{\boldsymbol{\psi}}$ (rank deficiency of 3 for the global transformation of all the frames) was correctly detected by the RRQR decomposition without any particular attention to the threshold $\epsilon_{\boldsymbol{\psi}}$ determined by the machine epsilon.

4.3.2 Qualitative Evaluation

We shall now demonstrate the overall behavior of our algorithm on some illustrative examples. It is for instance of interest to monitor the measurement batches considered as informative and whether the selection is consistent with our intuition. We shall also visualize the evolution of the posterior distribution over the calibration parameters as new batches are added. We will stick to the three scenarios presented thus far, *i.e.*, a straight trajectory (Fig. 1a), a straight trajectory followed by a sinusoidal path (Fig. 1d), and a sinusoidal path (Fig. 1c).

In Fig. 6, we summarize the relevant qualitative results of our simulation experiments. The first column corresponds to a straight (partially observable), the second to a sinusoidal (fully observable), and the third to the combination of a straight and sinusoidal trajectory. In the first row, we show the selected informative batches, along with the estimated robot's poses and landmark positions. Although these estimates are not of primary interest, their proximity to the ground truth is a pleasing indicator of the performance of our algorithm. For visualization, the estimates were aligned with the ground truth by aligning the landmarks [10]. Out of 25 batches, the straight trajectory used 4, the sinusoidal 11, and the straight-sinusoidal trajectory 15. In order to reveal the effect of batch selection on the calibration parameters, we display in the second to fourth rows, the evolution of the parameters as a function of the incoming batches. As expected, we witness that the parameters remain at their initial guess until they become observable. Finally, we show in the last row the evolution of the information gain as new batches are added to the online estimator. The behavior of the entropy difference concurs with our intuition.

4.3.3 Quantitative Evaluation

To complete the simulation experiments, we compare the performance of our algorithm against a full-batch Levenberg-Marquardt (LM) algorithm and an EKF. We generate a sinusoidal trajectory with varying amplitude such that it covers partial (low amplitude) to full observability (high amplitude) of the calibration parameters. To get significant statistical results, we repeat the experiment 100 times for a given amplitude and use the same initial conditions for the three methods at each run. While keeping identical calibration parameters over the entire experiment, we sample new landmark positions and orientation of the robot's trajectory at each run. The sinusoid function amplitudes range from $A = 0$ [m] to $A = 1$ [m], with steps of 0.01 [m] until $A = 0.1$ [m], and 0.1 [m] until $A = 1$ [m].

The estimated calibration parameters for all methods are displayed in Fig. 7. As we have noticed that the EKF is highly sensitive to the concentration of the initial prior on the calibration parameters, we have included three versions with different priors for comparison. While a tight prior induces little changes on the initial guess or a bias, a loose prior yields instable estimates with many outliers. The best-performing EKF has required a tedious fine-tuning. While our online batch method leaves the unobservable parameters at their initial guess, the LM and EKF perform poorly in that situation. In fully observable conditions (from $A \approx 0.5$ [m]), the three methods yield similar results, but our online batch algorithm uses on average less than 50% of the available data. It can be noted that we have chosen a conservative numerical rank threshold so that the parameters get updated only if there is enough confidence in the data. As we discount data in comparison to other methods, the estimates produced by our algorithm have a slightly larger spread.

One key question to ask yourself when looking at Fig. 7 is: *what behavior do you want from a calibration algorithm?* In cases that are completely unob-

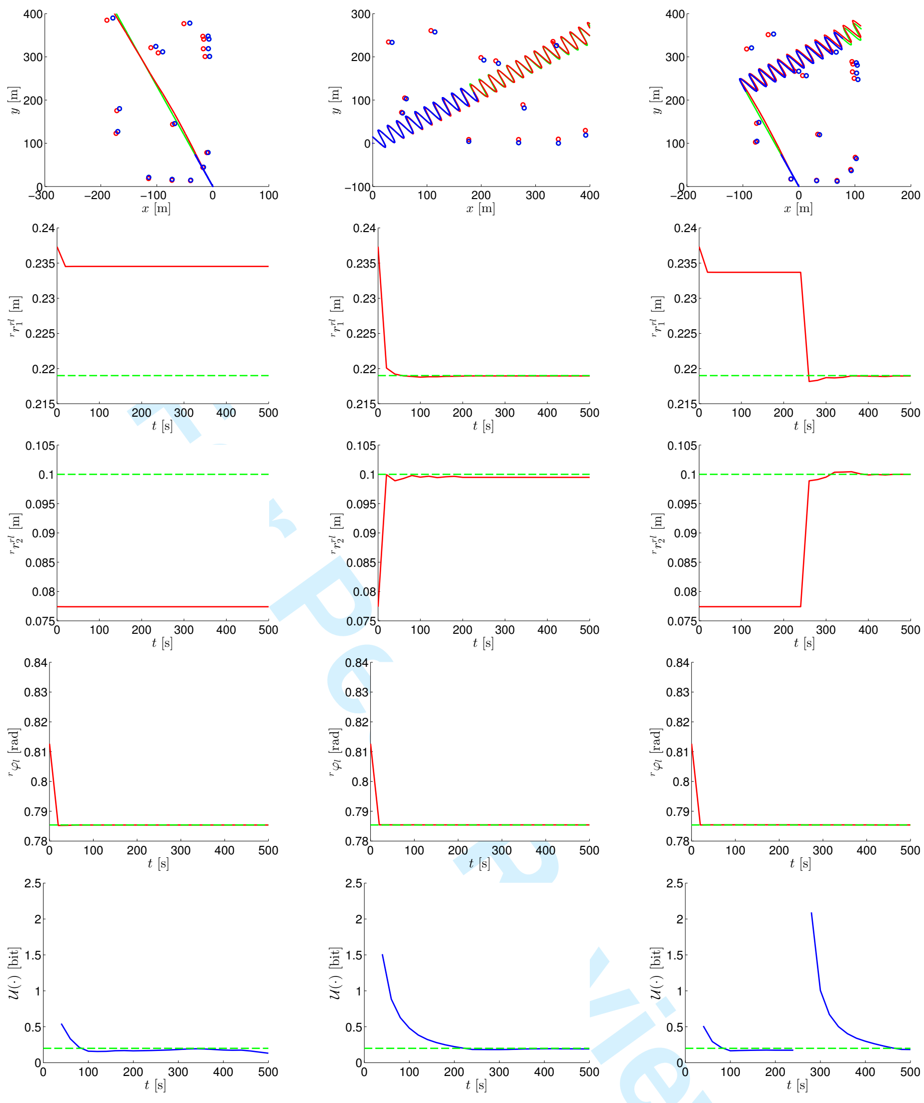


Figure 6: Qualitative results for the simulation experiments. In the first row, the ground truth trajectory, ${}^w\mathbf{T}_{r_{1:N}}$, is shown in green with initial guess in red, ground truth landmark positions, ${}^w\boldsymbol{\ell}_{1:Nl}$, in green circles with initial guess in red circles, trajectory estimate, ${}^w\widehat{\mathbf{T}}_{r_{1:N}}$, in blue, and landmark positions estimate, ${}^w\widehat{\boldsymbol{\ell}}_{1:Nl}$, in blue circles. The second, third, and fourth rows show the calibration parameter estimates, ${}^r\widehat{\mathbf{T}}_l = ({}^r\widehat{r}_1^{rl}, {}^r\widehat{r}_1^{rl}, {}^r\widehat{\varphi}_l)$ (red), as a function of the incoming batches, with the ground truth, ${}^r\mathbf{T}_l$, in green. The last row shows the information gain as a function of the incoming batches, with the information gain threshold, $\delta_{\mathcal{U}}$, in green. The first column is representative for a partially observable, the second for a fully observable, and the last for a combination of partially and fully observable scenario.

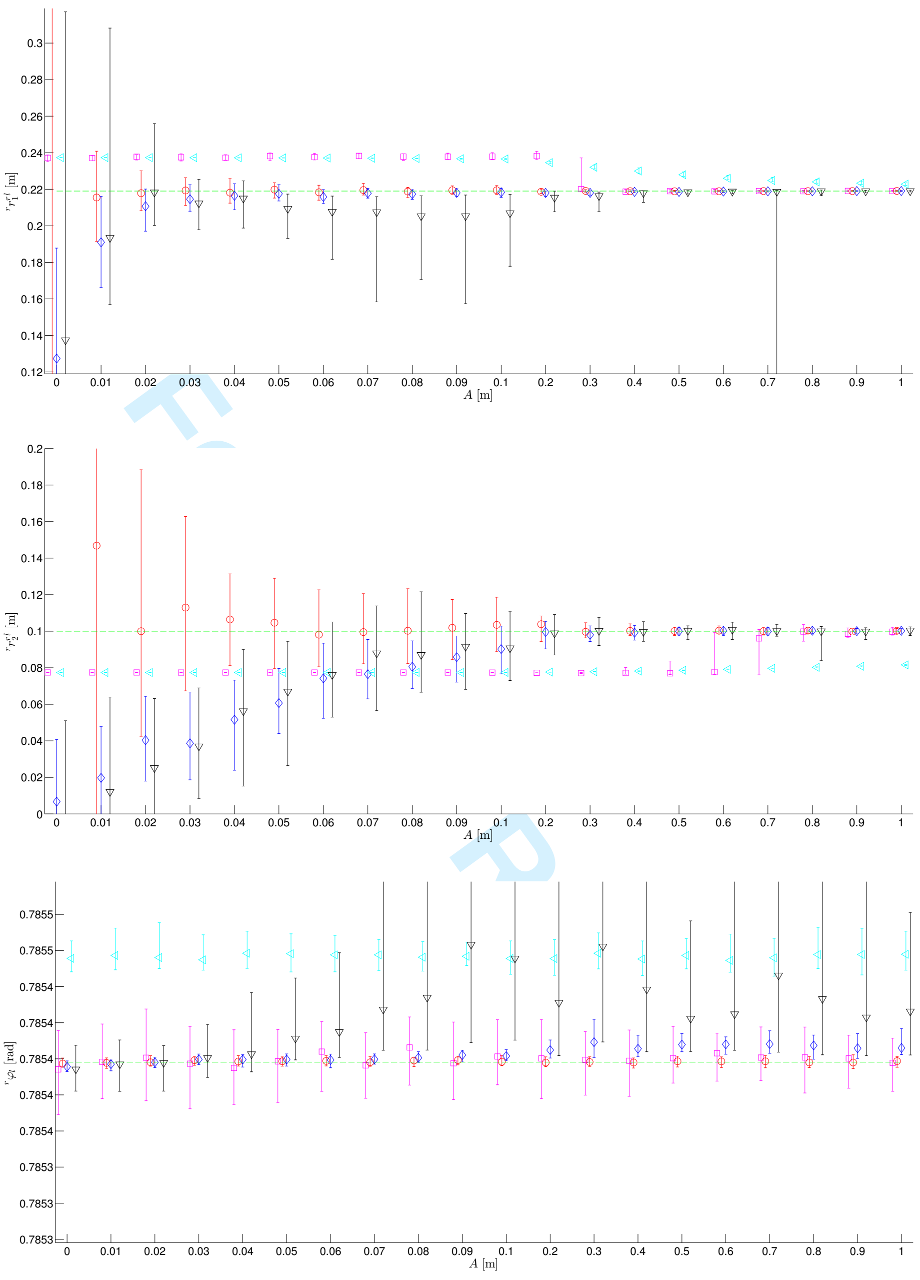


Figure 7: Evolution of the calibration parameters at different sinusoid amplitudes for LM (red circles), EKF with tuned prior (blue diamonds), EKF with tight prior (cyan triangles), EKF with loose prior (black triangles), and our online batch algorithm (magenta squares). The green lines represent the ground truth parameters, the markers the medians, and the error bars the first and third quantiles.

servable or weakly observable, the EKF and batch algorithms make a terrible mistake; both algorithms fit to the noise, resulting in widely varying estimates for the unobservable calibration parameters. This happens for different reasons. The EKF simply takes a random walk, fitting to noise in each step. The magnitude of this walk is determined by the strength of the prior. The batch algorithm computes the low curvature of the cost space along these axes and takes large jumps towards physically implausible values that minimize the cost function.

For the estimates produced by the batch algorithm, the median is mostly near to the true value, but the variability is absolutely undesirable for an online, safety-critical system (see for instance Fig. 8 that displays the maximum absolute errors for the calibration parameters). The three variations of the EKF show the difficulty in tuning the parameters for an application so that it will converge without biasing the results. In many papers about self-calibration, authors suggest to add a simple zero-velocity motion model to calibration parameters to allow them to change in the EKF over time. We have not done that for these experiments, but we believe that this represents another extremely difficult tuning process: getting the motion model just right so that it allows slow variation in your parameters without allowing excessive drift.

In contrast, our algorithm, behaves exactly as we would like; it listens to the incoming data, only updating parameters when it is convinced that the evidence is *signal* and not noise. In all cases, our algorithm produces low variance results, either around the initial guess, or near the true value.

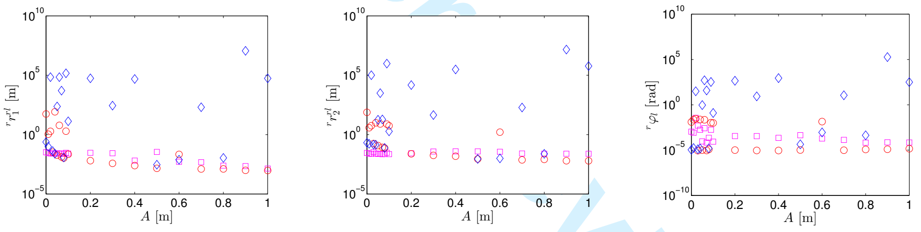


Figure 8: Maximum absolute errors for the calibration parameters at different sinusoid amplitudes for LM (red squares), EKF with tuned prior (blue diamonds), and our online batch algorithm (magenta circles).

4.4 Real-world Experiments

In order to validate our method on real-world data, we have used the "Lost in the Woods Dataset" provided with the courtesy of Tim Barfoot [44]. This dataset contains approximately 20 minutes of a robot driving amongst a forest of tubes which serve as landmarks. The ground truth comes from a motion capture system that tracks robot motion and tube locations. For the calibration parameters, we have only access to ${}^r r_1^{rl} = 0.219$ [m] that was roughly measured with a tape. We assume the others are implicitly set to 0 (*i.e.*, ${}^{r}r_2^{rl} = 0$ [m] and ${}^{r}\varphi_l = 0$ [rad]).

In a first effort, we investigate the influence of the online parameter set, $\mathcal{P}_O$, on the behavior of our algorithm. The aim of this experiment is to determine adequate values for these parameters. As we pointed out in Sec. 3.5 and as can be seen in the top-left plot of Fig. 9, the information gain threshold, $\delta_{\mathcal{U}}$, and the batch size, ΔN, are coupled. Here, we have first inserted a large batch of 150 [s] into our estimator and subsequently computed the information gain as a function of newly inserted batches of varying sizes. This plot shows that larger batch sizes yield higher information gain and provides a way of selecting $\delta_{\mathcal{U}}$ for a given ΔN. Using this relation, we have run our algorithm with varying batch sizes and we show in the top-middle and top-right plot of Fig. 9 the maximum number of flops and memory footprint, and in the bottom plots the final calibration estimates, along with their 3-σ bounds plotted around the full-batch LM output. In absence of accurate ground truth, we can consider this latter as the gold standard estimate. These plots provide a guideline for choosing the online parameters, *i.e.*, we require low memory footprint, a minimal number of flops, accuracy with respect to LM, and precision.

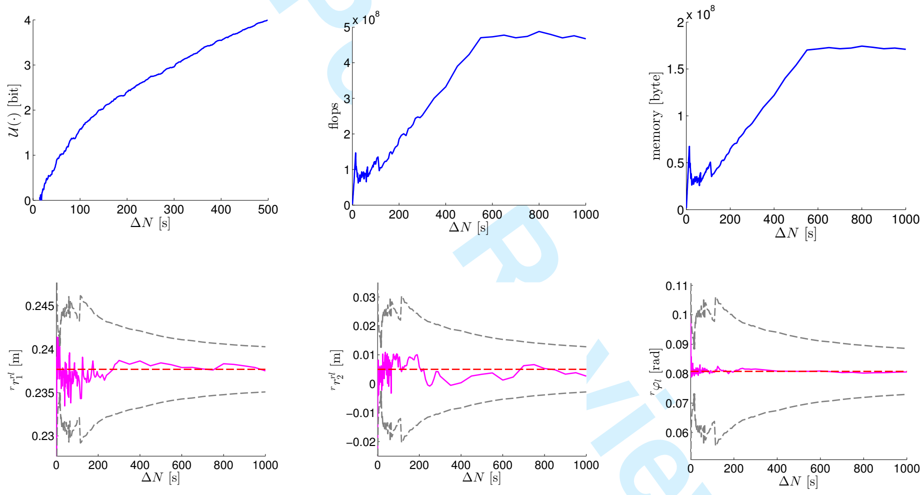


Figure 9: Online parameters analysis. The top-left plot shows the information gain, the top-middle plot the maximum number of flops, the top-right plot the maximum memory footprint, and the bottom plots the online batch calibration estimates (magenta), with their 3-σ bounds (grey) and LM (red), as a function of batch sizes.

From the parameter analysis, we select a batch size of $\Delta N = 30$ [s] and an information gain threshold of $\delta_{\mathcal{U}} = 0.2$ [bit], and we display in Fig. 10 the qualitative results of our online batch algorithm. Using less than 50% of the measurements, we could recover accurate landmark positions, robot's poses for the informative batches, and calibration parameters. We show in Fig. 11 the evolution of the calibration parameters as a function of the incoming batches with a comparison to LM and EKF. From this plot, we observe that the initial guess for ${}^{r}\mathbf{T}_{l}$ from the dataset was relatively correct. The dataset being well-behaved, *i.e.*, all the parameters are observable, the three methods give comparable results, as was already noticed in the simulation experiments. The higher variability in the second translational component, ${}^{r}r_{2}^{rl}$, was also observed in Fig. 7. We suspect that the lack of lateral motion of the robot renders it weakly observable.

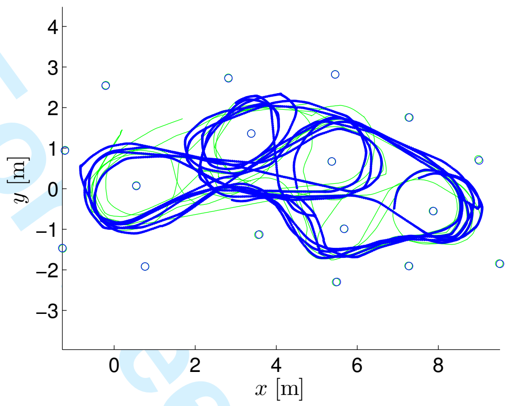


Figure 10: Qualitative results for the estimation of robot's poses and landmark positions for the "Lost in the Woods Dataset". The ground truth trajectory, ${}^{w}\mathbf{T}_{r_{1:N}}$, is shown in green, ground truth landmark positions, ${}^{w}\boldsymbol{\ell}_{1:Nl}$, in green circles, trajectory estimate, ${}^{w}\widehat{\mathbf{T}}_{r_{1:N}}$, in blue, and landmark positions estimate, ${}^{w}\widehat{\boldsymbol{\ell}}_{1:Nl}$, in blue circles.

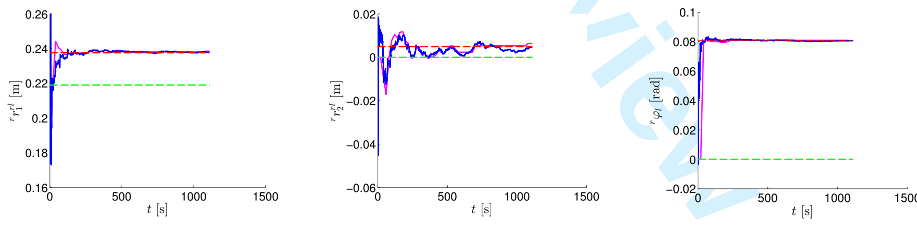


Figure 11: Evolution of the calibration parameter estimates, ${}^{r}\widehat{\mathbf{T}}_{l} = ({}^{r}\widehat{r}_{1}^{rl}, {}^{r}\widehat{r}_{1}^{rl}, {}^{r}\widehat{\varphi}_{l})$, as a function of the incoming batches for the "Lost in the Woods Dataset". The initial guess, ${}^{r}\mathbf{T}_{l}^{(0)}$, is shown in green, the estimates for LM in red, for EKF in blue, and for our online batch algorithm in magenta.

5 Conclusion

In this paper, we have investigated the development of an automated method that deals with degenerate configurations occurring during calibration runs. By means of matrix analysis, our algorithm identifies unobservable directions in the calibration parameter space and locks them at their initial guess until they become fully observable. To leverage the otherwise intractable computational and space complexity of nonlinear optimization methods, we have proposed an information-theoretic scheme that evaluates the utility of data sample batches for the calibration.

5.1 Contributions

The first contribution of this work concerns an online observability measure of the parameter space. Whilst the majority of state-of-the-art approaches are either not aware of singularities or perform algebraic observability analysis, we have presented a method based on rank-revealing QR and singular value decompositions of the Fisher information matrix that separates the calibration parameters into observable and unobservable subspaces. Splitting up the parameter space, our algorithm only updates the observable dimensions, leaving the rest intact. Along these lines, we have developed the concept of numerical observability, the counterpart of numerical rank in linear algebra, as opposed to the purely algebraic view in control theory. Through our simulated experiments, we have illustrated the difficulty in assessing observability in presence of noisy sensor data and shown the robustness of our algorithm to properly execute this task in comparison to other methods.

In a second effort, we have validated the theoretical intuitions on the behavior of the entropy difference between prior and posterior distributions when new data batches enter the estimator through simulated and real-world experiments. Linked to the observability, this measure perfectly fulfills our expectations for the selection of informative data and is hence a sound complement to nonlinear least-squares methods.

5.2 Discussion and Future Work

The presented algorithm for the online self-calibration of robotic sensors has several limitations and therefore offers vast opportunities for further research directions that we shall discuss in the rest of this section.

The key limitation in the current implementation lies in the assumption that the calibration parameters remain constant during the calibration run. For the proper deployment of our method into consumer-level sensors, we must be able to cope with slowly or abruptly varying parameters. As an example, we can consider an autonomous vehicle with cameras embedded into the side mirrors. Whenever the user opens or closes the doors, the relative poses of the cameras might change due to the mechanical shock. It is nevertheless possible to detect these cases by monitoring sudden variations in the prediction error of the estimator and restarting the calibration algorithm from scratch. A probably smarter approach would be to add a forgetting factor to our algorithm. Using this strategy, the forgetting factor would give exponentially less weight to older data batches, discarding them eventually, to favor fresh information. The covariance matrix of the estimates should be scaled accordingly to avoid a too strong convergence of our online algorithm, leading to discounting any novel data.

A current limitation of our algorithm is the dependence on a reasonable initial guess for the calibration parameters. To reach the fully unsupervised goal we have in mind, the calibration algorithm should be able to bootstrap itself from scratch. Along the same lines, our implementation uses Gauss-Newton update steps in the nonlinear optimization, which is hence only guaranteed to converge if the initial guess is close enough to a local minimum. To withdraw this assumption, we suggest a trust-region method such as Powell's dog-leg algorithm [38]. This class of methods uses a combination of gradient descent steps for global convergence and Gauss-Newton steps for local convergence.

The determination of the thresholds for the numerical rank detection requires further analysis. As we have seen in Sec. 4, since these thresholds are clearly dependent on the system noise, there must be a way to compute them analytically. From our experiments, we have nevertheless witnessed that these thresholds only need to be calculated once per sensor setup, which remains a manageable task for an expert before the deployment of the algorithm in the field. Skimming through the literature, we have observed that even experts [8] in numerical analysis employ a heuristic that "has worked well in practice". It is important to recall here that an incorrect numerical rank detection can lead to utterly wrong estimates as was demonstrated in Sec. 4. In that regard, we suggest a conservative approach, such that these thresholds may underestimate the numerical rank rather than overestimating it.

In the presence of outliers or other effects that violate the assumption of normality, we believe we can adopt an M-estimator framework to smoothly dampen the effect of outliers in the estimation process. This is motivated by the simplicity at which it can be integrated into a nonlinear optimization method. However, whenever the initial guess is not close enough to a local minimum, the M-estimator may incorrectly consider data points as being outliers and thus dramatically bias the optimization. We may address this issue by adapting the M-estimator at each iteration, *i.e.*, being conservative at the beginning and stricter as the algorithm converges to a local minimum. If the sensor data is absolutely not normally distributed, our method may fail and would require an ad-hoc modification, *e.g.*, accounting for multimodal distributions.

Finally, we will begin the work of integrating this algorithm into our complicated robotics systems for navigating in dynamic urban environments [27, 11] where safety and robustness are of the highest priority.

Acknowledgement

This work is supported in part by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant #269916 (V-Charge) and grant #610603 (EUROPA2).

References

[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[2] Jonathan Brookshire and Seth Teller. Automatic calibration of multiple coplanar sensors. In *Proc. of the Robotics: Science and Systems Conference (RSS)*, 2011.

[3] Jonathan Brookshire and Seth Teller. Extrinsic calibration from per-sensor egomotion. In *Proc. of the Robotics: Science and Systems Conference (RSS)*, 2012.

[4] Stephen L. Campbell and Carl D. Meyer. *Generalized Inverses of Linear Transformations*. Society for Industrial and Applied Mathematics (SIAM), 2009.

[5] Andrea Censi, Luca Marchionni, and Giuseppe Oriolo. Simultaneous maximum-likelihood calibration of odometry and sensor parameters. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2008.

[6] Tony F. Chan. Rank revealing QR factorizations. *Linear Algebra and its Applications*, 88-89:67–82, 1987.

[7] Tony F. Chan and Per Christian Hansen. Some applications of the rank revealing QR factorization. *SIAM Journal on Scientific and Statistical Computing*, 13(3):727–741, 1992.

[8] Timothy A. Davis. Algorithm 915: SuiteSparseQR: Multifrontal multithreaded rank-revealing sparse QR factorization. *ACM Transactions on Mathematical Software*, 38(1):1–24, 2011.

[9] Olivier D. Faugeras, Quang-Tuan Luong, and Stephen J. Maybank. Camera self-calibration: theory and experiments. In *Proc. of the European Conference on Computer Vision (ECCV)*, 1992.

[10] Paul D. Fiore. Efficient linear solution of exterior orientation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 23(2):140–148, 2001.

[11] Paul Furgale, Ulrich Schwesinger, Martin Rufli, Wojciech Derendarz, Hugo Grimmett, Peter Mühlfellner, Stefan Wonneberger, Julian Timpner Stephan Rottmann, Bo Li, Bastian Schmidt, Thien Nghia Nguyen, Elena Cardarelli, Stefano Cattani, Stefan Brüning, Sven Horstmann, Martin Stellmacher, Holger Mielenz, Kevin Köser, Markus Beermann, Christian Häne, Lionel Heng, Gim Hee Lee, Friedrich Fraundorfer, René Iser, Rudolph Triebel, Ingmar Posner, Paul Newman, Lars Wolf, Marc Pollefeys, Stefan Brosig, Jan Effertz, Cédric Pradalier, and Roland Siegwart. Toward automated driving in cities using close-to-market sensors, an overview of the V-Charge project. In *Proc. of the IEEE Intelligent Vehicles Symposium (IV)*, 2013.

[12] Chao Gao and John R. Spletzer. On-line calibration of multiple LIDARs on a mobile vehicle platform. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2010.

[13] Andrew Gelman, John B. Carlin, Hal S. Stern, and Donald B. Rubin. *Bayesian Data Analysis*. Chapman and Hall/CRC, 2003.

[14] Alan George and Esmond Ng. On the complexity of sparse QR and LU factorization of finite-element matrices. *SIAM Journal on Scientific and Statistical Computing*, 9(5):849–861, 1988.

[15] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, 1996.

[16] Per Christian Hansen. The truncated SVD as a method for regularization. *BIT Numerical Mathematics*, 27(4):534–553, 1987.

[17] Per Christian Hansen. The 2-norm of random matrices. *Journal of Computational and Applied Mathematics*, 23(1):117–120, 1988.

[18] Per Christian Hansen. *Rank-Deficient and Discrete Ill-Posed Problems: Numerical Aspects of Linear Inversion*. Society for Industrial and Applied Mathematics (SIAM), 1998.

[19] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Proc. of the 4th Alvey Vision Conference*, 1988.

[20] Lionel Heng, Bo Li, and Marc Pollefeys. CamOdoCal: Automatic intrinsic and extrinsic calibration of a rig with multiple generic cameras and odometry. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[21] Claude Jauffret. Observability and Fisher information matrix in nonlinear regression. *IEEE Transactions on Aerospace and Electronic Systems*, 43(2):756–759, 2007.

[22] Andrew H. Jazwinski. *Stochastic Processes and Filtering Theory*. Academic Press, 1970.

[23] C.T. Kelley. *Iterative Methods for Optimization*. Society for Industrial and Applied Mathematics (SIAM), 1999.

[24] Jonathan Kelly and Gaurav S. Sukhatme. Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration. *The International Journal of Robotics Research (IJRR)*, 30(1):56–79, 2011.

[25] Jae-Hean Kim and Myung Jin Chung. Absolute motion and structure from stereo image sequences without stereo correspondence and analysis of degenerate cases. *Pattern Recognition*, 39:1649–1661, 2006.

[26] Rainer Kuemmerle, Giorgio Grisetti, and Wolfram Burgard. Simultaneous calibration, localization, and mapping. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011.

[27] Rainer Kummerle, Michael Ruhnke, Bastian Steder, Cyrill Stachniss, and Wolfram Burgard. A navigation system for robots operating in crowded urban environments. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[28] Pierre Lébraly, Eric Royer, Omar Ait-Aider, and Michel Dhome. Calibration of non-overlapping cameras - application to vision-based robotics. In *Proc. of the British Machine Vision Conference (BMVC)*, 2010.

[29] Jesse Levinson and Sebastian Thrun. Unsupervised calibration for multibeam lasers. In *Proc. of the International Symposium on Experimental Robotics (ISER)*, 2010.

[30] Dennis Victor Lindley. On a measure of the information provided by an experiment. *The Annals of Mathematical Statistics*, 27(4):879–1212, 1956.

[31] Will Maddern, Alastair Harrison, and Paul Newman. Lost in translation (and rotation): Rapid extrinsic calibration for 2D and 3D LIDARs. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2012.

[32] Agostino Martinelli, Davide Scaramuzza, and Roland Siegwart. Automatic self-calibration of a vision system during robot motion. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2006.

[33] Stephen J. Maybank and Olivier D. Faugeras. A theory of self-calibration of a moving camera. *International Journal of Computer Vision (IJCV)*, 8(2):123–151, 1992.

[34] Jerome Maye, Paul Furgale, and Roland Siegwart. Self-supervised calibration for robotic systems. In *Proc. of the IEEE Intelligent Vehicles Symposium (IV)*, 2013.

[35] Chia-Hsiang Menq, Jin-Hwan Born, and Jim Z. Lai. Identification and observability measure of a basis set of error parameters in robot calibration. *Journal of Mechanisms, Transmissions, and Automation in Design*, 111:513–518, 1989.

[36] Faraz M. Mirzaei and Stergios I. Roumeliotis. A Kalman filter-based algorithm for IMU-camera calibration: Observability analysis and performance evaluation. *IEEE Transactions on Robotics*, 24(5):1143 – 1156, 2008.

[37] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective.* MIT Press, 2012.

[38] Michael J.D. Powell. A new algorithm for unconstrained optimization. In J.B. Rosen, O.L. Mangassarian, and K. Ritter, editors, *Nonlinear Programming*, pages 31–65. Academic Press, 1970.

[39] Martin Rufli, Davide Scaramuzza, and Roland Siegwart. Automatic detection of checkerboards on blurred and distorted images. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2008.

[40] Mark Sheehan, Alastair Harrison, and Paul Newman. Automatic self-calibration of a full field-of-view 3D n-laser scanner. In *Proc. of the International Symposium on Experimental Robotics (ISER)*, 2010.

[41] Mark Sheehan, Alastair Harrison, and Paul Newman. Self-calibration for a 3D laser. *The International Journal of Robotics Research (IJRR)*, 31(5):675–687, 2012.

[42] Peter F. Sturm and Stephen J. Maybank. On plane-based camera calibration: A general algorithm, singularities, applications. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1999.

[43] Albert Tarantola. *Inverse Problem Theory and Methods for Model Parameter Estimation.* Society for Industrial and Applied Mathematics (SIAM), 2005.

[44] Chi Hay Tong, Paul Furgale, and Timothy D. Barfoot. Gaussian process Gauss-Newton: Non-parametric state estimation. In *Proc. of the Conference on Computer and Robot Vision (CRV)*, 2012.

[45] James Underwood, Andrew Hill, and Steve Scheding. Calibration of range sensor pose on mobile platforms. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2007.

[46] Qilong Zhang and Robert Pless. Extrinsic calibration of a camera and laser range finder (improves camera calibration). In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2004.

[47] Zhengyou Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *Proc. of IEEE International Conference on Computer Vision (ICCV)*, 1999.