DISS. ETH NO. 26905

TOWARDS GENERIC CALIBRATION FOR MOBILE ROBOTS

A thesis submitted to attain the degree of

DOCTOR OF SCIENCES of ETH ZURICH

(Dr. sc. ETH Zurich)

presented by

HANNES SOMMER

Dipl.-Math.

born on December 29, 1980
citizen of Germany

accepted on the recommendation of

Prof. Dr. Roland Siegwart, Examiner
Prof. Dr. Stephan Weiss, Co-examiner

2020

Autonomous Systems Lab
Department of Mechanical and Process Engineering
ETH Zurich
Switzerland

© 2020 Hannes Sommer. All rights reserved.

Abstract

Every robotic system has a set of parameters — scale factors, sensor locations, link lengths, communication latencies, etc. — that must be known with some accuracy for state estimation, planning, and control to perform well. After construction of such a system, some parameters are typically not known accurately enough due to the high cost of both accurate and precise production. Furthermore, some will change over the lifetime of a robot due to normal wear and tear or accidents. The effects of assuming poor parameter values range from degraded performance to critical safety issues.

The big calibration dream for mobile robots would be that the systems calibrated themselves after their assembly and while being operational without requiring any supervision or special environments. Unfortunately, this goal is inherently hard to achieve, mostly due to inaccurate or incomplete knowledge of the environment the sensors perceive, which cannot be easily improved on without accurate calibration. This circular dependency leads to a problem type similar to Simultaneous Localization and Mapping (SLAM), called Simultaneous Calibration, Localization, and Mapping (SCLAM). In a way, it is a truly bigger problem, because it has SLAM as a subproblem. In contrast to SLAM, there are typically less computational resource constraints and a calibration solution may restrict itself to smaller maps. It is also more difficult, because — in contrast to SLAM — self-calibration cannot build on something like a sufficient calibration that already mitigates many of the production flaws and unknowns before SLAM takes place. This is no surprise considering that one key purpose of calibration is to make SLAM more accurate and easy.

This thesis contributes to the dream of *lifelong self-calibration* for mobile robots by focusing on some of the generic sub-problems which make the development of such powerful calibration systems hard to realize today. Specifically, it contributes (i) to a potential deprecation of a second quaternion multiplication introduced in the 1980s that causes significant confusion and failures until today in the practice of state estimation, control, and calibration by means of a thorough comparison and principled analysis, (ii) to the continuous-time state estimation / calibration for non vector space Lie-group valued variables such as SO(3), by making an idea for unit-length quaternion valued B-splines accessible for nonlinear optimization, (iii) an efficient, easy to use, and safe automatic differentiation of differentiable models, by proposing the Block Automatic Differentiation (BAD) technique, (iv) an efficient way to gain accurate and detailed information on timing of global shutter cameras and time-of-flight LIDARs, (v) a novel way to associate observations of LIDAR-like sensors for the purpose of calibration.

In addition to the above theoretic contributions, several c++ libraries, usually with python exports, were developed, maintained, extended, and used for the experiments and evaluations of the corresponding contributions. These were mainly, a library for fast and optimizable Lie-group valued b-splines based on (ii), a BAD library and a prototype for a much faster library both based on and evaluated in (iii), a tiny library for automatic propagation of kinematic quantities such as angular velocity through a kinematic chain while supporting automatic differentiation with respect to all quantities involved building on the BAD concept, a timestamp translation library to be used in sensor drivers required for (iv), a library for modular layered parameter trees to assemble multiple parameter trees in an unified and transparent way, and a modular c++ continuous-time calibration framework, Object-Oriented Modular Abstract Calibration Toolbox (OOMACT), based on all the libraries and contributions before.

OOMACT was primarily developed to serve as a basis for the lifelong self-calibration and fault-detection system developed for the European Robotic Pedestrian Assistant 2.0 (EUROPA2) project, which was simultaneously developed on top of it. Additionally, it was used to realize the local calibration step in a new library dedicated to the *hand-eye* calibration problem. This local calibration step — yielding highly accurate results quickly even for poor initial guesses — could be realized with only a few lines of c++ code thanks to the highly configurable and modular nature of the toolbox.

Within these applications we could confirm that the contributions of this thesis could indeed greatly simplify several of the otherwise very tedious or difficult tasks necessary to develop high quality automatic calibration systems.

Zusammenfassung

Jeder Roboter hat eine Reihe von Parametern - Skalierungsfaktoren, Sensorpositionen, Verbindungslängen, Kommunikationslatenzen usw. —, die mit einer gewissen Genauigkeit bekannt sein müssen, damit Zustandsschätzung, Planung und Steuerung gut funktionieren. Nach der Konstruktion eines solchen Systems sind einige Parameter typischerweise nicht genau genug bekannt, was an den hohen Kosten für eine genaue und präzise Produktion liegt. Außerdem werden sich einige im Laufe der Lebensdauer eines Roboters aufgrund von normalem Verschleiß oder Unfällen ändern. Die Auswirkungen von zu schlechten Parameterwerten reichen von verminderter Leistung bis hin zu kritischen Sicherheitsproblemen.

Der große Kalibrierungstraum für mobile Roboter wäre, dass sich die Systeme nach ihrer Montage und während ihres Betriebs selbst kalibrieren, ohne dass eine Überwachung oder spezielle Umgebungen erforderlich sind. Leider ist dieses Ziel von Natur aus schwer zu erreichen, meist aufgrund ungenauer oder unvollständiger Kenntnis der von den Sensoren wahrgenommenen Umgebung, die sich ohne genaue Kalibrierung nicht einfach verbessern lässt. Diese zirkuläre Abhängigkeit führt zu einem ähnlichen Problemtyp wie Simultaneous Localization and Mapping (SLAM), genannt Simultaneous Calibration, Localization, and Mapping (SCLAM). In gewisser Weise ist es ein wirklich größeres Problem, weil es SLAM als Unterproblem hat. Im Gegensatz zu SLAM gibt es typischerweise weniger Beschränkungen bei den Rechenressourcen und eine Kalibrierungslösung kann sich auf kleinere Karten beschränken. Sie ist auch schwieriger, weil — im Gegensatz zu SLAM — die Selbstkalibrierung nicht auf so etwas wie einer ausreichenden Kalibrierung aufbauen kann, die bereits viele der Produktionsfehler und Unbekannten abmildert, bevor SLAM stattfindet. Dies ist keine Überraschung, wenn man bedenkt, dass ein Hauptzweck der Kalibrierung darin besteht, SLAM genauer und einfacher zu machen.

Diese Arbeit leistet einen Beitrag zum Traum von *lebenslanger Selbs-Kalibrierung* für mobile Roboter, indem sie sich auf einige der generischen Teilprobleme konzentriert, die die Entwicklung solch leistungsfähiger Kalibrierungssysteme heute schwer zu realisieren machen. Insbesondere trägt sie bei, (i) zu einer potentiellen Aufgabe einer alternativen Quaternionenmultiplikation, die in den 1980er Jahren eingeführt wurde und bis heute in der Praxis der Zustandsschätzung, Steuerung und Kalibrierung für erhebliche Verwirrung und Fehler sorgt, durch einen gründlichen Vergleich und eine prinzipielle Analyse, (ii) zu der zeitkontinuierliche Zustandsschätzung / Kalibrierung für Lie-gruppenwertige Variablen wie SO(3), indem eine Idee für einheitslängen-quaternionenwertige B-Spline Kurven für die nichtlineare Optimierung zugänglich gemacht wird, (iii) eine effiziente, einfach zu handhabende und sichere automatische Differenzierung differenzierbarer Modelle, indem die Block Automatic Differentiation (BAD)-Technik vorgeschlagen wird, (iv) einen effizienten Weg, um genaue und detaillierte Informationen über das Timing von Global-Shutter-Kameras und Time-of-Flight-LIDARs zu erhalten, (v) eine neuartige Möglichkeit, Beobachtungen von LIDAR-ähnlichen Sensoren zum Zweck der Kalibrierung zuzuordnen.

Zusätzlich zu den oben genannten theoretischen Beiträgen wurden mehrere C++-Bibliotheken, meist mit Python-Export, entwickelt, gepflegt, erweitert und für die Experimente und Auswertungen der entsprechenden Beiträge verwendet. Diese waren hauptsächlich, eine Bibliothek für schnelle und optimierbare Lie-gruppenwertige B-Spline Kurven basierend auf (ii), eine BAD-Bibliothek und ein Prototyp für eine viel schnellere Bibliothek, die beide auf (iii) basieren und evaluiert wurden, eine winzige Bibliothek zur automatischen Propagierung kinematischer Größen wie z. B. der Winkelgeschwindigkeit durch eine kinematische Kette bei gleichzeitiger Unterstützung der automatischen Differenzierung in Bezug auf alle beteiligten Größen, aufbauend auf dem BAD-Konzept, eine Bibliothek für die Übersetzung von Zeitstempeln zur Verwendung in Sensortreibern, die für (iv) benötigt werden, eine Bibliothek für modular geschichtete Parameterbäume, um mehrere Parameterbäume auf einheitliche und transparente Weise zusammenzustellen und ein modulares C++-Framework für die zeitkontinuierliche Kalibrierung, Object-Oriented Modular Abstract Calibration Toolbox (OOMACT), das auf allen vorherigen Bibliotheken und Beiträgen basiert.

OOMACT wurde in erster Linie entwickelt um als Grundlage zu dienen für das System zur lebenslangen Selbstkalibrierung und Fehlererkennung, das gleichzeitig für das Projekt European Robotic Pedestrian Assistant 2.0 (EUROPA2) entwickelte wurde. Darüber hinaus wurde es verwendet, um den lokalen Kalibrierungsschritt in einer neuen Bibliothek zu implementieren, die dem *hand-eye*-Kalibrierungsproblem gewidmet ist. Dieser lokale Kalibrierungsschritt — der auch bei schlechten Ausgangswerten schnell zu hochgenauen Ergebnissen führt — konnte dank der hochgradig konfigurierbaren und modularen Natur der Toolbox mit nur wenigen Zeilen C++-Code realisiert werden.

Innerhalb dieser Anwendungen konnten wir bestätigen, dass die Beiträge dieser Arbeit in der Tat mehrere der sonst sehr langwierigen oder schwierigen Aufgaben, die zur Entwicklung hochwertiger automatischer Kalibrierungssysteme erforderlich sind, stark vereinfachen können.

Acknowledgements

June 30, 2020 *Hannes Sommer*

Over the past years I've benefited greatly from various kinds of support, much of which I probably never realized. From what I realized and still remember, I'd like to thank my supervisors, Paul Furgale, Gabriel Agamennoni, Igor Gilitschenski, and Juan Nieto. I'm very thankful for everything that they taught me and helped me with over the years. In particular I want to deeply thank Juan Nieto for his spectacularly relaxed, caring, and humane way of supervising and leading, from which I benefited greatly. In doing so he is setting a great example. I am especially grateful to Zachary Taylor for his readiness to invest significantly into helping me out of difficult situations far more than once! Francis Colas, I'd like to thank for setting an exceptional scientific example for me and for all the wonderful discussions and for initiating the dessert group meetings.

From my very early days at the ASL I still feel gratitude to François Pomerleau, for his welcoming and courageous nature, Andreas Breitenmoser, for his curiosity, Stephan Weiss, for his encouragement and inspiration, and Cédric Pradalier, and Paul Furgale for their trust and support.

Of my many colleagues I'd like to thank foremost and in no particular order, Abel Gawel, Renaud Dubé, Gregory Hitz, Mark Pfeiffer, Raghav Khanna, Gianluca Cesari, François Pomerleau, Raphael Schranz, Ulrich Schwesinger, Stéphane Magnenat, Andreas Breitenmoser, Jérôme Maye, Fadri Furrer, Rik Bähnemann, Michael Blösch, Michael Burri, Jörn Rehder, Ralf Kaestner, They all contributed greatly to making the ASL a place I enjoyed spending time at. In particular I'd like to thank Mark, Renaud and Abel for their invaluable support.

I want to thank Landis Markley for his support and for setting such a great example as a human and an honest scientist by how he reacted to my publication and criticism.

I'd like to thank Prof. Roland Siegwart and his exceptional deputies, Cédric Pradalier, Paul Furgale, Michael Bosse, and Juan Nieto for their trust, their support, and for running and providing such a motivating and inspiring work environment, that almost magically brings together so many great individuals.

A tremendous and essential support came almost invisibly from ETH Zurich, D-MAVT, and the staff at the ASL, Stefan Bertschi, Amir Melzer, Konrad Meyer, Michael Riner, Luciana Borsatti, Cornelia Della Casa, as well as from the technical and the magnificent cleaning personnel. To all I am very thankful for their great, competent and friendly support.

Importantly, I thank my parents and my close friends for their kind backing, advice and mental support throughout the entire period. Anne-Katrin Schlegel, I thank additionally for her various practical support including all the extremely helpful proofreading.

Financial Support

This project has received funding from the European Union’s Seventh Framework Programme (FP7) for research, technological development and demonstration under grant agreement no 610603, EUROPA2.

Contents

Abstract **i**

Zusammenfassung **iii**

Acknowledgements **v**

Preface **1**

1. Introduction **3**
- 1.1. Motivation and Objectives . . . 3
  - 1.1.1. The dream and drama of calibrating robots . . . 3
  - 1.1.2. Scope of this thesis . . . 4
- 1.2. Relevance . . . 6
- 1.3. Approach . . . 6
  - 1.3.1. Theoretical . . . 6
  - 1.3.2. Implementation . . . 7

2. Contribution **11**
- 2.1. Conceptual contributions . . . 11
  - 2.1.1. Paper I . . . 12
  - 2.1.2. Paper II . . . 13
  - 2.1.3. Paper III . . . 15
  - 2.1.4. Paper IV . . . 16
- 2.2. Applications . . . 19
  - 2.2.1. Hand-eye calibration . . . 19
  - 2.2.2. Report V . . . 22
- 2.3. List of Publications . . . 24
  - 2.3.1. Journals . . . 24
  - 2.3.2. Conferences . . . 24
  - 2.3.3. Technical report . . . 25

3. Conclusion and Outlook **27**

A. CONCEPTUAL CONTRIBUTIONS **31**

PaperI: Automatic Differentiation on Differentiable Manifolds as a Tool for Robotics **33**
1. Introduction 34
2. Theory 36
3. Implementation 42
4. Experiments 44
5. Conclusions 47
6. Acknowledgement 48

PaperII: Why and How to Avoid the Flipped Quaternion Multiplication **49**
1. Introduction 50
2. Notation 52
3. Background: Duality of Rotation Representations 53
4. The Proposed Alternative Solution 53
5. Literature Review 54
6. Rationale against Shuster's Multiplication 56
7. Recipes 62
8. Conclusion 65

PaperIII: Continuous-Time Estimation of attitude using B-splines on Lie groups **67**
1. Introduction 68
2. Related Work 70
3. Theory 72
4. Experiments 86
5. Results 91
6. Conclusions 101

PaperIV: A Low-Cost System for High-Rate, High-Accuracy Temporal Calibration for LIDARs and Cameras **107**
1. Introduction 108
2. Related work 109
3. Design 111
4. Experiments and Results 116
5. Conclusions 121

B. APPLICATIONS **125**

ReportV: Life-Long Self-Calibration and Fault Detection **127**
1. Introduction 129
2. The EUROPA2 calibration problem 130
3. Our Approach 133
4. Software 139
5. Evaluation 140
6. Unresolved issues 155

7. Conclusion 156

C. APPENDICES 159

A. Supplemental work on rotation quaternions 161
A.1. Proof sketch: Correctness of migration recipe 161
A.2. Proof: Hamilton QM-convention always yields more similarity 162
A.3. Representing 3d proper rotations with matrices 163

B. Further unpublished proofs 167
B.1. Correctness proof for the hierarchical task optimization algorithm 167

C. Errata 173
C.1. Erratum of Online Self-Calibration for Robotic Systems 173

Bibliography 175

Curriculum Vitae 187

Preface

This is a cumulative doctoral thesis which comprises 4 published papers and one technical report.

It begins with an introduction, Chapter 1 which first motivates the work and then explains the overall scope and fundamental calibration approach that orients and motivates the individual contributions. In Chapter 2, we briefly go through all the papers and explain their context within the overall scope, their contribution, and mutual interrelation. The last chapter, Chapter 3, provides conclusion and outlook.

In Part A the conceptual papers are provided while the applied papers and the technical report are composing Part B. In the Part C various proofs and derivations are collected that did not fit into the publications.

Chapter 1

Introduction

1.1. Motivation and Objectives

Every robotic system has a set of parameters — scale factors, sensor locations, link lengths, communication latencies, etc. — that must be known with some accuracy for state estimation, planning, and control to perform well. Some parameters will not be given accurately enough due to production inaccuracies and some will change over the lifetime of a robot due to normal wear and tear or accidents. In the best case, incorrect parameter values degrade performance. In the worst case, they cause critical safety issues.

1.1.1. The dream and drama of calibrating robots

In theory, a powerful calibration system is beneficial for almost any robotic system: it allows higher accuracy in performing motion or interaction tasks or allows reducing a systems costs by tolerating lower quality sensors or less accurate manufacturing, and it can make the difference between a task being feasible, and safe, or not.

However, a modern (mobile) robotic application easily asks for a set of traits for its calibration solution that is very difficult or impossible to realize in practice, given the currently available building blocks and limited resources. The following defines predicates representing some of these attractive traits of a calibration system or concept. A calibration system is called:

- *target-less*: iff it can be applied in the robot's normal operation environment rather than an environment specifically prepared with objects of well known geometry – so called calibration targets,
- *unsupervised*: iff it does not need supervision by an external system or a human,
- *intrinsic*: iff it can calibrate parameters which are purely intrinsic to a component that gets mounted to the robot as a whole, such as sensor, or actuator (for instance the lense distortion of a camera's lense),

- *extrinsic*: iff it can calibrate parameters between components such as mutual location (e.g., relative pose) or differences in their individual clocks,
- *incremental*: iff it can incorporate new information over time while gaining certainty or updating calibration,
- *lifelong*: iff it can handle changes in calibration parameters over time,
- *accurate*: iff it can provide accurate estimates given sufficient data,
- *global*: iff it does converge globally and therefore does not require an initial guess close enough to a correct calibration given enough data,
- *computation / data efficient*: iff it can deliver useful results given a relatively small amount of computation / data,
- *conservative*: iff calibration parameters only get updated when there is sufficient evidence to justify a change considering the cost for a calibration update (this is a stronger requirement than being a robust estimator),
- *easy to develop*: iff the difficulty and complexity of it is well manageable for the developer of a concrete solution,
- *generic*: iff a wide range of sensors / actuator configurations can be calibrated with it or at least integrated conceptually.

The ultimate calibration approach for most applications would in fact combine all of the above traits. Unfortunately, such a solution is very far away from the state of the art because combining even small subsets of them becomes very difficult with the methods and techniques currently available. For instance, being global and efficient are natural opponents and so are being target-less and accurate, or being easy and generic and many more.

One consequence of the inherent difficulty of the overall problem is that typically a given practical approach only combines a few of these traits. Another consequence is that most contemporary mobile robotic systems lack a truly satisfying calibration system. Some of the conflicts can be reduced by developing good tools and building blocks. The goal of this thesis is to contribute some building blocks and tools that bring potential applications at least a bit closer to this big dream of calibration, mostly by reducing the practical tradeoff between some of the attractive traits.

1.1.2. Scope of this thesis

To limit the large scope of making calibration easier we leave out some of the attractive traits from the last section early on.

One attractive trait which causes significant conceptual conflicts with most of the other traits is being global. Local approaches can be widely used in practice because today's design and manufacturing processes typically provide rather good initial guesses for all the calibration parameters. Furthermore, accidents that render the calibration from before useless as initial guess for a local calibration approach are likely to violate operational or safety requirements beyond pure calibration requirements.

At the same time computational efficiency is considered a rather soft requirement because computational power can be safely assumed to be increasingly available and energy efficient in the future, in particular since the calibration problem allows highly parallel solutions. We allow our approach to require moderately powerful computers at the level of a powerful laptop.

One of the dominant problem separations (in the sense of the divide and conquer method) in calibration is the one between extrinsic and intrinsic calibration. The underlying assumption is that intrinsics can be calibrated separately per component and consequentially do not need being calibrated in the assembled robotic system or can be calibrated individually while being mounted. Purely extrinsic calibration approaches expect the intrinsic calibration parameters as a given and static input. Most generic approaches aim at extrinsic calibration because the intrinsic parameters are highly specific. A lifelong calibration approach cannot ignore that some intrinsic calibration parameters might also change over time, such as the wheel diameter of a wheel-odometry. This thesis is further excluding parameters that can safely be considered static intrinsics of individual components and focuses mostly on the spatiotemporal extrinsic calibration of complex systems.

Overall the *scope* of this thesis can now be stated as making *unsupervised, local, incremental, lifelong, target-less, multi-sensor, extrinsic calibration for mobile robots*, more easy while keeping the system configuration rather open and expecting only moderately high computational and memory resources available for the calibration task. More specifically, the goal is to contribute to reducing some of the current conflicts between the attractive traits from Section 1.1.

This scope is well aligned with one of the work packages of the European Robotic Pedestrian Assistant 2.0 (EUROPA2) project.

Within this high level scope, the specific theoretic endeavors shall be motivated and guided by an imaginary prototypical practical goal, also inspired by EUROPA2. Namely, the development of a Maximum Posteriori Estimation (MPE) based, generic and efficient calibration software library / tool with the following features:

- Capable of lifelong online operation
- Full spatiotemporal extrinsic calibration for sensors and actuators, incorporating hand measurement or manufacturing specifications as initial calibration
- Supporting the integration of all popular sensors types: wheel odometry, gyroscopes, accelerometer, vision and range finder type, as well as locomotion and simple joint actuations
- Easy setup and input / modification of system model, calibration strategies, and failure case detection

In fact, a prototype of such a calibration library, Object-Oriented Modular Abstract Calibration Toolbox (OOMACT), was developed in parallel to this thesis along with various other libraries serving it as smaller building-blocks. This library served greatly as a device to find practical challenges with the selected approach and as a test bench for potential solutions.

While developing and employing it, we faced various conceptual challenges. Some of those challenges are specifically addressed in this thesis:

1. supporting non-flat continuous-time state spaces components (e.g. orientations),
2. supporting easy prototyping of differentiable measurement and kinematic models through suitable software design,
3. accurate and high frequency inspection of temporal extrinsic calibration for cameras and time of flight LIDARs,
4. overwhelming and prohibitive conventional chaos regarding rotation quaternions,
5. associating LIDAR measurements with one another with support for a probabilistic measurement model.

In addition to these conceptual contributions, the thesis contains contributions applying the approach outlined in Section 1.3 to classical hand-eye calibration in Section 2.2.1, and to the automatic self-calibration work package of the EUROPA2 project in ReportV.

1.2. Relevance

Calibration appears to be a notoriously underestimated problem. Prototypes in industry and research are often calibrated manually by specialist with tedious and slow procedures. This cannot scale to mass production because every single robotic system must be calibrated individually. Another common practice is avoiding to fuse data of sensors to save the effort of extrinsic calibration. This can never unfold the full potential of the hardware and hence will have a sub-optimal cost performance ratio compared to a system with an affordable high quality calibration solution. To make this bad situation worse there is only a small fraction of robotic engineers interested or even specialized in calibration. This is probably because from further away calibration seems to appear to be a mere nuisance. From a little bit closer it starts appearing as a very difficult nuisance that cannot easily be solved satisfyingly with machine learning approaches, because of its purely definition based interface to the rest of the system using the calibration. As a result, industry will face significant costs for (not properly) calibrating mass products. Reducing costs by employing an insufficient calibration solution may even have negative safety implications and could cause accidents. At the same time robotics research is wasting considerable time by employing tedious calibration methods or by chasing after performance problems ultimately caused by bad calibration. Given this situation, every contribution to making useful calibration approaches more accessible and ready to adapt to a given problem with less effort and pitfalls may be considered a valuable contribution of practical relevance for society.

The contributions of this thesis, that all aim at making some typical subproblems of calibration easier to solve in practise, are explained in Chapter 2.

1.3. Approach

1.3.1. Theoretical

For the remainder of this document we adopt the following calibration notation. A system's parametrized model shall define a *calibration space*, $\mathscr{C}$, of possible values for the *calibration* *estimate*, $\hat{\mathbf{c}}$, a *state space*, $\mathscr{X}$, of all possible state estimates and an *input space*, $\mathscr{U}$, of possible (control) inputs to the systems — each at a given instant in time. For simplicity's sake, we will write as if there was a *true* calibration and state within these spaces at any time, knowing that the model is artificial and with it all quantities expressed with respect to it and that the truth will almost surely lie outside what the model can express. The "true calibration" shall mean the elements of the calibration space that – interpreted according to the model – come closest to reality. The split between calibration and state space is not a natural one and must be considered part of a given model. However, the key difference shall be their modeled temporal development. The calibration part is typically modeled to be not at all or very slowly changing or rarely jumping (e.g. through accidents). The state part is its complement. It is often co-estimated with the calibration out of conceptual necessity. The calibration space may also contain hyper or noise parameter components.

As fundamental calibration approach, given such a model, we choose the principled and powerful Bayesian continuous-time MPE of $k \in \mathbb{N}$ mini-batches over non-overlapping time intervals $\mathbf{T}_{i=1}^{k}$. It shall jointly estimate the calibration tuple, $\mathbf{c} \in \mathscr{C}^k$, providing one calibration per mini-batch, and the state trajectory tuple, $\mathbf{x} \in \bigtimes_{i=1}^{k} \mathscr{F}_{\mathscr{X}}(\mathbf{T}_i)$, constrained to finite dimensional function sub spaces[1], given a calibration prior density, $p(c_1)$, observations, $\mathbf{Z}_{i=1}^{k}$, input trajectories $\mathbf{u} \in \bigtimes_{i=1}^{k} \mathscr{F}_{\mathscr{U}}(\mathbf{T}_i)$, and process models providing $p(\mathbf{x}|\mathbf{c},\mathbf{u})$ and $p(\mathbf{c}|c_1,\mathbf{T}_{i=1}^{k})$.

$$\hat{\mathbf{c}}, \hat{\mathbf{x}} = \underset{\mathbf{c} \in \mathscr{C}^k, \mathbf{x} \in \bigtimes_{i=1}^{k} \mathscr{F}_{\mathscr{X}}(\mathbf{T}_i)}{\operatorname{argmax}} \; p(c_1)\, p(\mathbf{c}|c_1,\mathbf{T}_{i=1}^{k})\, p(\mathbf{x}|\mathbf{c},\mathbf{u}) \prod_{i=1}^{k} p(\mathbf{Z}_i|\mathbf{c}_i,\mathbf{x}_i).$$

In non-lifelong calibration systems, usually no changes over time in calibration are considered and $p(\mathbf{c}|c_1,\mathbf{T}_{i=1}^{k})$ is assumed to be concentrated on c_1 for $\forall_{i=2}^{k} c_i$. Otherwise less trivial stochastic process models must be assumed to derive $p(\mathbf{c}|c_1,\mathbf{T}_{i=1}^{k})$ as approximation and the $\mathbf{T}_i$ must be kept short enough to keep the approximation error low. If rare but significant jumps of the calibration parameters are expected, e.g. through accidents, then one should exclude the intervals that probably contain the jumps.

1.3.2. Implementation

Following this mathematical approach we developed a modular and abstract calibration system design that can be realized as a set of software libraries. It resembles a data flow as depicted in Figure 1.1 (overview) and Figure 1.2 (calibration core).

For such a design to be generic in practice it must be very modular. For instance, the input processor of a single data source, e.g. a LIDAR, depends on the middleware (communication) and the sensors (data format) while at the same time sensors of similar type should be modeled with the same code to avoid duplication. Hence there should be input processors modules that connect for instance a generic LIDAR model with a given sensor through a given middleware. Similarly, the preferred representation of state trajectories might depend on the typical dynamics of the system and should be therefore provided by trajectory modules that allow being composed to represent trajectories of composed state spaces. The clients of a state representation, like

---

[1] The usual continuous-time approximation: $\forall_{i=1}^{k} \mathscr{F}_{\mathscr{X}}(\mathbf{T}_i) \subset \mathbf{T}_i \to \mathscr{X} \wedge \dim(\mathscr{F}_{\mathscr{X}}(\mathbf{T}_i)) \in \mathbb{N}$. This can be achieved for instance by using B-splines, as we do in this thesis.

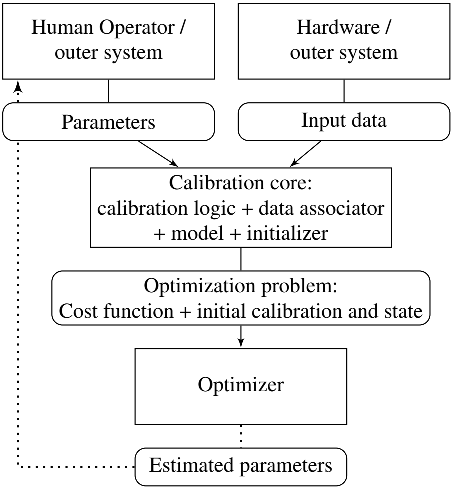


Figure 1.1.: Calibration system data flow. Round and sharp cornered rectangles represent data (including functional data) and processing entities respectively.

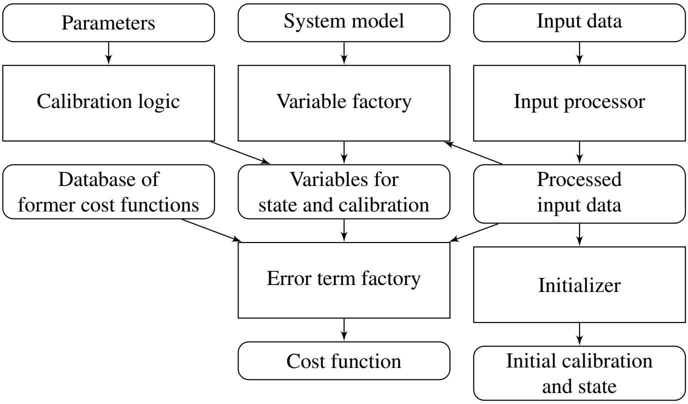


Figure 1.2.: Calibration core

the functions that generate measurement error terms (depending on the state) should be further modules independent of the trajectory choice, requiring an unified access to the relevant state at the measurement time. The measurement time itself might be unknown and required to be estimated due to unknown or random latencies. This requires introducing additional latency variables, possibly per measurement. Their treatment should not be part of the measurement error generator, because otherwise it would need to be implemented for all of them individually.

This level of modularity can be achieved with a toolbox system and a rich set of interfaces between them such that they can be easily composed. A calibration toolbox should contain pre-implemented modules for common tasks but also support easy integration of user provided modules since covering all needs that could be satisfied with this generic approach is unrealistic.

Our prototype corresponding design-wise to Figure 1.2 consists of various types of building blocks. One group is representing sensors, actuators, joints, and frames that compose the *system model*. This model is used by the *error term factory* to automatically build up the *cost function*, i.e. the negative log-likelihood of the *input data* to be minimized. It utilizes generic error terms representing individual latent variables (e.g. measurement errors) with their assumed probability distributions. These error terms depend on state and calibration variables introduced by the *variable factory* based on the model and the input data. To create these variables the variable factory is assembling further building blocks, namely trajectories to represent state-trajectory space components and calibration variables of various types to represent calibration space components. Both calibration and state spaces are here assumed to be Cartesian products of such components. The *calibration logic* is also an extension point with existing implementations for batch MP and a multi stage MP. The task of the calibration logic is to control which data should be used and which variables get estimated at which stage of the overall optimization. Choosing the right calibration logic for a given problem is a challenge, because a simple full batch single stage approach typically only works for sufficient data with rather good initial guesses for the calibration variables and rather low noise levels. How to pick one of the many alternatives is a very open question that is typically answered in practice based on experience and intuition of experts. For a generic library the question is how to provide a flexible input form for the user to specify this logic. The way we implemented this was by providing an easy to use programming / scripting interface with which users can program their calibration logic and give rich automatic feedback to support the user in iterating quickly to improve their calibration logic.

Chapter 2

Contribution

In this chapter the scientific context and contribution of each publication contained in this thesis is presented. Additionally, their mutual interrelations and their relation to the overall goal of this thesis is explained. All represented research and development was conducted in strong collaboration with the co-authors.

We start with the more conceptual contributions, then go through their applications, and finish with a list of all publications I (co-)authored. With *conceptual* I mean developing concepts and tools that are useful for and motivated by the calibration task or some of its many subproblems while typically not being specific to calibration. That does not mean being particularly theoretical in nature. For instance, the last conceptual work, PaperIV, is very practical in comparison to the first three while still developing a concept and tools, which are not only usable for calibration.

2.1. Conceptual contributions

Fully automatic calibration is a very valuable feature for real world mobile robotic systems. However, in practice it is rarely deployed because of the significant investment currently necessary to reach a reliable and useful solution. This is acceptable for short-term research prototypes or small series installation, where one can work instead with semi-automatic calibration conducted only once by experts. However, it is not acceptable for mass production of long-term deployed robots, such as autonomous cars, which even need some form of lifelong calibration in particular if the calibration is safety relevant.

Important reasons for the high cost of automatic calibration are a severe lack of easy to use generic building blocks that could reduce the implementation effort and the lack of robust methods that can promise useful performance in advance without much further research. This thesis contributes to mitigating a few typical stumbling stones that make automatic self-calibration too expensive or unrealistic in practice.

The developed concepts and tools are motivated by but not at all limited to self-calibration. In fact, most of them provide benefits to other problems in robotics such as planning, control, state estimation, mapping or supervised factory calibration. This is also why all the conceptual papers do not directly discuss any applications to self-calibration but instead more immediate and less complex applications serving as test bench for the tools.

2.1.1. Paper I

Hannes Sommer, Cédric Pradalier, and Paul Furgale, "Automatic Differentiation on Differentiable Manifolds as a Tool for Robotics". In *Robotics Research. Springer Tracts in Advanced Robotics*, 2016.

Context

A concrete calibration problem typically comprises variables with small and or highly nonlinear effects on the observed data. Small effects often require the joint consideration of much data to achieve satisfying accuracy. Self-calibration requires joint estimation of the calibration and state and map variables increasing the problem much further. This leads to large optimization problems to be solved in high accuracy calibration. A popular and very efficient generic way to reduce computational complexity for local optimization is to provide analytic derivations with respect to the variables where possible (partial differentiability) in order to compute local updates.

Here an unfortunate tradeoff appears: More accurate calibration of complex systems requires more complex models and more data. But using derivative based optimization in the prototyping phase of the calibration system is the harder and more error-prone and time consuming the more complex the model is. The latter is particularly true if some of the variables live in a non vector space Differentiable Manifold (DM), which typically requires special expertise to be treated correctly. In mobile robotics, at least one such DM is very likely to be part of the calibration problem, for instance S^2, or SO(3).

Contribution

This paper develops one particular well suited approach to automatically provide derivatives for the calibration model prototyping phase for mobile robots. It finds a good compromise between introduced workflow complexity and runtime efficiency while at the same time helping the developer greatly to write correct and modular implementations of the model in at least two ways: first, the derivatives are generated automatically with a pretested procedure out of pretested building blocks, and second, the type system of statically typed languages, such as c++, allows for libraries to elegantly prevent invalid operations that make no sense mathematically, for instance operations that would change the magnitude of an unit-length quaternion. Effectively, such a library can employ the compiler to do mathematical type checks.

We call this approach Block Automatic Differentiation (BAD). It is an extension of the traditional dual number approach to Automatic Differentiation (AD) that incorporates complex (not in the sense of complex numbers but in the sense of value-tuples) variables (blocks) that traditionally would be represented by multiple variables, e.g. 4 real numbers instead of one unit-length quaternion.

It allows an user of a BAD-library to assemble complex functions, such as the maximum a posteriori density of a calibration problem, at runtime by writing typical mathematical expressions, like $a+b$ or $q.\text{rotate}(v)$, while possibly looping over input data of a size determined at runtime (difficult for traditional code generators for derivatives). Internally, it maintains an acyclic directed evaluation graph that can then be traversed behind the scenes to compute values and derivatives as needed while caching intermediate values that otherwise would need to be computed multiple times.

Interrelations

An experimental implementation of the BAD concept is used for almost all experiments and applications mentioned in this thesis. It was a great help in the process of adapting the differentiable models and allowed to try out modified models quickly without the burden of deriving the derivatives or employing tedious code generation procedures that typically create a lot of interfacing overhead. As users of this system, we never had to evaluate the chain-rule ourselves. Only new elemental operations required determining their derivative once to make it available. The error-prone work of combining them was taken care of by the library. Effectively it allowed to explore models that would normally have been avoided simply to save the effort a human would need to put in deriving and debugging them.

The presented runtime performance comparison with a well-tuned traditional AD implementation shows big potential (up to 14 times faster) – when looking at computing individual Jacobians of some test functions. The advantageous effect is rather small ($\sim 13\%$) when comparing the timings for solving the bigger problem of allocating and assembling the entire (sparse) Jacobian of the batch attitude estimation problem we presented in the paper as application. This is because the part done by BAD only has a small share of this particular problem, mostly because computing the Jacobians of the many invocations of rather simple measurement functions is very quick in both ways compared to the allocation and assembly of a big matrix containing all the individual Jacobians. This can change drastically when individual measurement models and the state representation get more complex – for instance, when doing continuous-time (self-)calibration for mobile multi sensor systems as in ReportV. In this sense, the example in the paper sadly fails to showcase the full runtime performance potential of BAD in the calibration domain. However, its performance advantage is not the most important contribution to calibration. Instead, it is the gain in ease of use and failure safety in the prototyping and maintenance phases of a calibration system – while providing sufficient runtime speed for complex models at the same time. Together these advantages enable faster development iterations with more complex models.

2.1.2. Paper II

Hannes Sommer, Igor Gilitschenski, Michael Bloesch, Stephan Weiss, Roland Siegwart, and Juan Nieto , "Why and How to Avoid the Flipped Quaternion Multiplication". In *MDPI Aerospace*, 2018.

Context

Unit quaternions are the state of the art representatives of SO(3), typically modeling attitude, in state estimation and calibration. They are efficient to use and almost isomorphic — merely a double cover of the target Lie group. More specifically, $\mathbf{q}$ and $-\mathbf{q}$ always represent the same element of SO(3).

Surprisingly, the biggest challenge for beginners when employing quaternions to represent rotations is that there is a whole family of competing conventions associated with them, which are often not made explicit. This makes combining formulas and results from different sources error-prone, in particular, when one is not aware of the multitude of conventions.

One binary choice distinguishing two halves of the conventions is particularly troublesome. The difference is very easy to overlook because many formulas are effectively invariant under the change. Almost no source stresses the choice explicitly. Yet, mixing formulas of both choices can easily lead to great failures that are hard to find and understand. It is the choice between the original quaternion multiplication, $\odot$[1], as introduced by Hamilton, and its alternative, $\otimes$, with flipped arguments, i.e. $\mathbf{q} \otimes \mathbf{p} := \mathbf{p} \odot \mathbf{q}$, which is actually a different multiplication since $\odot$ is not commutative.

Contribution

Our contribution was to make aware of this conventional split and to show that it is superfluous and wasteful. Furthermore, we explain how it is linked to the problem which to solve $\otimes$ was introduced for. For this problem, we promote a widely unknown alternative solution, which is inverting all employed quaternions while using the original multiplication, and show its advantage through a thorough comparison.

The fact that today's conventional situation regarding rotation quaternions feels like a dangerous battlefield for engineers might root in the fact that this almost forgotten solution was overlooked or ignored for so long. This is because three possible, attractive features of rotation quaternion-matrix conventions, being (i) passive world to body, (ii) homomorphic[2], and (iii) in sync with the quaternion definition in mathematics, are incompatible as shown by the very influential [146] in 1993. Authors and teachers influenced by this publication had to decide which of the three to sacrifice for the other two. However, the derivation was flawed in the sense that it did not declare all its relevant assumptions. The undeclared binary choice which made the triple incompatible was a single sign in the conversion from rotation vector to rotation quaternion. However, it is one choice that is almost irrelevant in modern applications and should therefore readily be questioned if it can unlock the attractive feature triple — even despite the fact that this inverts the value of all quaternions.

Consequentially, that is what we propose. As an additional incentive, we show that this sign flip yields a level of form similarity with the corresponding matrix formulas that is impossible to achieve with the problematic quaternion multiplication, $\otimes$.

---

[1] The notation is non-standard and only adopted for this discussion for clearer distinction.

[2] Rotation quaternions multiply the same order as corresponding rotation matrices

Interrelations

We suffered from the problem early on, while working on the BAD concept, PaperI, and the unit-length quaternion valued B-splines, PaperIII. The publications, experiments, and implementations of both confirm that our alternative solution works flawlessly in theory and practice.

We believe this publication can help to ultimately heal a conventional wound that caused a lot of trouble in the state estimation and calibration community for far too long.

2.1.3. Paper III

Hannes Sommer, James Richard Forbes, Roland Siegwart, Paul Furgale, "Continuous-Time Estimation of attitude using B-splines on Lie groups". In *Journal of Guidance, Control, and Dynamics*, 2015.

Context

State or calibration space components that are not of vector space type are a well known difficulty for estimation algorithms. Finding good representations for such quantities is non-trivial. For continuous-time state representation this adds additional difficulty because it requires the representation of entire non vector space valued curves, typically with time as curve-parameter. The common interpolation or spline techniques readily available for vector spaces typically make great use of the vector space properties and cannot be carried over easily to more general spaces.

Contribution

In the field of computer graphics the idea to construct unit-length quaternion valued B-splines of arbitrary spline order is already well known ([86]). However, employing this construction for estimation comes with additional problems. In particular, it requires efficient optimization techniques. We found practical solutions for all the additional problems that come with optimizing such B-splines. At the same time we formulated the theory for more general Lie-groups in order for it to be usable beyond unit-length quaternions and to understand where specific properties of quaternions need replacements.

The paper also contains results from simulation based experiments with spacecraft attitude estimation. This problem is much smaller than our calibration approach from Section 1.3 but very well suited to study the usability of the approach for the sub-problem of continuous-time state estimation.

Future work

Some of the questions we investigated empirically with simulated data should allow for more analytic answers. Most notably the following questions:

- *Within which bounds will the initialization method work as intended?* It should be possible to prove within some suitable bounds that given some requirements the proposed initialization will do its intended work. One promising approach could be to assess the maximal difference between the Lie group valued B-spline and the corresponding B-spline in the ambient algebra that is used for the initialization. After all, we are using the very same initialization, except for the succeeding projection onto the embedded Lie group. It should be possible to to bound the change caused by the projection by constraining the velocity of the original signal and the knot spacing.

- *How big is the typical approximation error given a system model and a B-spline?* Given the knots of a B-spline and the system model and inputs an upper bound for the approximation error of unperturbed solutions could be derived. The existing work on $\mathbb{R}^n$ (e.g. [44]) could be a good starting point but would of course require the transfer to Lie groups. Of course, this would loose applicability the bigger the process noise. Deriving similar (probabilistic) bounds for the stochastic system model would require deeper knowledge of stochastic differential equations on differentiable manifolds.

- *How does the measurement noise influence the expected estimation quality?*

  The answer will depend on the spline parameters, knots and order, the Lie group, the measurement density and functions, and the actual trajectory and the system model and its input if used.

  Assuming a problem and an estimator of the kind presented in the paper, an interesting non-sampling based approach could be to propagate the uncertainty in the measurements, Z, through the estimator using the fact that through the argmin an implicit function is defined (assuming a unique solution). It is also a differentiable function assuming smooth enough measurement models. This should allow applying the implicit function theorem to compute the necessary Jacobian of the spline control vertices X with respect to Z. The invertibility precondition of the theorem should correspond to X being weakly observable (in the sense of distinguishable from some neighborhood). Unfortunately, this will require second order Jacobians with respect to the control vertices which could be prohibitive in particular when on a non-vector space Lie group.

  Additionally, this will yield only an approximative answer regarding the estimator's variance induced by the sensor noise. For its actual accuracy, there is the additional problem to consider that the B-splines will not be able to match the true trajectory. It should be possible though, to analyse that error contribution separately similarly to a numerically introduced variance. See the last question for an ideas on how to do that.

Interrelations

The theoretical solutions and implementations from this work are employed throughout the work described in Part B. They enabled us to use the continuous-time approach for representing attitude within complex calibration problems, which in turn enabled us to implement temporal extrinsic calibration with exceptional simplicity and genericity.

2.1.4. Paper IV

Hannes Sommer, Raghav Khanna, Igor Gilitschenski, Zachary Taylor, Roland Siegwart, and Juan Nieto, “A Low-Cost System for High-Rate, High-Accuracy Temporal Calibration for LIDARs and Cameras”. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

Context

Validating calibration accuracy is a general problem due to the lack of accurate external measurement options. For instance, it is impossible to tell where the optical frame of non-pinhole camera is physically located. In fact, it is more an artifact of the interface between intrinsic and extrinsic camera calibration. Similar problems exist with other sensors and actuators.

However, for temporal calibration, in some circumstances, there is an opportunity to rely on the physical measurement process of active sensors, such as LIDARs, or to manipulate the environment quickly and with well known latency in order to assign verifiable timestamps to the sensors detections of the changes in the environment. For example, one can generate illumination changes for cameras or mechanical shocks for accelerometers. If the opportunity can be exploited at a high rate, ideally once per sensor timestamp, and at a high accuracy, then it allows to not only retrieve the mean latency — typically an important calibration parameter — but also the actual random delays in the sensor to processor communication as well as their correlation to other observable parameters, such as system load or temperature. This allows to develop noise models and specialized timestamp translators that can incorporate additional observables for a given sensor, or simply decide whether to use the timestamps of the sensor’s clock or to ignore them and rely on receive times only. Timestamp translators are required for operating most current sensors when timing is critical – due to the unfortunate scarcity of support for clock synchronization.

Contribution

We developed novel methods to detect the measurement time of time-of-flight LIDARs, using photodiodes, and to locate camera images of a specialized LED array accurately in the time of the LED controller. Additionally, we proposed an overall low cost solution to realize this methods in situ for existing robotic systems, involving simple electrical devices specialized for the task. The results of the experiments we conducted on five different LIDARs and two cameras with our prototypical devices clearly confirmed the usefulness of the entire setup for the intended use cases. Additionally, we discovered very surprising peculiarities of the analyzed LIDARs’ timestamps. These specific peculiarities are not very valuable knowledge in themselves because they typically do not generalize to other sensors but the possibility to detect them easily per sensor is of great value since they can diminish the performance of a system and are very hard to find otherwise.

Interrelations

This work produced valuable tools to pin down some of the many unknowns when doing calibration of multi sensor systems. They clearly allow us to answer some of the questions that came up while we were applying our self-calibration to systems as described in the application part, Section 2.2. Most importantly these were:

- *How to explain poor repeatability of the temporal calibration across different occasions?* When calibrating the EUROPA2 platform we discovered by overlaying LIDAR points on camera images of dynamic scenes that what was a good temporal extrinsic calibration on one occasion was actually poor on another occasion. Potential sources for the problem were the assignment of timestamps inside the sensors and our understanding of it, the performance of the sensors' clocks, and the necessary translation of the sensor timestamps into the system's time. They all must be good enough to allow for a useful temporal calibration. The lack of tools to validate any of these aspects was the very reason for this research and publication. We developed tools which allowed us to analyse the sensor clocks and the timestamping of the LIDARs and cameras and found a weakness in the timestamp translation that we employed: The convex hull based translation was reducing the jitter very well but could sometimes drift significantly off over a longer period – effectively altering the required temporal extrinsic calibration. This can be mitigated by periodically switching over to a fresher translator that has only seen the more recent timestamps.

- *How to validate temporal calibration?* Validating the accuracy of temporal calibration is generally hard due to the lack of more accurate and reliable methods to retrieve the exact same quantities. With the tools developed here we can retrieve temporal extrinsic calibration between some (important) sensor pairs in a very immediate way to validate the calibration method in question. Also, their respective validations concepts allowed us to quantify their accuracy with high certainty.

- *How long does it take for the sensor clocks to become steady and reliable e.g. during warm up?* Since our tools also allow to continuously estimate the measurement timestamps we can use them to assess how features of the timestamps develop over time and in relation with other observables. In particular, we can observe correlations with the uptime of a sensor or its temperature. Unfortunately, we did not have the resources to follow up on this idea in practice. But at least, we now have the tools to do it – for some sensors.

- *Can we trust / improve on the manufacturer specification regarding timestamping semantics and our understanding of them?* Typically, the manufacturer specification is unclear or incomplete. In PaperIV we analyzed many sensors in this regard including the LIDARs and camera of the EUROPA2 platform and found surprising and unspecified behavior in about half of them. Being able to quantify the temporal errors introduced by these deviations we could easily decide which needed consideration in the system. Among them, only the undocumented and very unexpected assignment of start timestamps inside the LMS151 was big enough in effect ($\sim$ 20ms) to justify special treatment for the EUROPA2 application.

- *Which is the best way to retrieve measurement timestamps for a given sensor? Translate hardware timestamps (with which algorithm) or rather rely on receive timestamps?* For most sensors and connection technologies (USB, Ethernet, ..) receive timestamps have too much randomness for many applications. Therefore, the device timestamps sent alongside the measurements and not corrupted by transport or processing delays are typically relied upon. However, this often requires translating the device timestamps into system time which always comes with some design choices, additional risks, and extra complexity. Hence, it is important to know about the quality of the device timestamps compared to receive times and about how the design choices affect the accuracy. We show how to retrieve the necessary data to make an informed decisions regarding these questions. Regarding EUROPA2, we switched to receive times for its nose, the Hokuyo UTM-30LX, after finding that its hardware timestamps were inferior in quality.

2.2. Applications

2.2.1. Hand-eye calibration

Introduction

One famous and important calibration problem is the spatiotemporal extrinsic calibration of a rigidly connected pair of pose sensors, i.e., sensors that can observe changes in position and attitude, called *hand-eye calibration*. In principle, two sufficiently dissimilar, abrupt, and jointly observed displacements are enough to globally retrieve the spatiotemporal extrinsic calibration with accuracies bounded from below by the individual spatial accuracies and measurement rates respectively. However, the need for higher accuracy asks for fusing the observation of many or continuous displacements. The traditional approaches collect many displacement pairs per sensor, align the timestamps with a suitable preprocessing, compute the spatial extrinsic calibration for each corresponding pair of observed displacement pairs, and finally employ sample statistics to retrieve a more accurate estimate of the calibration.

The recent [17] proposes new and efficient versions of the traditional approach for hand-eye calibration by prefiltering intelligently the candidates of pose displacement pairs and evaluates the gain in performance. As a co-author, I contributed a post optimization step, *the refinement step*, realizing the idea of fusing all measurements to improve the calibration accuracy.

The non-linear refinement step

The paper solves the global spatiotemporal extrinsic hand-eye calibration problem in three steps. In the first step, the two measurement series (from hand and eye) are aligned in time by correlating the angular velocity norms of both (relative) pose signals. In the second, they are aligned in space using a majority voting scheme for individual spatial analytic solutions to the classical hand-eye calibration problem. Together, this solves the global spatiotemporal extrinsic hand-eye calibration problem. However, the expected and experienced accuracy is limited mostly due to the facts that the errors of individual measurements are not modeled and that much of the data is rejected to find a consistent solution. A joint maximum likelihood optimization of calibration and trajectory given the measurements allows higher accuracy. This optimization is hard to solve as global problem but using the results from the first part as an initial guess a local likelihood maximization can improve the accuracy of the calibration. We perform this joint batch estimation as a third step, the *refinement step*, with a continuous-time representation for the trajectory, as described in 1.3 – with a single batch and the global approach from the paper's first part for *initialization*. More specifically, we model the joint problem with one trajectory

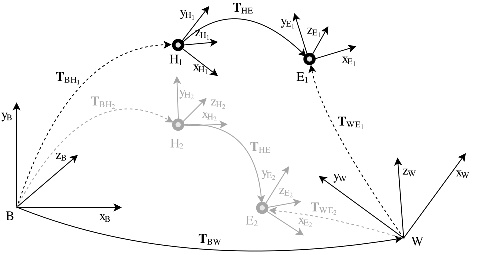


Figure 2.1.: The transformations relevant for the hand-eye calibration. Black are the transformations of the first pose pair and in grey the second pose pairs. Solid lines indicate static transformations, where dashed lines indicate transformations that change over time.

for the moving hand frame, H, $\mathbf{T}_{BH}(t) =: \mathbf{X}(t)$. The eye-frame, E, is assumed to be rigidly connected to the hand frame by the spatial calibration, $\mathbf{T}_{HE}$, as depicted in Figure 2.1. The pose measurement timestamps for H and E are assumed to be connected through a fixed time-offset Δt. Their errors we assume to be generated from isotropic multivariate Cauchy distributions[3] with three degrees of freedom independently for both translation (with zero mean) and rotation (with identity mean)[4] with respect to body-, B and world-frame, W, respectively. Or, if the eye is only emitting relative pose estimates (as, e.g., in visual inertial odometry), the same type of error source is assumed but with respect to the pose of the last measurement event. This yields the following negative log likelihood function, l, which we minimize:

$$
\begin{aligned}
&l\left(\mathbf{T}_{WE}, \mathbf{T}_{HE}, \Delta t, \mathbf{X} \,\middle|\, (\mathbf{T}_{WE\,i}, t_{E\,i})_{i=1}^{k}, (\mathbf{T}_{BH\,i}, t_{H\,i})_{i=1}^{l}\right) \\
&= \sum_{i=2}^{k_E} \rho(\|d_E\,(\mathbf{T}_{WE\,i-1}, \mathbf{T}_{WE\,i}, \mathbf{T}_{WE}(t_{E\,i-1}), \mathbf{T}_{WE}(t_{E\,i}))\,\|_{\Sigma_E}^2) \\
&+ \sum_{i=2}^{k_H} \rho(\|d_H\,(\mathbf{T}_{BH\,i-1}, \mathbf{T}_{BH\,i}, \mathbf{T}_{BH}(t_{H\,i-1}), \mathbf{T}_{BH}(t_{H\,i}))\,\|_{\Sigma_H}^2),
\end{aligned} \tag{2.1}
$$

where $(\mathbf{T}_{WE\,i}, t_{E\,i})_{i=1}^{k}, (\mathbf{T}_{BH\,i}, t_{H\,i})_{i=1}^{l}$ denote the measurement sequences, $\mathbf{T}_{BH}(t) := \mathbf{X}(t)$, $\mathbf{T}_{WE}(t) := \mathbf{T}_{WB}\mathbf{T}_{BH}(t - \Delta t)\mathbf{T}_{HE}$, $\rho(s) = log(1+s)$ the Cauchy-loss, and d_E and d_H are either relative, $(\mathbf{A}', \mathbf{A}, \mathbf{B}', \mathbf{B}) \mapsto d(\mathbf{A}'^{-1}\mathbf{A}, \mathbf{B}'^{-1}\mathbf{B})$, or absolute $(\mathbf{A}', \mathbf{A}, \mathbf{B}', \mathbf{B}) \mapsto d(\mathbf{A}, \mathbf{B})$ [5]. As displacement vector $d(\mathbf{A}, \mathbf{B}) \in \mathbb{R}^6$ on SE(3) we use coordinates of $(\log_{SO(3)}(\mathbf{R}), \mathbf{u})$ with respect to a fixed positive orthonormal basis, where $\mathbf{u}$ is a translation and $\mathbf{R}$ a proper rotation such that (uniquely) $\mathbf{u} \circ \mathbf{R} := \mathbf{A}^{-1}\mathbf{B}$. Please note that l becomes independent of $\mathbf{T}_{WB}$ iff d_E is relative because then it

---

[3] This is equivalent to least squares with a Cauchy loss function.

[4] Approximated with zero-mean Cauchy distributions in the Lie algebra projected to SO(3) using the exponential map.

[5] For absolute d_E, d_H the corresponding first measurements, $i = 1$, are assumed to be dummy variables while the real measurements start with $i = 2$.

cancels out in $\mathbf{B}'^{-1}\mathbf{B}$. If not, it must be co-estimated.

Results concerning the refinement step

Evaluating any calibration accuracy is difficult due to the usual lack of ground truth information. However, for hand-eye calibration there is a nice trick to get just that for a very special arrangement of sensors: using three pose sensors rigidly connected the hand-eye calibration can be performed on all three unordered pairs and the resulting three Euclidean transformations should compose to identity because of the circular topology of the three pairs. The deviation from this ideal result, the *circular calibration error*, is therefore a direct measure of the methods accuracy over the three problem instances.

We conducted an evaluation using two datasets with three Google Tango tablets [61] that are rigidly mounted on an aluminum profile. Sample statistics of the circular error is plotted in Figure 2.2 for both datasets. The refinement step is labeled with O. All other labels refer to the competing approaches for the first, global step. For their details see Section 3.3.1 of [17]. The samples for these approaches are 20 runs of the randomized algorithms. The samples for the deterministic refinement step are the various initializations generated by the different approaches and all their runs. After the refinement step we get a mean circular position and orientation error of 4.02 mm and 0.091° for the *Tango 1* dataset, and 7.19 mm and 0.139° for the *Tango 2* dataset. Please note, that this is the accumulated error over all three device pairs.

The very small spread of the results after the refinement step indicates for those experiments that the approach is not very sensitive to the quality of its initialization and that even the simplest and fastest approach among the candidates is good enough. The significantly smaller mean error suggest that adding the refinement step is indeed a valuable addition.

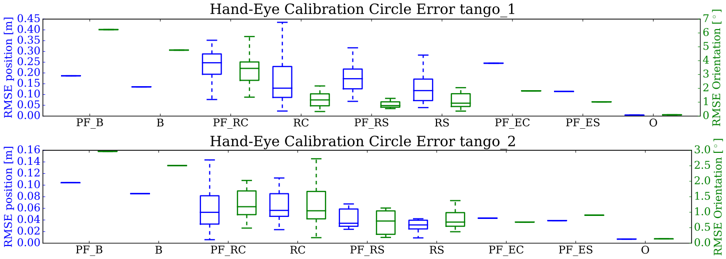


Figure 2.2.: The circular error of the individual methods and the combined results after applying the refinement step to the other methods, denoted with O.

Implementation

This paper uses the calibration library, OOMACT, developed alongside this thesis and building on the contributions of the papers PaperI, and PaperIII, to perform the refinement step. To solve the non-linear optimization we use an extension of Levenberg-Marquardt to Lie groups (as introduced e.g. in PaperIII).

Thanks to the generic and modular design of the library this application requires about 10 well readable lines of c++ code to setup and solve this calibration problem:

```
const auto config = loadConfigFile();

FrameGraphModel model(config.getChild("model"));
PoseSensor pose1_sensor(model, "pose1");
PoseSensor pose2_sensor(model, "pose2");
PoseTrajectory traj(model, "traj");
model.addModulesAndInit(pose1_sensor, pose2_sensor, traj);

const auto calib = createBatchCalibrator(
            config.getChild("calibrator"), model);

readPosesFromCsv(FLAGS_pose1_csv, pose1_sensor, *calib);
readPosesFromCsv(FLAGS_pose2_csv, pose2_sensor, *calib);

calib->calibrate();

writeCalibrationResult(FLAGS_output_file, model);
```

Here, the only functions not provided by OOMACT and its support libraries are `loadConfigFile`, `readPosesFromCsv`, and `writeCalibrationResult`, which purely adapt to the specific input and output data format and location. All the remaining specifics for this calibration problem are provided through the configuration.

2.2.2. Report V

Hannes Sommer, Zachary Taylor, "Life-Long Self-Calibration and Fault Detection". In *EU 7th framework programme deliverables*, 2016.

Context

The EUROPA platform is almost the perfect environment for developing a high quality generic calibration system because it has rather high computational resources and many kinds of sensors crucial for modern mobile robots. They are one 3d and five 2d LIDARs, one actuated, two stereo camera systems, wheel odometry, and a high quality IMU.

Contribution

The report presents the details of our approach for and implementation of the life-long self-calibration system that we developed for the EUROPA2 platform estimating the spatiotemporal extrinsic calibration of all its sensors and actuators (wheels locomotion system + LIDAR tilting mechanism) and intrinsics of the wheel odometry. In particular, it presents a novel approach to perform LIDAR-LIDAR (where up to one may be a 2d-LIDAR) and LIDAR-stereo-camera extrinsic self-calibration. It also contains an extensive, experimental evaluation of the system for offline and online calibration in in-laboratory conditions.

Interrelations

The calibration system presented in the report is the primary application of the prototypical calibration libraries developed in parallel to this thesis. It provided the demand and the opportunity for developing and testing these tools. It builds on the contributions from PaperI, PaperIII, and [4]. and it provided the motivation and opportunity for PaperII and PaperIV.

2.3. List of Publications

Until the end of the author's doctoral studies the following articles were published. They are grouped by type and sorted chronologically.

2.3.1. Journals

[1] Marco Hutter, Hannes **Sommer**, Christian Gehring, Mark Hoepflinger, Michael Bloesch, and Roland Siegwart. Quadrupedal locomotion using hierarchical operational space control. *The International Journal of Robotics Research*, 33(8):1047–1062, 2014.

[2] Hannes **Sommer**, James Richard Forbes, Roland Siegwart, and Paul Furgale. Continuous-time estimation of attitude using b-splines on lie groups. *Journal of Guidance, Control, and Dynamics*, 39(2):242–261, 2015.

[3] Jemin Hwangbo, Christian Gehring, Hannes **Sommer**, Roland Siegwart, and Jonas Buchli. Policy learning with an efficient black-box optimization algorithm. *International Journal of Humanoid Robotics*, 12(03):1550029, 2015.

[4] Jérôme Maye, Hannes **Sommer**, Gabriel Agamennoni, Roland Siegwart, and Paul Furgale. Online self-calibration for robotic systems. *The International Journal of Robotics Research*, 35(4):357–380, 2016.

[5] Michael Bloesch, Michael Burri, Hannes **Sommer**, Roland Siegwart, and Marco Hutter. The two-state implicit filter recursive estimation for mobile robots. *IEEE Robotics and Automation Letters*, 3(1):573–580, 2018.

[6] Renaud Dubé, Mattia G Gollub, Hannes **Sommer**, Igor Gilitschenski, Roland Siegwart, Cesar Cadena, and Juan Nieto. Incremental-segment-based localization in 3-d point clouds. *IEEE Robotics and Automation Letters*, 3(3):1832–1839, 2018.

[7] Hannes **Sommer**, Igor Gilitschenski, Michael Bloesch, Stephan Weiss, Roland Siegwart, and Juan Nieto. Why and how to avoid the flipped quaternion multiplication. *Aerospace*, 5(3), 2018. ISSN 2226-4310. doi:10.3390/aerospace5030072. URL `http://www.mdpi.com/2226-4310/5/3/72`.

2.3.2. Conferences

[8] A. Breitenmoser, H. **Sommer**, and R. Siegwart. Adaptive Multi-Robot Coverage of Curved Surfaces. In *Proc. of the International Symposium on Distributed Autonomous Robotic Systems (DARS)*, Nov. 2012.

[9] Michael Bloesch, Sammy Omari, Péter Fankhauser, Hannes **Sommer**, Christian Gehring, Jemin Hwangbo, Mark A Hoepflinger, Marco Hutter, and Roland Siegwart. Fusion of optical flow and inertial measurements for robust egomotion estimation. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 3102–3107. IEEE, 2014.

[10] Jemin Hwangbo, Christian Gehring, Hannes **Sommer**, Roland Siegwart, and Jonas Buchli. Rock*-efficient black-box optimization for policy learning. In *Humanoid Robots (Humanoids), 2014 14th IEEE-RAS International Conference on*, pages 535–540. IEEE, 2014.

[11] Mark Pfeiffer, Ulrich Schwesinger, Hannes **Sommer**, Enric Galceran, and Roland Siegwart. Predicting actions to act predictably: Cooperative partial motion planning with maximum entropy models. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 2096–2101. IEEE, 2016.

[12] Hannes **Sommer**, Cédric Pradalier, and Paul Furgale. Automatic differentiation on differentiable manifolds as a tool for robotics. In *Robotics Research*, pages 505–520. Springer, 2016.

[13] Renaud Dubé, Hannes **Sommer**, Abel Gawel, Michael Bosse, and Roland Siegwart. Non-uniform sampling strategies for continuous correction based trajectory estimation. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 4792–4798. IEEE, 2016.

[14] Hannes **Sommer**, Raghav Khanna, Igor Gilitschenski, Zachary Taylor, Roland Siegwart, and Juan Nieto. A low-cost system for high-rate, high-accuracy temporal calibration for lidars and cameras. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pages 2219–2226. IEEE, 2017.

[15] Renaud Dubé, Abel Gawel, Hannes **Sommer**, Juan Nieto, Roland Siegwart, and Cesar Cadena. An online multi-robot slam system for 3d lidars. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pages 1004–1011. IEEE, 2017.

[16] Mattia G Gollub, Renaud Dubé, Hannes **Sommer**, Igor Gilitschenski, and Roland Siegwart. A partitioned approach for efficient graph–based place recognition. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst./Workshop Planning, Perception Navigat. Intell. Veh*, 2017.

[17] Fadri Furrer, Marius Fehr, Tonci Novkovic, Hannes **Sommer**, Igor Gilitschenski, and Roland Siegwart. Evaluation of combined time-offset estimation and hand-eye calibration on robotic datasets. In *Field and Service Robotics*, pages 145–159. Springer, 2018.

2.3.3. Technical report

[18] Michael Bloesch, Hannes **Sommer**, Tristan Laidlow, Michael Burri, Gabriel Nuetzi, Péter Fankhauser, Dario Bellicoso, Christian Gehring, Stefan Leutenegger, Marco Hutter, et al. A primer on the differential calculus of 3d orientations. *arXiv preprint arXiv:1606.05285*, 2016.

Chapter 3

Conclusion and Outlook

This concluding chapter provides a brief summary of the most important findings and suggest some potential directions for future research.

The dream of easy lifelong self-calibration is a big dream and we are still quit far away from it — despite the fact that the required statistical knowledge and problem understanding has been available for a long time already. This work contributes to some of the more fundamental practical problems which make this dream hard to achieve in practice:

- *Quaternion multiplication:* Educating about the background of and problems with the flipped quaternion multiplication once introduced by NASA's JPL and providing an alternative solution to the original problem will hopefully help with future decisions for a quaternion convention and help preventing some expensive and painful confusion when dealing with rotation quaternions. One of the two main authors of the flipped quaternion multiplication, Dr. Landis Markley, agreed mostly with our critique even publicly, and gave his valuable support to publishing our work. Some libraries and paper already refer to our argumentation when explaining their choice of quaternion convention. The latter was precisely our aim and hope.

- *The block automatic differentiation (BAD)* has great potential to save development time and increase software flexibility and correctness. Unfortunately our fast prototype could not (yet) mature into something that can be used practically. However, our publication inspired at least two great developments building on the concept: the GTSAM expressions – an important new feature of the recent version 4.0 –, which we helped to implement, and [91] which was strongly influenced by our work and even exceeds the performance of our fast prototype in some aspects, while lacking some of the flexibility oriented features.

- *The Lie group valued B-splines* enables the great simplification for temporal calibration through adequate continuous-time state representation also for attitude without sacrificing the important rotational bi-equivariance. The paper demonstrated the superior estimation performance of this approach compared to state of the art EKF and first order batch estimation for some example cases of orbital space-craft attitude estimation. It is at the functional heart of the calibration library we developed. There it excels in representing attitude trajectories – in particular when delays must be estimated. The open source, prototypical implementation is to the best of our knowledge still the only available library providing hand optimized Jacobians computations (up to second temporal derivative) for arbitrary order unit-length quaternion B-splines.

- *Our novel method regarding sensor timestamping analysis* provides the means to inspect the timestamping of LIDARs and cameras in a high rate and high accuracy fashion. This supports designing, validation, and debugging of timestamp translation, clock synchronisation, and temporal calibration. It was very valuable in finding the roots of the difficulties we experienced with sensor delay estimation and revealed various interesting unspecified behaviors of several LIDAR sensors' timestamp generation.

- *The evaluation of self-calibration within the EUROPA2 project* demonstrates the applicability of the overall concept and the developed algorithms and software libraries. Furthermore, it introduces a novel concept to do self-calibration for 3d point cloud sensors.

Together these contributions make developing calibration software for a complex multi-sensor system such as the EUROPA2 platform significantly easier to realize. Some further building-block developed and implemented for this task were never separately evaluated and published due to the lack of resources and time, such as the extremely useful automatic kinematic chain library (prototypical code available as open source), and the novel point cloud registration concept for calibration described in the ReportV (code not (yet) available). Nevertheless, some of the useful blocks were published and hopefully aid future work in this domain and some of these have been picked up by others already.

In fact, there is a lot of future work to do on the paths we started on:

- Thanks to BAD, the batch oriented approach, and Lie-group valued B-splines it would be easy to represent the temporal uncertainty of each individual measurement with a scalar variable plus a factor to model the uncertainty distribution for each measurement and study the effect on the calibration quality.

- The Lie-group valued B-splines we used are based on the Lie-group exponential and therefore build on one-parameter subgroups as primitives (which at the same time resemble a concept of geometric straightness if there is a compatible Riemannian geometric structure). However, typically this structure has little to do with physical laws which should ideally determine the shape of trajectory candidates. As a consequence, it could be promising (but computational expensive) to directly use the Lie-group structure induced by the unperturbed (deterministic) dynamic system as action space on the physical configuration instead. In particular, if the actual process model noise is small and the configuration space is rather simple, this should improve trajectory estimates significantly even with low knot rate.

- For some systems vibrations while moving are a serious problem for extrinsic calibration since they are hard to observe or model accurately. It could be worth to try giving the poses at a measurement instance some pose-slag w.r.t. the trajectory with some uncertainty of its deviation from zero.

- Using a compiler to perform mathematical type checking as done in our BAD concept for manifolds can and should be extended to physical modeling such as by checking the correct choice of reference frames and units. The latter ideas are not new but they are particularly valuable in a model based calibration framework and still lacking in this domain to the best of our knowledge.

- In order to support debugging the models our calibration framework is keeping track of where all the factors originate from and which calibration variables they depend on — all with human readable labels — to allow an extremely useful group-wise error statistics per iteration. This concept should be integrated into general purpose optimizers (for research and development) so that they can provide the developers with semantic labels and statistics based on such labels. This should be helpful in any domain using optimization with complex models.

- Adding noise parameters, such as sensor measurement variance, to the optimizable variables should be fairly straight forward in our design and could contribute greatly to the practical problem of finding realistic noise parameters as well as to finding problematic sensors in a given system.

Part A

CONCEPTUAL CONTRIBUTIONS

Paper

Automatic Differentiation on Differentiable Manifolds as a Tool for Robotics

Hannes Sommer, Cédric Pradalier, and Paul Furgale

Abstract

Automatic differentiation (AD) is an useful tool for computing Jacobians of functions needed in estimation and control algorithms. However, for many interesting problems in robotics, state variables live on a differentiable manifold. The most common example are robot orientations that are elements of the Lie group SO(3). This causes problems for AD algorithms that only consider differentiation at the scalar level. Jacobians produced by scalar AD are correct, but scalar-focused methods are unable to apply simplifications based on the structure of the specific manifold. In this paper we extend the theory of AD to encompass handling of differentiable manifolds and provide a C++ library that exploits strong typing and expression templates for fast, easy-to-use Jacobian evaluation. This method has a number of benefits over scalar AD. First, it allows the exploitation of algebraic simplifications that make Jacobian evaluations more efficient than their scalar counterparts. Second, strong typing reduces the likelihood of programming errors arising from misinterpretation that are possible when using simple arrays of scalars. To the best of our knowledge, this is the first work to consider the structure of differentiable manifolds directly in AD.

Published in:
Robotics Research. Springer Tracts in Advanced Robotics, 2016
doi:10.1007/978-3-319-28872-7_29

1. Introduction

Computation of the Jacobian matrix of a nonlinear function is an essential part of many estimation and control algorithms and, as such, it is a ubiquitous task within robotics and computer vision. When faced with the task of implementing Jacobian computation within a computer program, there are essentially four choices[1]. First, one can hand-compute the analytical expression. This is easy for simple functions, careful attention may be paid to the correct handling of singularities in the operations, and the computations may be hand-optimized for speed. However, it may become arduous to compute and verify Jacobians every time a small change is needed. Second, one may approximate the derivatives numerically. This is easy to implement but the resulting Jacobians may be inaccurate for highly nonlinear functions and finite differencing schemes can fail completely when the functions include conditional statements. Third, one may use symbolic differentiation tools to generate code from the nonlinear functions. The resulting Jacobians are accurate and highly efficient to evaluate but this method involves a pre-processing step and there is no guarantee that the automatically generated code correctly handles singularities. Finally, one may use Automatic Differentiation (AD) to compute the Jacobian matrices algorithmically. AD algorithms compute derivatives by exploiting the deterministic nature of derivative computation. The derivatives of atomic operations are implemented by the AD toolkit. The derivatives of more complex functions are constructed by applying the chain rule at each operation and bookkeeping the results. The application of AD is extremely easy using one of the many tools available[2]. Jacobians computed by this method are as accurate as their hand-coded counterparts but they are less efficient to compute and, again, may not handle all singularities.

In terms of accuracy and evaluation speed, hand coding or code generation are the clear choices. However, they share the common drawback that any change in the original function requires the Jacobian computation to be updated in lockstep. When prototyping, AD alleviates this requirement as changes in the nonlinear function are automatically reflected in the Jacobian computation. In our experience, researchers live continually in the prototyping phase and so, for problems in robotics focusing on estimation and control, AD is a tool to accelerate research.

However, for many interesting problems in robotics, state variables live on a Differentiable Manifold (DM). In robotics the most important DMs are the proper rotation and Euclidean groups in two- and three-dimensional Euclidean space ({SO(2), SO(3) } and {SE(2), SE(3) } respectively) for rigid-body kinematics, as well as the two-dimensional sphere, S^2, (e.g. for bearing vectors in sensor models [93]). Unit-length quaternions (elements of $S^0(\mathbb{H})$) are another popular choice for representing orientations.

The handling of state variables on DMs within estimation and control has been the subject of active research for many years in robotics and aerospace [19, 27, 65, 73, 146], but this analysis has not made it into the AD literature. The theory and implementation of AD is decidedly focused on the derivatives of *scalar operations* as the computational atoms [133]. It is possible to coax AD packages to compute the correct Jacobians for functions that operate on elements of a DM, but this involves lifting the derivative computations to the outer space. Working in the

---

[1] This paper will focus on the scale of problem generally encountered in estimation and control algorithms in robotics and not deal with methods used for larger-scale problems such as finding the Jacobian of a fluid simulation with respect to airfoil parameters (c.f. [109]).

[2] See `http://www.autodiff.org/` for an extensive list of AD tool-kits spanning many popular programming languages.

outer space precludes the possibility to utilize the special structure of the manifold to simplify derivative computation.

In this paper we extend the theory of AD to DMs by considering block operations as the atoms of computation, and exploiting the specific structure of the manifold to increase the efficiency of computation. Throughout this paper we will use expressions involving unit-length quaternions as our main example but the method is applicable to any DM. A pictorial representation of our contribution is presented in Figure 4.1. This figure compares scalar AD with our approach by plotting the computation graph for a simple example in which an unit-length quaternion, q, is used to rotate a point, $\mathbf{v}$.

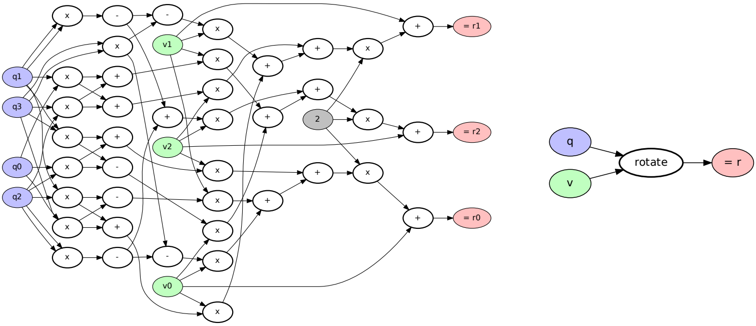


Figure 4.1.: A simple example in which an unit-length quaternion, q, is used to rotate a point, $\mathbf{v}$. Left: The computational graph associated with the calculation from the point of view of standard scalar-focused AD algorithms. Right: The computational graph from the point of view of our method. Although the evaluation of these two graphs is the same, computing the Jacobian from the left graph with a dual-number approach requires **168 multiplication and 156 additions**, whereas only **3 multiplications** are required by taking advantage of the DM structure on the right (see section 2.3 for details). Note, that because our framework knows about the underlying DM, it effectively only need to differentiate this very simple graph on the right, when the normal AD needs to differentiate the large complex graph on the left because it works at the scalar level.

We see the contributions of this work to be the following:

- We extend the theory of AD to operate on DMs. Because the atoms of computation become block operations, we call this Block Automatic Differentiation (BAD). To the best of our knowledge, this is the first work to make this extension.

- We present a prototype C++ implementation of BAD with a number of desirable properties. First, it is faster than AD that does not explicitly consider differentiation with respect to the underlying manifold. Second, rather that simply overloading operators on scalar or matrix types (a standard method of developing AD), we develop a type-safe system where every value belongs to a specific mathematical type with a well-defined tangent space for differentiation. This makes the system easier to use than standard scalar AD and guards against coding errors possible from the accidental misinterpretation of arrays of scalar types.

- We perform a rigorous comparison of our method against the dual-number approach to AD implemented in the Google Ceres optimization package [20].

The rest of the paper is organized as follows. Section 2 reviews the background theory and presents the basis for our BAD algorithm. Section 3 describes our prototype implementation of BAD as a C++ library. Our experimental results are presented in Section 4 and we conclude in Section 5.

2. Theory

This section reviews the relevant theory on AD and DMs, and then describes our extension to bring the two concepts together.

2.1. Automatic Differentiation

An excellent introduction to Automatic (or Algorithmic) Differentiation is presented in [133] and [132]. This section attempts to summarize the content therein to bring the minimum of context on the topic. The interested reader is referred to these documents and the references therein for further details.

In the realm of computer science, AD is an ancient field, first developed to compute derivatives on specialized computers in the late 1960s. Widespread application of the technique came with the development of numeric tools in Fortran in the last two decades of the 20^{th} century.

The basic idea of AD is easy to explain by imagining the evaluation of mathematical expressions represented by a *computation graph*. When interpreted by a compiler, a numeric expression is represented as a computation tree, with constants or variables as leaves and operators (‘$+$’,‘$-$’,...) and functions (cos, sin, exp, ...) as interior nodes. Because of sub-expressions being reused or compiler optimization, this tree becomes a computation graph (more precisely, a directed acyclic graph). Figure 4.1 depict such computation graph for rotating a 3D point with a quaternion.

The foundation of AD stems from the fact that the rules of differentiation are deterministic and they can be applied mechanically and recursively to the computation graph. Rather than deriving a formula for the derivative of an expression, the derivative is computed algorithmically by traversing the graph and using a combination of the chain rule and known operator differentiation rules. Traversing the graph from the leaves to the root is known as ‘forward’ AD and traversing it from the root to the leaves is known as ‘backward’ AD. Backward AD is more complex to implement and requires more storage but typically requires fewer operations, whereas forward AD can be implemented in a straightforward manner with operator overloading and/or dual numbers. In the latter case, the AD system evaluate the expression of interest, initially designed for numeric values, on a specific data structure grouping the expression value and its derivative(s)

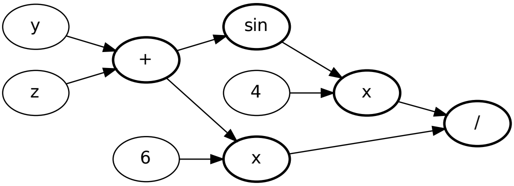


Figure 4.2.: Computation graph corresponding to expression $\frac{6(y+z)}{4\sin(y+z)}$. Nodes circled in bold line represent operators or functions.

at each node of the graph. A recent example of such dual-numbers is the *Jet* class used by Google Ceres[20] to implement its AD feature.

In traditional languages not supporting operator overloading (e.g. Fortran 77), AD has been implemented by a pre-processor stage that would parse an expression in the original language and generate code to compute the expression and its derivative(s), to be compiled and linked in the final program.

Our contribution lies in the fact that adding knowledge about the underlying DM, we can dramatically simplify the computational graph (e.g. Figure 4.1). By considering differentiation and the chain rule at the block level, we are able to exploit specific problem structure and increase computational efficiency. We compare our approach to the "dual-number" AD [127] implemented in Google Ceres[20].

The basic idea of the dual-number concept to do AD for scalar functions is to calculate the derivatives value alongside the functions values through the computation graph from the leaves to the root. This is done by applying all scalar operations to pairs of value v and derivative's value d $\langle v, d\rangle$. The derivative's value starts with 1, while the values start with the value the function is evaluated at. The scalar operations apply normally on the value parts, but a special corresponding operation to the derivative. For example $\langle v_1, d_1\rangle * \langle v_2, d_2\rangle := \langle v_1 v_2, d_1 v_2 + d_2 v_1\rangle$. To calculate gradients of functions in multiple scalar variables one simply keeps one second number for each variable and initializes each variable with the corresponding second number as 1 and the others as zero. For example $a * b$ one would then calculate the value and gradient at $a = 2, b = 3$ as follows : $\langle 2,1,0\rangle + \langle 3,0,1\rangle = \langle 2*3, 1*3+0*2, 0*3+1*2\rangle = \langle 6,3,2\rangle$. The gradient can then be extracted as the row vector of all the resulting second numbers $(3,2)$.

2.2. Differentiable Manifolds and Jacobians

The concepts of differential geometry have been adopted by the robotics community as the mathematical underpinnings of kinematics and dynamics. An excellent introduction is available in Murray et al. [120] but we will provide a brief overview of the concepts necessary to understand the idea of BAD.

For the scope of this paper a m-dimensional DM (with $m \in \mathbb{N}$) is a set, $\mathscr{M}$, together with an *m-atlas*, $A_{\mathscr{M}}$, inducing a second-countable Hausdorff topology on $\mathscr{M}$. To precisely define the notion of an atlas is beyond the scope of this paper, but informally $A_{\mathscr{M}}$ is a collection of *charts*, where each chart, $\varphi : U_\varphi \to \mathscr{M}$, is an invertible mapping defined on an open subset U_φ of $\mathbb{R}^m$ onto a subset of $\mathscr{M}$. The atlas provides a *differentiable structure* to the manifold $\mathscr{M}$. Using these charts, we can do differential calculus for functions between DMs, which include all finite dimensional vector spaces (by assigning DMs that reproduce the usual calculus on vector spaces).

Defining a Local Jacobian

Consider a mapping $f : \mathscr{M} \to \mathscr{N}$ between the m-dimensional DM $\mathscr{M}$ and the n-dimensional DM $\mathscr{N}$. At a point $p \in \mathscr{M}$, for f there exist a notion of differentiability and a formal local linearization, the *differential*, denoted with $\mathrm{d}_p f$. Its rigorous definition is also beyond the scope of this paper. Informally it plays the role of a derivative in a DM context. For algorithmic treatment, one requires a matrix representing this differential, but the usual Jacobian of nonlinear functions is only defined for mappings between vector spaces.

However, after choosing charts $\varphi_{\mathscr{M}}$ and $\varphi_{\mathscr{N}}$ around p and $f(p)$ respectively, one may define $\hat{\mathbf{f}} := \varphi_{\mathscr{N}}^{-1} \circ f \circ \varphi_{\mathscr{M}}$ at $\hat{\mathbf{p}} := \varphi_{\mathscr{M}}^{-1}(p)$, a mapping $\mathbb{R}^m \supset \mathrm{dom}(\varphi_{\mathscr{M}}) \to \mathbb{R}^n$, where $\mathrm{dom}(\varphi_{\mathscr{M}})$ denotes the domain that $\varphi_{\mathscr{M}}$ is defined on. We call f differentiable at p iff $\hat{\mathbf{f}}$ is differentiable at $\hat{\mathbf{p}}$ and define f's *Jacobian* in these charts with,

$$\mathrm{J}_p f := \mathrm{J}_{\hat{\mathbf{p}}} \hat{\mathbf{f}} = \mathrm{J}_{\varphi_{\mathscr{M}}^{-1}(p)} (\varphi_{\mathscr{N}}^{-1} \circ f \circ \varphi_{\mathscr{M}}). \tag{4.1}$$

These Jacobians can then play the same role in algorithms dealing with manifolds as the usual Jacobians do for nonlinear mappings between vector spaces. The algorithmic complexity to calculate them depends not only on f but also on the chosen charts. The latter feature is one of the underlying principles that BAD tries to exploit.

Defining a Global Jacobian

Usually manifolds are algorithmically represented as *embedded sub-manifold* of an *outer* $\mathbb{R}$-*vector space* $\mathscr{O}_{\mathscr{M}} := \mathbb{R}^{O_{\mathscr{M}}}$, with $O_{\mathscr{M}} > m$. This means informally that $\mathscr{M}$ is represented by a subset of $\mathscr{O}_{\mathscr{M}}$ for which the differentiable structure inherited from $\mathscr{O}_{\mathscr{M}}$ makes it a DM diffeomorphic to $\mathscr{M}$. Hence, points of $\mathscr{M}$ can easily be represented as the corresponding elements of $\mathscr{O}_{\mathscr{M}}$. A simple example is the Lie group of unit-length quaternions, a three-dimensional manifold that is often stored and manipulated as the S^3 sub-manifold of $\mathbb{R}^4$. In the following, we will assume that $\mathscr{M}$ and $\mathscr{N}$ are embedded sub-manifolds of $\mathbb{R}^{O_{\mathscr{M}}}$ and $\mathbb{R}^{O_{\mathscr{N}}}$ respectively.

This situation allows for a special way to acquire a Jacobian for f. One can choose a mapping $\tilde{\mathbf{f}} : V \to \mathscr{O}_{\mathscr{N}}$ defined on an open environment $V \subset \mathscr{O}_{\mathscr{M}}$ of p such that $f|_{\mathscr{M} \cap V} = \tilde{\mathbf{f}}|_{\mathscr{M} \cap V}$ and calculate the Jacobian $\mathrm{J}_p \tilde{\mathbf{f}} \in \mathbb{R}^{O_{\mathscr{N}} \times O_{\mathscr{M}}}$ of $\tilde{\mathbf{f}}$ instead. Here, $F|_A$ denotes the restriction of a mapping F on a set A. We will call $\mathrm{J}_p \tilde{\mathbf{f}}$ a *global Jacobian* of f. Note that it does not require a choice of charts but depends on the choice of $\tilde{\mathbf{f}}$ instead. For some applications this global Jacobian is already usable but for many it introduces instabilities and performance loss because it calculates a matrix that is bigger than necessary (recall that the *local* $\mathrm{J}_p f \in \mathbb{R}^{n\times m}$) introducing extra degrees of freedom. In those cases one can calculate $\mathrm{J}_p f$ in a second step after choosing charts from $\mathrm{J}_p\tilde{\mathbf{f}}$ by exploiting,

$$\begin{aligned} \varphi_{\mathscr{N}}^{-1} \circ f \circ \varphi_{\mathscr{M}}|_U &= \vartheta \circ \tilde{\mathbf{f}} \circ \varphi_{\mathscr{M}}|_U && (4.2) \\ \Longrightarrow \mathrm{J}_p f &= \mathrm{J}_{f(p)}\vartheta \cdot \mathrm{J}_p\tilde{\mathbf{f}} \cdot \mathrm{J}_{\hat{\mathbf{p}}}\varphi_{\mathscr{M}}, && (4.3) \end{aligned}$$

for an open set $U \subset \mathrm{dom}(\varphi_{\mathscr{M}}) \subset \mathbb{R}^m$ containing $\hat{\mathbf{p}}$, small enough, and a differentiable $\vartheta : \tilde{\mathbf{f}}(\varphi_{\mathscr{M}}(U)) \to \mathbb{R}^n$, such that $\vartheta|_{\mathrm{dom}(\varphi_{\mathscr{N}}^{-1})\cap\mathrm{dom}(\vartheta)} = \varphi_{\mathscr{N}}^{-1}|_{\mathrm{dom}(\varphi_{\mathscr{N}}^{-1})\cap\mathrm{dom}(\vartheta)}$.

Even though this introduces an unnecessary step in the algorithm, it is precisely the path suggested by a scalar-based AD. Because the manifolds are represented as embedded submanifolds and thus the algorithm evaluating f maps, in practice, elements of $\mathscr{O}_{\mathscr{M}}$ to elements of $\mathscr{O}_{\mathscr{N}}$. Hence, applying a scalar-based AD approach directly on this evaluating algorithm will calculate a global Jacobian, $\mathrm{J}_p\tilde{\mathbf{f}}$, typically concealing the fact that a choice of $\tilde{\mathbf{f}}$ does happen in this step.

The missing two Jacobians in (4.3) can either be calculated using scalar-based AD or by manually supplied algorithms. The latter is exactly the concept behind the current Ceres implementations when one uses its *local parametrization*, which plays here the role of the chosen chart $\varphi_{\mathscr{M}}$. As Ceres does not currently support manifold-valued error terms it can only be used in cases where $\mathscr{N}$ is a vector space and thus ϑ can be chosen trivial.

2.3. Block Automatic Differentiation

This section provides an overview of the BAD concept and a simple worked example showing the magnitude of speedup that is possible.

Overview

The motivation behind the development of BAD is to be able to inject knowledge from the specific structure of a DMs into the AD process in order to speed up the calculation of Jacobian matrices. The speedups gained by this approach will be specific to each manifold.

To enable an AD library to do that, it needs to know which DMs are involved in a mapping $f : \mathscr{M} \to \mathscr{N}$ that should be differentiated. To achieve that, the primary idea is to consider a computation graph on a mathematically higher level. Instead of primitive scalar operations $(+, *, \dots)$ on a single scalar type, we consider a set of basic operations that operate on points in manifolds. For example, this encompasses the usual vector and matrix operations, but also geometric operations like exponential maps, or special operations like rotation of a 3D point by an unit-length quaternion.

In practice most differentiable functions of interest between DMs can be deconstructed into a computation graph compounding such basic operations. These are the computational atoms that we think in when building up mathematical models and, in robotics, there is a surprisingly small set of such basic operations needed to implement many fundamental algorithms.

Such a high level computation graph can then be grouped by a BAD library into sub-blocks for which there is optimized (Jacobian) evaluation code. This optimized code can be either be manually written or the output of a code generator of a symbolic toolkit.

In this way, the BAD concept is a mixture of manual, symbolic, and automatic differentiation, allowing a series of novel opportunities to optimize the computational complexity:

- to derive a suitable intermediate value-cache structure for the evaluation of the compound function and Jacobian evaluation;
- to automatically apply mathematically simplified algorithms for whole compound functions;
- to choose, based on the expression, the Jacobian evaluation direction or even mixtures of forward and backward steps; and
- to use highly optimized matrix manipulation libraries to do the final calculations.

However, there is one important obvious drawback when compared to AD: there are many more combinations of manifolds and basic operations on them than scalar operations. While it is easily possible to make a scalar AD library complete, a BAD library will never be. Because of this it is very important for a BAD library to be user extensible and to grow over time, eventually reverting to scalar AD as a last resort. To address this, our implementation efforts have been focused on building up a core framework that tries to make it as easy as possible to add support for manifolds and operations.

Ultimately, a BAD library will be less optimal than the output of an ideal symbolic tool optimizing the whole function f. Nevertheless, current symbolic tools have many important drawbacks compared to the BAD concept:

- the work flow from the mathematical model to the running code involves extra steps (i.e. code generation during the compilation phase) that may take much more processing time, especially when the expressions get quite complex;
- they are bad in factoring out repeated blocks to functions and thus produce huge code that is impossible to read and hard to maintain;
- they don't allow dynamic (at run-time) construction of the function f, which can be essential for special problems; and
- they usually do not incorporate matrix manipulation libraries and thus neglect a very important intermediate level of optimization.

Example

Here we provide the simple example of the rotation of a 3×1 vector, $\mathbf{v}$, by a rotation, $\mathbf{C}$, represented by an unit-length quaternion, q, to illustrate the potential of mathematical simplification of the Jacobian evaluation by exploiting knowledge about the underlying structures. Let v denote the pure imaginary quaternion corresponding to $\mathbf{v}$. The function we will analyze can then be defined as,

$$\mathbf{r} : S^0(\mathbb{H}) \to \mathbb{R}^3,\ \ q \mapsto \operatorname{Im}(qv\bar{q}), \tag{4.4}$$

where multiplication of non-bold quantities is quaternion multiplication, the overbar denotes the quaternion conjugate operation, and $\mathrm{Im}(\cdot)$ extracts the imaginary components of the quaternion as a 3×1 vector. Here, $\mathbf{r}(\cdot)$ is only defined on the unit-length quaternions.

To be able to write an algorithm that evaluates this function or its Jacobian, we have to define how to represent the involved quantities. We will represent q with a 4-tuple (q_0, q_1, q_2, q_3) identified with q's coordinates in default ordered basis of $\mathbb{H}$, $(1, i, j, k)$. $\mathbf{v}$ and $\mathbf{r}(q)$'s value will be represented by the usual 3-tuples. To evaluate (4.4), the scalar calculations shown in Listing 4.1 can be executed (taken directly from the Ceres source code). (see Figure 4.1 for the equivalent computation graph).

These are $21m + 18a$ Floating Point Operation (FLOP)s for a single evaluation[3].

Listing 4.1: Algorithm evaluating $\mathbf{r}(q)$ defined in eq. (4.4)

```
t1=-q3*q3; t2= q0*q1; t3= q0*q2; t4= q0*q3; t5=-q1*q1;
t6= q1*q2; t7= q1*q3; t8=-q2*q2; t9= q2*q3;
r0=2*((t8+t1)*v0+(t6-t4)*v1+(t3+t7)*v2)+v0;
r1=2*((t4+t6)*v0+(t5+t1)*v1+(t9-t2)*v2)+v1;
r2=2*((t7-t3)*v0+(t2+t9)*v1+(t5+t8)*v2)+v2;
```

To calculate the local Jacobian matrix $\mathrm{J}_q\mathbf{r}$ at an arbitrary $q \in S^0(\mathbb{H})$ using a dual-number approach, we follow the method utilizing the outer vector space encompassed by (4.3)). To start we analyze the complexity of the first step in which one would calculate the Jacobian of $\tilde{\mathbf{r}} : \mathbb{H} \to \mathbb{R}^3$, defined by the pseudo code in Listing 4.1 (corresponding to $\tilde{\mathbf{f}}$ in Section 2.2). Using the dual-number approach, it is necessary to propagate four extra variables through each operation (one partial derivative per component of (q_0, q_1, q_2, q_3)) (see 2.1). For each extra variable, a multiplication in the original code requires an extra $2m + 1a$ and each addition requires an extra $1a$. This results in $21(2m + 1a) + 18a = 42m + 39a$ FLOPs per variable and $4(42m + 39a) = 168m + 156a$ for the full $4x3$ global Jacobian matrix. To build the local $\mathrm{J}_q\mathbf{r}$ we have to choose a chart around q. We will use the common exponential chart $\varphi_q : U \subset \mathbb{R}^3 \to S^0(\mathbb{H}), \mathbf{w} \mapsto \exp(w)q$, where w denotes the pure imaginary quaternion with vector part $\mathbf{w}$ and U is an environment of $\mathbf{0}$ small enough to make the map injective. Its Jacobian at $\mathbf{w} = \mathbf{0}$ can be calculated from q with negations only. The remaining step is to multiply these matrices, making the total cost $9(4m + 3a) = 36m + 27a$, assuming a straightforward implementation.

Our approach would exploit the following simplification of the general directional derivative in direction $\mathbf{w} \in \mathbb{R}^3$ to calculate $\mathrm{J}_q\mathbf{r} = \mathrm{J}_{\mathbf{0}}(\mathbf{r} \circ \phi_q)$.

$$
\begin{aligned}
\mathrm{J}_{\mathbf{0}}(\mathbf{r} \circ \varphi_q) \cdot \mathbf{w} &= (\mathrm{d}_q \mathrm{Im}(qv\bar{q}))(wq) && \text{(4.5a)}\\
&= \mathrm{Im}(wqv\bar{q} + qv\overline{wq}) && \text{(4.5b)}\\
&= \mathrm{Im}(wqv\bar{q} + qv\bar{q}\bar{w}) && \text{(4.5c)}\\
&= \mathrm{Im}(wqv\bar{q} - qv\bar{q}w) && \text{(4.5d)}\\
&= \mathrm{Im}([w, qv\bar{q}]) && \text{(4.5e)}\\
&= 2\mathbf{w} \times \mathrm{Im}(qv\bar{q}) && \text{(4.5f)}
\end{aligned}
$$

---

[3] Here, m represents multiplication operations and a represents addition operations. We ignore negations.

$$= 2\mathbf{w} \times \mathbf{r}(q) \tag{4.5g}$$

In the above, $[\cdot,\cdot]$ is the commutator of quaternions and × represents the cross product of 3×1 vectors. Note that the equations above just sketch the proof, sometimes taking advantage of concepts requiring more in-depth knowledge of DM and specifically of $\mathbb{H}$. For the sake of brevity we will stay at this level of details here.

To calculate the columns of **r**'s Jacobian matrix, one would evaluate the last expression for w substituted with each default basis vector of $\mathbb{R}^3$. In these evaluations, the cross product becomes trivial (i.e. only requiring negations), reducing the FLOP count to $3m$ needed to scale the components by a factor 2.

The example above shows how we can go from $204m + 183a$ FLOPs to $3m$ FLOPs to calculate the Jacobian by choosing a beneficial local chart and injecting knowledge about the underlying DM and the outer space at the block level. Any manifold has the potential to benefit from this method based on its specific structure, or by choosing advantageous embeddings or tangent space basis vectors. For unit-length quaternions representing rotations, one would implement multiplication and inversion, along with the log and exponential map, resulting in a full-featured BAD system for manipulation of expressions for this specific DM. For any other manifold of interest, one would follow the same procedure, writing down the operations to be supported, deriving some efficient analytical expressions for the Jacobians induced by these operations, and implementing these as operations supported by the library.

3. Implementation

This section describes the usage of our implementation and compares it to the AD package provided by the Ceres optimizer.

Using the Ceres AutoDiffCostFunction to calculate the local Jacobian of the expression $\mathbf{r}(q)$, one would write the C++ code presented in Listing 4.2. Please note the necessary extra step on line 27-29 to convert the Jacobian in global coordinates (of the embedding space) to the Jacobian in local parametrization.

Listing 4.3 illustrates how our approach could be used to solve the same problem. In this example, we use the "auto" keyword from C++ 2011 to simplify the code and let the compiler deduce the type of an expression[4].

In line 6, the values pQ is converted into something one can later take a derivative with respect to (=Diffable). The Ref template only enforces capturing per reference (as also in line 7 with v a pV).

In line 9, the high-level computation graph converted to a type and becomes (thanks to *auto*) the type of r. The data of r will only contain the references to pQ and pV.

In line 11, an intermediate derived-value cache will be created. Its type will depend on r's type and essentially include storage to store intermediate values that could be needed repeatedly in the (Jacobian) evaluation (e.g. the conjugate of q or its rotation matrix counterpart). The resulting storage is embedded per reference into the cache, to have an unified storage for all

---

[4] In fact without "auto" the whole concept would become quite cumbersome to use because the type names quickly become huge and unreadable. To allow the use of the library without knowing about these generated types is one of the big implementation challenges.

Listing 4.2: Calculating the local Jacobian of the function $\mathbf{r}(\cdot)$ with Ceres AutoDiffCostFunction

```
1   using namespace ceres;
2   struct Cv {
3     const double *v;
4     Cv(const double *v) : v(v) {}
5
6     template <typename T>
7     bool operator()(const T* const q, T* residuals) const
8     {
9       UnitQuaternionRotatePoint(q, v, residuals);
10      return true;
11    }
12  };
13
14  QuaternionParameterization quaternionParameterization;
15
16  void calcJacobian(const double *q, const double *v,
17      double *result, double *qJ)
18  {
19    const double *parameters[] = {q};
20    double qGlobalJ[3 * 4];
21    double *jacobians[] = {qGlobalJ};
22    double qLocalParamJ[4 * 3];
23
24    AutoDiffCostFunction<Cv,3,4> r(new Cv(v));
25
26    r.Evaluate(parameters, result, jacobians);
27
28    quaternionParameterization.ComputeJacobian(q, qLocalParamJ);
29    internal::MatrixMatrixMultiply<3, 4, 4, 3, 0>(
30      qLocalParamJ, 3, 4, localParamJ, 4, 3, qJ, 0, 0,  3, 3);
31  }
```

values, without the need to copy the result data back.
In line 13, intermediate values that are needed will be calculated (e.g. always the final evaluation result — here the rotated vector). In line 14, the Jacobian is computed directly into the provided storage qJ.

Because the mathematical types of all variables and operations in the expression are encoded in their C++ types, it is possible to do the following:

- to derive a suitable intermediate and derived value cache structure for the evaluation of this expression and especially its Jacobian;
- to automatically apply mathematically simplified algorithms (using template specialization and overload resolution to look them up) to evaluate the expression and especially its derivatives using this cache structure;
- to implement the Jacobian evaluation using (block) forward or backward evaluation (or some mixture of the two), whatever is the fastest for this particular fragment of the computation graph.

Listing 4.3: Using the BAD to calculate the local Jacobian of the function $\mathbf{r}(\cdot)$.

```
using namespace tex; using namespace Eigen;
void calcJacobian(const UnitQuaternion & pQ,
    const EuclideanPoint<3> & pV,
    EuclideanPoint<3> & result, Matrix3d & qJ)
{
  Diffable<Ref<UnitQuaternion>, 0> q(pQ);
  Ref<EuclideanPoint<3>> v(pV);

  auto r = q.rotate(v);

  auto cache = createCache(r, result);

  cache.update(r);
  evalFullDiffInto(r, q, cache, qJ);
}
```

4. Experiments

In this section we describe two experiments comparing our BAD prototype implementation with the dual-number approach implemented by Ceres. In the first experiment, we compare the timing of Jacobian evaluations on increasingly large problems. In the second experiment, we use our approach on real-world data in a nonlinear optimization problem whose goal is to estimate the time-varying orientation of an elephant seal in a wildlife monitoring context.

4.1. Comparison with state-of-the-art Automatic Differentiation

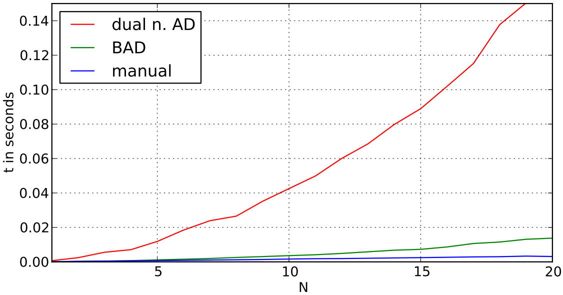


Figure 4.3.: Performance comparison for evaluating the value and Jacobian of $\prod_{i=1}^{N} \mathbf{C}_i \mathbf{v}$, as a function of N. The red line (dual n. AD) refers to the dual-number approach implemented by Ceres, the green line corresponds to BAD and the blue line is a hand-tuned version of the evaluation.

In the first experiment, we compared the AD implementation of Ceres to our prototype BAD implementation. To measure performance on an increasingly complex problem, we measured the time required to calculate the value and the Jacobian of expressions of the form,

$$\begin{array}{ll}\mathbf{C}_1\mathbf{v} & \\ \mathbf{C}_2\mathbf{C}_1\mathbf{v} & \text{for } N \in 1\ldots 20 \\ \mathbf{C}_N\ldots\mathbf{C}_1\mathbf{v}, & \end{array} \tag{4.6}$$

where $\mathbf{C}_i$ is a rotation (represented in our case by an unit-length quaternion), and $\mathbf{v}$ is a 3×1 vector. The time required by Ceres (denoted as dual-number AD) and by our approach, as a function of the problem size, is shown in Figure 4.3. On these artificial problems, there is an obvious benefit for using the specific structure of the DM to compute the Jacobian. However, we also compared it with a hand-tuned implementation of the Jacobian calculation. This results in the lowest curve (blue) in Figure 4.3. Numerically, on this specific example, we find that BAD is 12 times faster than Ceres, but still 4 time slower than the hand-tuned evaluation, independently of the problem size.

4.2. Application: Elephant-Seal Attitude Estimation

In this section, we will compare the performance of the different optimization solutions on a specific application: the estimation of the attitude of elephant seals (Fig. 4.4) based on accelerometer and magnetometer recording collected over weeks or months with sensors attached to the animals while they are at sea. Although the data-set is peculiar and might seem far outside the field of robotics, the problem of batch attitude or position estimation from initial measurement unit is a common issue for underwater and flying robots.

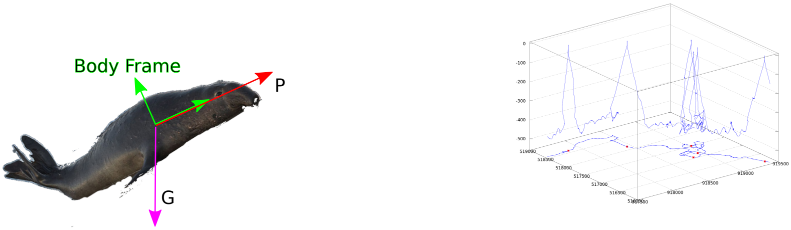


Figure 4.4.: Left: Elephant seal and attached body frame. Right: typical 3D dive reconstructed by integrating an estimated velocity vector.

The sensors attached to the animals record the following data at 1Hz: depth, acceleration, magnetic field. A sound-based system estimates the animal linear velocity but due to energy saving considerations, this is switched on only for 3 hours every 12 hours. At the surface, global positioning is retrieved from the ARGOS satellites. To illustrate the point of this paper, we will focus on using the accelerometer and the magnetometer to retrieve the animal attitude. In a later stage, this could be used in combination with the velocity measurement to estimate a 3D trajectory (Fig. 4.4), or in combination with GPS data to estimate sea currents. In a real situation, it is also necessary to use the batch estimation process to estimate some sensor calibration parameters. All these extensions will be kept out of scope for this paper.

Problem Statement

Formally, the state we are estimating is the seal attitude as a rotation matrix C_t at time t and the propulsion force P_t it applies along the x axis in its body frame (see Fig. 4.4). To this end, we have the measurements of two constant vectors in the world frame: the acceleration $G = [0,0,9.81]^T$ and the magnetic field $B = [B_x, B_y, B_z]^T$. The exact value of the magnetic field can be found using the IGRF[5] magnetic field model, which depends on the time, the latitude and the longitude. For sake of simplicity, we will assume here that the reference magnetic field is constant between two GPS fix. We will denote b_t the magnetic field reported by the sensor in the body frame, and a_t the accelerometer output. Note that the measured acceleration results from the combination of gravity and the propulsion force $P_t \cdot \vec{x}$ applied by the seal, where $\vec{x}$ is the longitudinal axis of the animal.

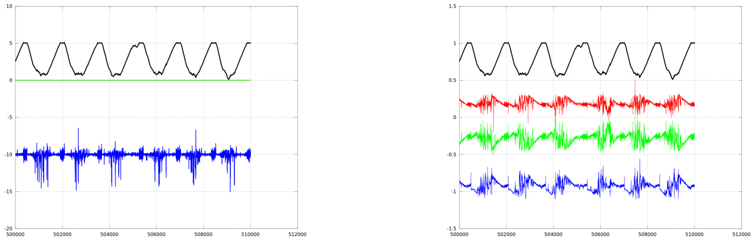


Figure 4.5.: Left: measured acceleration corrected for attitude and propulsion. Right: measured magnetic field corrected for attitude. In both graphs: the black curves represent the shape of the depth profile (scaled), the red, green and blue curves represent the x, y and z axis of the signal.

With these variables we can build the following error terms for the accelerometer and magnetometer:

$$F_{acc}(t) = C_t \cdot [a_t - P_t \cdot \vec{x}] - G \text{ and } F_{mag}(t) = C_t \cdot b_t - B \tag{4.7}$$

In addition, we can make some continuity hypothesis on the attitude and the propulsion and express them as the following error terms:

$$F_p(t) = P_t - P_{t-1} \text{ and } F_C(t) = C_t \cdot C_{t-1} \tag{4.8}$$

The initial values used for the optimizer is propulsion at zero and Euler angles (roll, pitch and yaw) estimated independently from the accelerometer and magnetometer.

---

[5] `http://www.mathworks.fr/matlabcentral/fileexchange/28874-igrf-magnetic-field`

Optimization Results

Figure 4.5 illustrates a selection of 10’000 seconds of the optimization result, out of the 1.1e6 seconds (approx. 13 days) of the complete data-set. On the left, the measured acceleration corrected for attitude and propulsion is displayed. As expected, the value is nearly constant except when unmodelled forces are applied on the animal. This happens when the seal is breathing at the surface and moves up and down with the swell, or when the seal brutally accelerates to catch a prey. The former is seen as regular and constant length high vertical acceleration every approximately 1800s, the latter is seen as more irregular event with stronger peak accelerations. On the right, the corrected magnetic field is shown. The signal is more noisy but approximately constant over these 10’000 seconds.

Comparison of optimization performance

Figure 4.6a and 4.6b shows the time, t, required to calculate the Jacobians necessary to solve the simplified seals optimization problem using single-threaded execution on a benchmark PC after loading the first N input lines of the measured data. In Figure 4.6a, we measure the time (using the x86 RDTSC instruction) in total spent in the Jacobian calculation function per error term. In Figure 4.6b, we show the time measured by the Ceres optimizer for the whole Jacobian evaluation step (with the clock used in Figure 4.6a disabled). The solid line represents our implementation of BAD and the dashed one shows Ceres implementation of the dual-number approach. Both show the performance improvement by BAD. In Figure 4.6a, BAD is approximately 2.5-times faster, in Figure 4.6b it is only about 13.5%. The explanation for this example is that Ceres spends a relatively large amount of time constructing the overall Jacobian for the complete optimization problem from the Jacobians of the individual error term, which are rather simple error terms. Ceres includes the full sparse Jacobian matrix construction in what it reports as the Jacobian calculation time.

To be complete, we must state here that the comparison here is slightly unfair in favor of Ceres for two reasons. Ceres expects global Jacobians (the natural output of a scalar AD) instead of the local Jacobians that they later calculate from these global ones by multiplying by the Jacobian of the local parametrization (see (4.3)). To fit with this requirement, we calculate $n \times 4$ matrices from our $n \times 3$ Jacobian and Ceres then calculates the $n \times 3$ back (n is the dimension of the error term). Each of these conversions requires a matrix multiplication (first 3×4 then 4×3). The first multiplication delays our implementation in Figure 4.6. It would be fairer to skip these artificial steps but it would require changes in the Ceres code, which would further improve the performance of BAD in Figure 4.6b. In addition, Ceres uses non-aligned Eigen::Vectors, while our implementation needs aligned ones, which requires us to copy the data into aligned vectors. The time spent copying these vectors is also counted in the data depicted in Figure 4.6.

5. Conclusions

In this paper we have brought together the concepts of AD and DM to develop a block AD approach that we cal BAD. This approach has the potential to be much more computationally efficient than traditional scalar AD by exploiting the specific structure of the DM. We presented a worked example of an unit-length quaternion rotating a 3×1 vector and showed how it can

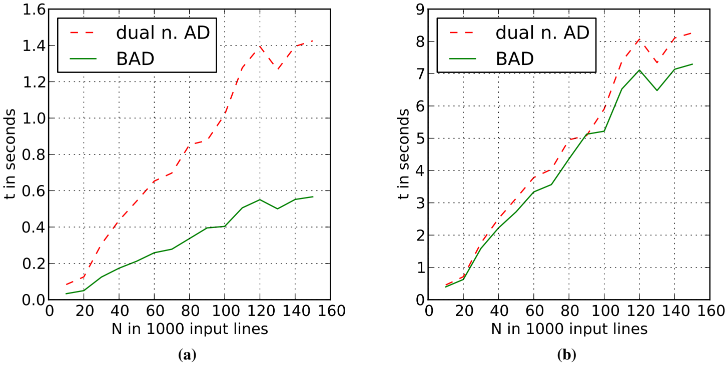


Figure 4.6.: Time required to compute the Jacobian as a function of the number of error terms for the simplified seal problem. On the left, only Jacobian computation time is included, on the right, the time for the creation of the full sparse Jacobian matrix is also included. See text for details.

greatly reduce the number of instructions needed for Jacobian computation. Finally, we presented experimental results on simulated and real data that show that our prototype implementation of BAD is indeed faster than a state-of-the-art AD approach.

Our next step will be to finalize the interface and implement a full-featured BAD library in C++ encompassing the most common DMs and operations needed in robotics.

6. Acknowledgement

Cédric Pradalier is supported by the European Commission within the Noptilus project, EC Grant agreement 270180. The elephant-seal data-set was kindly provided by Ch. Guinet and B. Picard from the CNRS Center for Biological Study in Chizé, France, based on data collected on the Kerguelen Islands, South Indian Ocean.

Paper II

Why and How to Avoid the Flipped Quaternion Multiplication

Hannes Sommer, Igor Gilitschenski, Michael Bloesch, Stephan Weiss,
Roland Siegwart, and Juan Nieto

Abstract

Over the last decades quaternions have become a crucial and very successful tool for attitude representation in robotics and aerospace. However, there is a major problem that is continuously causing trouble in practice when it comes to exchanging formulas or implementations: there are two quaternion multiplications commonly in use, Hamilton's multiplication and its flipped version, which is often associated with NASA's Jet Propulsion Laboratory. This paper explains the underlying problem for the popular passive world-to-body usage of rotation quaternions, and promotes an alternative solution compatible with Hamilton's multiplication. Furthermore, it argues for discontinuing the flipped multiplication. Additionally, it provides recipes for efficiently detecting relevant conventions and migrating formulas or algorithms between them.

Published in:
MDPI Aerospace, 2018
doi:10.3390/aerospace5030072

Nomenclature

$\mathbb{R}^{k\times 1}$	$\simeq \mathbb{R}^k$ **implicitly** identified $\forall k \in \mathbb{N}$
$\cdot \times \cdot$	*cross product*
$\cdot^{\times}$	$: \mathbb{R}^3 \to \mathbb{R}^{3\times 3}, \forall \mathbf{x},\mathbf{y} \in \mathbb{R}^3 : (\mathbf{x}^{\times})\mathbf{y} = \mathbf{x}\times\mathbf{y}$
$g\circ f$	$: \mathrm{dom}(f) \to \mathrm{codom}(g), x \mapsto g(f(x))$, g *after* f
$\langle\cdot,\cdot\rangle$	inner product (also called scalar or dot product)
$\mathbb{H}$	4D Euclidean $\mathbb{R}$-vector space of *quaternions* (without quaternion-multiplication)
$\mathscr{B}_{\mathbb{H}}$	$:= (\mathbf{1},\mathbf{i},\mathbf{j},\mathbf{k})$, orthonormal *standard basis* for $\mathbb{H}$
$\mathscr{U}$	$:= \{\mathbf{q}\in\mathbb{H} \mid \|\mathbf{q}\| = 1\}$, *unit-length* quaternions
$\odot$	Hamilton's quaternion multiplication, i.e. $\mathbf{i}\odot\mathbf{j}\odot\mathbf{k} = -\mathbf{1}$
$\otimes$	Shuster's flipped quaternion multiplication [146]: $\forall \mathbf{p},\mathbf{q}\in\mathbb{H} : \mathbf{q}\otimes\mathbf{p} = \mathbf{p}\odot\mathbf{q}$
$\mathscr{I}(\mathbf{q})$	$: \mathbb{H}\to\mathbb{R}^3, \mathbf{q}\mapsto (\langle\mathbf{q},\mathbf{i}\rangle, \langle\mathbf{q},\mathbf{j}\rangle, \langle\mathbf{q},\mathbf{k}\rangle)$, *imaginary* components of $\mathbf{q}$
$\overrightarrow{\mathbf{q}}$	$:= \mathscr{I}(\mathbf{q})$ (shorthand)
$\mathscr{I}^*(\mathbf{x})$	$: \mathbb{R}^3\to\mathbb{H}, (x_i)_{i=1}^3 \mapsto x_1\mathbf{i}+x_2\mathbf{j}+x_3\mathbf{k} \Rightarrow \mathscr{I}\circ\mathscr{I}^* = \mathrm{id}_{\mathbb{R}^3}$
$\mathbf{q}_r$	$:= \langle\mathbf{q},\mathbf{1}\rangle \in \mathbb{R}$, *real component of* $\mathbf{q}$
$\bar{\cdot}$	$: \mathbb{H}\to\mathbb{H}, \mathbf{q}\mapsto q_r\mathbf{1} + \langle\mathbf{q},\mathbf{i}\rangle\mathbf{i} + \langle\mathbf{q},\mathbf{j}\rangle\mathbf{j} + \langle\mathbf{q},\mathbf{k}\rangle\mathbf{k}$, *conjugation*
SO(3)	$:= (\{\mathbf{M}\in\mathbb{R}^{3\times 3} \mid \mathbf{M}^{-1} = \mathbf{M}^T \wedge \det(\mathbf{M}) = 1\}, \cdot)$

1. Introduction

"*The quaternion [146] is one of the most important representations of the attitude in spacecraft attitude estimation and control.*" With these words Malcolm D. Shuster opened his introduction of [147], "The nature of the quaternion" (in 2008). It details on a conventional shift from Hamilton's original quaternion multiplication, $\odot$[1], to its flipped version, $\otimes$, Shuster had successfully advocated for in the 1990s—in particular with the very influential [146].[2] Shuster called $\otimes$ the *natural* multiplication, and called the $\odot$ *traditional*. We will refer to the former as *Shuster's multiplication* and to the latter as *Hamilton's multiplication*, because we consider "natural" too biased and because Shuster's labels are generally rather unknown. Back then, the goal of the introduction was to prevent confusion and mistakes when dealing with chains of rotations (*compositions*) motivated by the application of spacecraft attitude estimation and control [147]. However, the introduced conventional split from other fields (e.g., mathematics, physics, or computer graphics/visualization) caused a great deal of confusion and is a source for mistakes in practice (see Section 1.2). Therefore, in this paper, we advocate to undo this split and return to only using Hamilton's original multiplication. At the same time, we show how an alternative approach can be used to solve the original problem addressed in [146].

---

[1] Hamilton used the symbol $\cdot$ instead. We use $\odot$ in this paper to better contrast $\otimes$ and stress the symmetry. For documents using Hamilton's multiplication only we recommend to just use the plain $\cdot$ or the default operator (omitted symbol), because they are the default multiplication symbols (for quaternions) in mathematics, which should not be needlessly questioned as the standardizing authority. See also Section 2.

[2] For example, the well known *JPL-convention* chooses $\otimes$ but it also includes other conventional decisions [150].

1.1. Original Problem and Shuster's Solution

The *historical problem* with Hamilton's multiplication only appeared in connection with an arguably arbitrary, specific convention on how to assign direction cosine matrices (DCM)[3], $\mathbf{C}_S : \mathscr{U} \to \mathrm{SO}(3)$ to unit-length quaternions. When using this assignment, as in [146], the product of two quaternions, $\mathbf{p} \odot \mathbf{q}$, would correspond to the product of the two corresponding DCMs, but with reversed order:

$$\forall \mathbf{p}, \mathbf{q} \in \mathscr{U} : \mathbf{C}_S(\mathbf{p} \odot \mathbf{q}) = \mathbf{C}_S(\mathbf{q}) \cdot \mathbf{C}_S(\mathbf{p}), \tag{5.1}$$

i.e., the mapping, $\mathbf{C}_S$, was not a *homomorphism* $(\mathscr{U}, \odot) \to (\mathrm{SO}(3), \cdot)$ but an *antihomomorphism*, which is generally more uncommon and confusing and therefore more error-prone. Introducing $\otimes$ is *Shuster's solution* to this problem [146]:

$$\mathbf{p} \otimes \mathbf{q} := \mathbf{q} \odot \mathbf{p} \;\; \Rightarrow \;\; \mathbf{C}_S(\mathbf{p} \otimes \mathbf{q}) = \mathbf{C}_S(\mathbf{p}) \cdot \mathbf{C}_S(\mathbf{q}). \tag{5.2}$$

1.2. The Problem Today with Shuster's Solution

Since its introduction, a great fraction of spacecraft literature has adopted $\otimes$ [147]. On the other hand, virtually the entire rest of the scientific community dealing with quaternions is apparently not following, and partially not even aware of this transition in the spacecraft community, as we will show in Section 5. Having two different quaternion multiplications in use comes at a significant and ongoing cost: formulas and implementations need translation or adaptation, let alone the fact that one must first identify which multiplication is used in a given formula or algorithm. The latter can be particularly tedious if the authors are not aware of the two possibilities and hence do not mention which one they use[4]. If a reader is unaware of the split, the discovery that two different quaternion multiplications are in use, and that in fact "the other" was employed, might be made only after several failures.

The troubles may be little per subject but they are continuously affecting a significant number of scientists and engineers, causing delays in the best case. This accumulated cost is particularly large for interdisciplinary domains such as robotics that lean on publications from both conventional worlds, for instance aerospace and computer vision. In fact, in robotics, there are several recent publications for each of the two multiplications (see Section 5). It is also a pressing matter because the effort to correct becomes larger with every new implementation or publication using a convention different from what the community will agree on eventually.

1.3. Contribution

Given this current situation, it is an interesting question whether the proposed switch was necessary or at least the best of all available options. Motivated by this question, this work provides the following contributions[5]:

---

[3] Equivalent in value to a corresponding coordinate transformation matrix (between oriented orthonormal bases).

[4] As a small self experiment, the reader in doubt is invited to try to figure out which quaternion multiplication the popular c++ template library for linear algebra, Eigen, is using.

[5] An extended version of this paper with in-depth arguments, derivations, proofs, and more additional material can be found in [151].

- give an overview over the two quaternion multiplications and their consequences for representing SO(3);
- identify the problem with commonly using both;
- investigate and refute the necessity of the flipped multiplication defined in Equation (5.2) to resolve the antihomomorphy problem expressed in Equation (5.1);
- promote and explain a neglected alternative solution;
- demonstrate that this alternative yields a more formal similarity to corresponding formulas using matrices;
- advocate to discontinue Shuster's multiplication;
- provide recipes to detect and migrate between quaternion multiplication conventions.

To the best of our knowledge, this is the first work explicitly addressing the ambiguity of quaternion multiplication as a problem and proposing a potential solution.

1.4. Outline

We start with introducing our notation in Section 2 and important background information in Section 3. Next, we propose an alternative solution to the original problem in Section 4. In Section 5, we summarize relevant contributions on quaternion multiplication related conventions, and tabulate their popularity in recent or influential publications. Then we argue for discontinuing Shuster's multiplication in Section 6. Lastly, we provide recipes for detecting and migrating rotation quaternion related conventions in Section 7.

2. Notation

In this section, we introduce our nonstandard notation. An outline of the entire notation can be found in the nomenclature on page 50.

Regarding quaternion notation this paper has two major challenges: First, we must distinguish two different quaternion multiplications and consequentially two different quaternion structures having most of their structure in common. Second, we do not want to load most readers with a representation of quaternions different from what they are used to.[6] We approach these challenges purely axiomatically by assuming $\mathbb{H}$ to be only a 4D $\mathbb{R}$-vector space of *quaternions* with inner product $\langle\cdot,\cdot\rangle$ and an orthonormal standard basis $\mathscr{B}_{\mathbb{H}} := (\mathbf{1}, \mathbf{i}, \mathbf{j}, \mathbf{k})$. Pairing $\mathbb{H}$ with Hamilton's quaternion multiplication, $\odot$, yields Hamilton's quaternions, $(\mathbb{H}, \odot)$, a skew field (also division ring), with multiplicative identity $\mathbf{1}$ and additional defining relations:

$$\mathbf{i} \odot \mathbf{i} = \mathbf{j} \odot \mathbf{j} = \mathbf{k} \odot \mathbf{k} = \mathbf{i} \odot \mathbf{j} \odot \mathbf{k} = -\mathbf{1}.$$

We use $\odot$ instead of Hamilton's original $\cdot$ in this paper to better contrast $\otimes$ while at the same time stressing the symmetry. For documents using Hamilton's multiplication only, we recommend

---

[6] For instance, by assuming the real part to be the first component of a real quadruple instead of its last component.

using the plain $\cdot$ or the default operator (omitted symbol), because they are the default multiplication symbols (for quaternions) in mathematics, which should not be needlessly questioned as the standardizing authority.

Pairing $\mathbb{H}$ instead with Shuster's quaternion multiplication, $\otimes$, the flipped version of $\odot$, yields Shuster's quaternions, $(\mathbb{H}, \otimes)$, a similar skew field, but with slightly different additional relations:

$$\mathbf{i} \otimes \mathbf{i} = \mathbf{j} \otimes \mathbf{j} = \mathbf{k} \otimes \mathbf{k} = \mathbf{k} \otimes \mathbf{j} \otimes \mathbf{i} = -\mathbf{1}.$$

The real and imaginary coordinates/components of a quaternion $\mathbf{q} \in \mathbb{H}$ we denote with $q_r := \langle \mathbf{q}, \mathbf{1} \rangle \in \mathbb{R}$ and $\overrightarrow{\mathbf{q}} := \mathscr{I}(\mathbf{q}) := (\langle \mathbf{q}, \mathbf{i} \rangle, \langle \mathbf{q}, \mathbf{j} \rangle, \langle \mathbf{q}, \mathbf{k} \rangle) \in \mathbb{R}^3$, respectively. The right-inverse of $\mathscr{I}$, mapping an imaginary coordinate triple into a purely imaginary quaternion we denote with $\mathscr{I}^* : \mathbb{R}^3 \to \mathbb{H}, (x_i)_{i=1}^3 \mapsto x_1 \mathbf{i} + x_2 \mathbf{j} + x_3 \mathbf{k}$. The quaternion conjugation of $\mathbf{q}$ we denote with $\bar{\mathbf{q}}$.

3. Background: Duality of Rotation Representations

Physical rotations are conceptually rather unambiguous. However, when it comes to describing them, there are two popular ways to do so. The *active* way is to describe what happens to the rotated body's points from a fixed *world frame*'s perspective. The *passive* way is to describe how coordinates of fixed world points change from the perspective of the rotated *body frame* (the two ways are also distinguished as *alibi* for active and *alias* for passive). The passive way comes in two flavours, *world-to-body* (PWTB) and *body-to-world* (PBTW). The first describes the change from before rotation to after and the second from after to before[7]. We call these options the *usage* of the mathematical model to describe the physical rotation.[8] Mathematically, the described actions on the observed points have identical properties in all three cases matching precisely the actions of the special orthogonal group (SO)[9]. However, because, given the *same* physical rotation, PWTB describes the inverse compared to PBTW, they yield mutually inverse group elements. Despite being conceptually very different, the active usage always yields the same group element as PBTW. Consequentially, we will often not distinguish these two.The crucial consequence is that, while every *parametrization* of the SO can be employed for all usages, switching usage from or to PWTB corresponds to an inversion. The main target audience of this paper are those who favour the PWTB usage because the problem represented by Equation (5.1) is exclusive for the PWTB usage.

4. The Proposed Alternative Solution

The alternative solution for the original problem represented by Equation (5.1) we suggest is the use of a different correspondence rule, $\mathbf{C}_H : \mathscr{U} \ni \mathbf{q} \mapsto \mathbf{C}_S(\bar{\mathbf{q}})$.[10] Assuming we keep fixed how rotation matrices represent rotations, this new correspondence inverts the corresponding quaternion. It also yields the desired homomorphy from quaternions to rotation matrices together

---

[7] The names stem from assuming that body and world frame are initially aligned. Alternative: *local to global* vs. *global to local* as in [150].

[8] In [150], the authors use "*function*" for active vs. passive without flavours.

[9] This paper only deals with SO(3).

[10] Explicit forms for $\mathbf{C}_S$ and $\mathbf{C}_H$ would only detract from our argument here, but can be found in Table 5.2.

with $\odot$ and the usual matrix multiplication:

$$\forall \mathbf{p}, \mathbf{q} \in \mathscr{U} : \mathbf{C}_H(\mathbf{p} \odot \mathbf{q}) = \mathbf{C}_H(\mathbf{p}) \cdot \mathbf{C}_H(\mathbf{q}). \tag{5.3}$$

This suggestion might falsely give the negative impression of an "additional" inversion. In fact, it could equally well be that for $\mathbf{C}_S$ the "wrong" (inverted) quaternions were chosen. And our observations in Section 6.4 support this interpretation. Please note that both solutions yield the usual way to apply the rotation represented by a quaternion[11] (see also Example 7.2):

$$\forall \mathbf{x} \in \mathbb{R}^3, \mathbf{q} \in \mathscr{U} : \begin{array}{l} \mathbf{C}_H(\mathbf{q})\mathbf{x} = \mathscr{I}(\mathbf{q} \odot \mathscr{I}^*(\mathbf{x}) \odot \mathbf{q}^{-1}) \\ \mathbf{C}_S(\mathbf{q})\mathbf{x} = \mathscr{I}(\mathbf{q} \otimes \mathscr{I}^*(\mathbf{x}) \otimes \mathbf{q}^{-1}) \end{array}. \tag{5.4}$$

5. Literature Review

5.1. Literature on or Introducing Relevant Conventions

Aerospace

According to [40], Shuster's multiplication, $\otimes$, was introduced in [94]. According to [157], it was later proposed as a standard in [33]. Unfortunately, the exact details are impossible for us to reconstruct as NASA has never published [33]. In [81], the authors derive a homomorphic composition rule for the *Euler (symmetric) parameter* (ESP)[12], effectively equal to $\otimes$, and infer a required conjugation from ESP to Hamilton's quaternions. In Section 4, we propose the same quaternion values but with different derivation.

In [147], the author describes the transition from Hamilton's $\odot$ to Shuster's $\otimes$ for "*spacecraft attitude estimation and control*" literature, citing [146] as "*probably, more than any other work*", responsible for the conventional shift. In [146], $\otimes$ is indeed derived as necessity given some requirements, of which one was the correspondence $\mathbf{C}_S$. At the same time, Shuster already speculated why mathematicians had no reason to reform the quaternions: "*The concern of pure mathematics is not in representing physical reality efficiently but in exploring mathematical structures … As engineers, our interest is in "im-pure" mathematics, contaminated by the needs of practical application.*" A major struggle with the quaternion conventions within NASA that led to a conventional switch from the Space Shuttle program (STS) to the (American) International Space Station (ISS) software standard is reported in [162]. According to the author both conventions use PWTB quaternions, but with mutually inverse values. Hence, one must use $\mathbf{C}_S$ (ISS) and one $\mathbf{C}_H$ (STS). Both use Hamilton's multiplication, yielding an antihomomorphic quaternion-to-matrix conversion for ISS. Apparently, this drawback was preferred over using $\otimes$, which would yield the homomorphic convention from [146]. The author compares these two conventions with a third convention that he calls *Robotics*, which uses PBTW quaternions. Neither [146] nor [147] are mentioned in [162] despite their apparent relevance. Similarly, the problem of antihomomorphy is entirely left out. The only easily accessible reference for quaternions in [152, 162] uses Hamilton's multiplication and also does not mention Shuster's work. Sadly, its reference for the STS quaternions, [34], is not publicly available. Therefore,

---

[11] This fact is actually equivalent to both being homomorphic.

[12] Usually treated as equal to the rotation quaternion components

we cannot verify the impression that the STS quaternions are exactly what we are proposing in Section 4. If this is the case, something in the line of the argument quoted in [162] must have been the reason to abandon the STS quaternion convention: "*Define the application's fundamental quaternion with no negative signs.*"[13]. Again, this is impossible for us to verify due to NASA's lack of publications on the matter. For the same reason, we neither know the foundation of the quoted rule nor how it translates to other circumstances.

The *Navigation and Ancillary Information Facility* (NAIF), a sub-organization of NASA's Jet Propulsion Laboratory (JPL), uses Hamilton's multiplication and $\mathbf{C}_H$ everywhere in their primary software suite called *SPICE* [122]. The authors use the short terms *SPICE quaternions*, using $\odot$, and *Engineering quaternions*, using $\otimes$, and describe the first as "*Invented by Sir William Rowan Hamilton; Frequently used in mathematics and physics textbooks*" and the second as "*Widely used in aerospace engineering applications*" (e.g., cspice/qxq_c).

In [104, pp. 37,38] both multiplications are introduced, but the authors focus on $\otimes$ because it "*has proven to be more useful in attitude analysis*", while being aware of the trouble with quaternion conventions: "*Care must be taken to thoroughly understand the conventions embodied in any quaternion equation that one chooses to reference.*"

Robotics

In [150], the author proposes a classification of rotation quaternion-related conventions based on four binary choices leading to "*12 different combinations*" (only 12 instead of 16 because one binary choice, body-to-world vs. world-to-body is only distinguished in the passive case). The author also points out two most commonly used conventions, one using Shuster's and one using Hamilton's multiplication, which the author calls *JPL-convention* and *Hamilton-convention*, respectively. He chooses to use the latter because it "*coincides with many software libraries of widespread use in robotics, such as Eigen, ROS, Google Ceres, and with a vast amount of literature on Kalman filtering for attitude estimation using IMUs ([36, 92, 108, 126, 139], and many others).*" We discourage the usage of the name *Hamilton-convention*, because only one decision of the four is associated with the name Hamilton, namely the multiplication (or *algebra* as it is called in [150]) and there are many very different conventions using this multiplication. The conversions between rotation vector, matrix, and quaternion are not among the four choices. Instead, they are consequences and are only presented for the chosen convention (for quaternion, matrix, and vector) and not for the other possible quaternion conventions. Since the chosen convention includes PBTW (passive + *local-to-global*) as two further choices, the problem represented by Equation (5.1) is circumvented and not discussed as a potential problem for other combinations of the four choices (antihomomorphy occurs only when combining $\odot$ with PWTB usage of quaternions; see Section 3).

5.2. Popularity in Influential and Recent Literature

In Table 5.1, we group publications, which are recently dealing with rotation quaternions, by (i) which quaternion multiplication is used, (ii) the type of publication, and (iii) the scientific community. Of course, this is merely a small selection of corresponding publications in the

---

[13] It is referring to the conversion from rotation vector to quaternion; see Section 6.1.

respective fields.

Within these constraints the table shows that all relevant online general purpose or mathematical encyclopedias, which we could find, use Hamilton's multiplication (without mentioning any alternatives)—and so do all major software packages. Most of them also use precisely the quaternion-to-matrix mapping we suggest as part of an alternative solution to Equation (5.1) in Section 4. The only scientific communities we could find using Shuster's multiplication are those of aerospace and robotics. Both seem to be divided to this day, including even NASA itself and its substructure JPL. Please note that only [81] (from 1986) and probably [34] (from 1975; not publicly available; only indirect and incomplete knowledge through [162]) combines $\odot$ with the PWTB usage and homomorphic quaternion-to-matrix conversion. All other users of this multiplication either accept the antihomomorphy, or use PBTW or active quaternions (indicated in the table by the non-B flags in parentheses). The frequent use of PWTB (all except such with -, a, or w-flag) indicates that this is not due to a general lack of popularity or usefulness of the PWTB usage. To the contrary, it can be considered state of the art in aerospace (only two, from 1987 and 2004 and with little influence, do differently among our 17 samples). Additionally, it was used for the Space Shuttle program and is still used on the International Space Station [162]. We believe that precisely this popularity of PWTB together with a lack of examples or understanding how to combine it with $\odot$ still poses an unsolved problem for many who prefer or only know $\odot$. The alternative solution we propose in Section 4 solves this problem and unlocks PWTB for all proponents of $\odot$ without leading to an error-prone antihomomorphism.

6. Rationale against Shuster's Multiplication

6.1. Overview

First of all, it is important to emphasize that the question of which quaternion multiplication to use should be considered a pragmatic question. Conceptually, there is no problem with any of the two multiplications nor with having both commonly used. Having said that, our main argument is as follows.

Both Multiplications Are Equally Capable and Interchangeable for Every Application

As we explain in Section 6.2, one multiplication is as good as the other in terms of applicability—importantly also when representing 3D rotations.

One of the Two Multiplications Should Be Discontinued

Having two quaternion multiplications in formulas and software comes at a significant and ongoing cost (see Sections 1.2 and 5), and does not provide benefits (see Section 6.1). It is also not necessary to accept or maintain everything that is consistent and sound, as we explain in Section 6.3. Hence, we should discontinue one multiplication, i.e., at least avoid using it in future educative material, books, and software libraries.

Table 5.1.: Popularity of the two multiplications [1]

Type/Community	**Hamilton** [2]	**Shuster**
Online encyclopedias	Encyclopedia of Math., Wolfram Mathworld, Planetmath, Britannica, Wikipedia	
Mathematics	[92](B!), [70](Ba), [62](Ba), [96](Baw), [138](B-)	
Aerospace	[34], [81](B), [29](Bw), [92](B!), [152](B!), [143](w), [23](!)	[94], [146], [40], [43], [26], [21], [104], [129], [60], [112]
Robotics	[125](a), [36](aw), [121](Bw), [139](w), [108](w),[150](w), [56](w)	[157], [46], [26], [98], [100]
Mechanics	[141](w), [142](w), [75](Baw), [71](Ba), [53](Ba) [110](Ba)	
Control	[143](w), [31](!), [56](w), [64](!)	
Computer vision	[125](a), [119](aw), [139](w)	
Computer graphics/visualization	[144](!), [119](aw), [68](aw), [159](B!), [96](Baw), [90](-)	
Applications and software libraries	Wolfram Mathematica, Matlab's aerospace (!) and robotics toolbox, C++ library Eigen, Google Ceres, Boost, GNU Octave, ROS, NASA's SPICE (qxq_c, m2q_c)	Microsoft's DirectX-Math Library[3]

[1] The references are roughly in chronological order.

[2] Parentheses decorate differences from a default (article, passive world-to-body usage): (B) book, (-) no rotations used, (a) active usage, (w) passive body-to-world usage, (aw) $\simeq$ (a) or (w), (!) antihomomorphic QM-convention (i.e., using $\mathbf{C}_S$ in the Hamilton column).

[3] This is an unclear case since the documentation of the multiplication function states: "Returns the product of two quaternions as Q2*Q1", where the multiplication function arguments are in the order Q1, Q2; i.e., Shuster's multiplication is implemented, while the documentation uses Hamilton's multiplication (all implicitly).

Hamilton's Multiplication Is Preferable

Hamilton's multiplication is predominant over all existing publications. This is because (i) it is about 150 years older and (ii) Shuster's multiplication has not spread to mathematics or theoretical physics and probably never will. The latter arises from the fact that both disciplines have very different needs that typically do not even lead to the original problem expressed in Equation (5.1) as already observed in [146]. To the best of our knowledge, Shuster's multiplication only spread from aerospace to parts of robotics, which has similar needs and is strongly influenced by aerospace literature. Additionally, Hamilton's multiplication is used by the vast majority of current software (see Table 5.1). These observations make it extremely difficult to discontinue Hamilton's multiplication instead. Even more so, pure mathematics should be almost impossible to influence in that respect because it is against the very nature of mathematics to change such a definition without an inner-mathematical reason. Additionally, as we show in Section 6.4, Hamilton's multiplication has its benefits over Shuster's from a formal point of view. These benefits make it less error-prone if combined with $\mathbf{C}_H$, which is at least as old and well-known as $\mathbf{C}_S$ ($\mathbf{C}_H$ is usually referred to as the *Euler–Rodrigues formula*). It is typically used whenever rotations are represented actively or PBTW (see Section 3). Because of that, $\mathbf{C}_H$ is also already predominant among existing software implementations (see Table 5.1).

Employing $\mathbf{C}_H$ also for the very popular PWTB usage, as we suggest, has a consequence that one could consider its biggest drawback: It introduces the comparatively rarely seen *passive quaternion* (see Table 5.2, Figure 5.1), and thus induces an unusual conversion from an active rotation vector, $\boldsymbol{\phi}$: $\mathbf{q}(\boldsymbol{\phi}) = \exp(-0.5\boldsymbol{\phi})$. Eventually, the original problem, expressed by Equation (5.1) leaves two options besides not solving it: (i) introducing an unusual quaternion multiplication (Shuster's solution) and (ii) introducing an unusual conversion from the active rotation vector to the (passive) rotation quaternion (used, e.g., in [81]). Having only these two options is probably the very difficulty behind the overall struggle and what prevented a standardization so far (see Section 5). Our position, in favour of (ii), emerges from rephrasing the question equivalently: What should be specific for the PWTB usage of quaternions when compared to the active usage, their multiplication, or the conversion to the active[14] rotation vector? It reveals an immediate argument for (ii) the conversion from the active rotation vector to the PWTB quaternion should even be expected to be different from its conversion to an active quaternion (compare also Section 3). An argument against (i) is that the multiplication is used much more often, because conversions are merely interface operations and usually do not appear where the heavy lifting is done. Additionally, if we have to choose the multiplication depending on the usage of quaternions to represent rotations, which multiplication is right for other applications? Furthermore, it is very unusual and therefore unexpected that a fundamental algebraic operation such as multiplication is a matter of dispute and convention, while it is quite common for representations. Finally, the very same question for rotation matrices was answered historically: there is no flipped matrix multiplication in regular use. Instead, it is fully accepted that there is an active and a passive rotation matrix which relate differently to an active rotation vector. Why not accept the exact same for rotation quaternions when employing them for exactly the same purpose?

---

[14] There is good reason to call the usual rotation vector active. However, it is not required here. Our argument works as well for a neutral rotation vector notion.

6.2. Equal Capability Argument

In this section, we provide support for Section 6.1 by arguing why both multiplications are equally capable. The migration recipes in Section 7.2 show that any algorithm or formula can be migrated between the two multiplications in very simple steps: just introduce quaternion conjugation at some places and exchange all multiplications. This already shows that both multiplications can provide the same service to the user (writer, reader, or programmer), as the migration concepts work in both directions.

The question remains whether one implementation could be computationally more efficient than the other. However, one should easily be convinced that the extra cost for some additional conjugations is negligible for most use cases since it only involves scalar negations. Furthermore, for very performance-critical parts of an implementation, the negations can almost always be statically merged into other operations or eliminated through algebraic simplification.

6.3. Why Consistency Is Not Enough

One could counter argue against Section 6.1 that what ever is mathematically sound and consistent should or even must be accepted out of principle. However, this is a disastrous position, because when taken to the extreme that would mean that every publication should be allowed to introduce or refer to its own version of mathematics or even use its own private language! However, the immense power and usefulness of mathematics and language comes not only from its consistency but very much from how much it is shared. Especially as engineers or scientists, we benefit from the effort put into standardization and unification by, e.g., mathematicians on a daily basis. It is only due to this effort that we can readily use answers from a mathematical software or encyclopedia. Furthermore, it would work even better if we all cared more about standardization. For example, imagine engineers chose freely, albeit in a consistent manner, between the standard matrix product and its flipped version in their implementations and publications. What pain and massive economic drawback would this little additional freedom cause? Every paper would need to introduce its matrix multiplication before using it—just as it is now with quaternions. We are not in that situation **only** because there is an accepted standard matrix product and it is treated as such.

When it comes to representation of 3D rotations, we are used to having many conventions and this might reinforce this false idea that we must accept any consistent convention. In this case, one of the underlying reasons is that there are in fact conventional options that have benefits and drawbacks given an application, such as the usage (see Section 3), and others that do not. It is important to distinguish these two cases. The position of this paper is that the quaternion multiplication is of the second type.

In summary, we should not adopt the simplistic position from above. Instead, we must invest in the assessment for every consistent convention candidate whether it is unnecessary or useful and worth its additional (future) cost. And unfortunately, sometimes we must even do that for a convention that is already widely used.

6.4. Formal Differences

In this section, we are going to investigate the formal differences of the two alternative solutions to the original problem expressed in Equation (5.1) to support parts of Section 6.1. Please note that this does not contradict our observation of equal capability in Section 6.2, because neither similarity nor symmetry affect the practical capability of a mathematical tool. Instead, it affects mostly the probability of human error when dealing with the tool, which can be a strong argument, too. In fact, it is the very type of argument with which [146] justifies the introduction of Shuster's multiplication.

We start without any assumptions about how unit quaternions and proper orthogonal matrices are conceptually used for modeling rotations[15] and only focus on how quaternions are converted to matrices. This conversion must be at least implicitly determined by all consistent rotation conventions involving rotation matrices and rotation quaternions. The benefit is that our abstract observations automatically apply to all such conventions. In particular, the $\mathbf{C}$ in $\mathbf{C}_S$ and $\mathbf{C}_H$ no longer stands for the direction cosine matrix, but for any rotation matrix concept.

QM-Conventions

We call a convention that determines a specific conversion, $\mathbf{C} : \mathscr{U} \to \mathrm{SO}(3)$, and a particular quaternion multiplication, $\star$, a *QM-convention* and represent it with $(\mathbf{C}, \star)$. In this way, many common conventions collapse to a single representative. However, this is an intended abstraction. The ignored details do not impact the following analysis, i.e., it holds equally for all conventions collapsing to a given representative. We call a QM-convention *homomorphic* iff the multiplication order is preserved when switching between quaternions and matrices, as in Equation (5.3), *antihomomorphic* iff the order flips, as in Equation (5.1), and inconsistent otherwise. There are only four options for consistent QM-conventions, namely all pairs in $\{\mathbf{C}_H, \mathbf{C}_S\} \times \{\odot, \otimes\}$. It seems commonly accepted that a homomorphy is preferable and less error-prone reducing the good candidates to the two *homomorphic QM-convention*s, $(\mathbf{C}_H, \odot)$ and $(\mathbf{C}_S, \otimes)$, with their homomorphy given in Equation (5.3) and Equation (5.2), respectively.

Formal Comparison Independent of the Usage

In Table 5.2, we compare the two homomorphic conventions. Here $\boldsymbol{\phi}$ denotes a *rotation vector* in $\mathbb{R}^3$. The *convention-factors* $\alpha_{\mathbf{C}}, \alpha_{\boldsymbol{\phi}} \in \{-1, 1\}$[16] encode the usage of rotation matrices ($\alpha_{\mathbf{C}}$) and rotation vectors ($\alpha_{\boldsymbol{\phi}}$). The usage of the quaternion follows from $\alpha_{\mathbf{C}}$ given the chosen $(\mathbf{C}, \star)$ and is identical to $\alpha_{\mathbf{C}}$ for homomorphic QM-conventions. We'll look deeper into their effect in Section 6.4. The indices A and B for the angular velocities, $\boldsymbol{\omega}$, indicate differences stemming from frame assignments for $\boldsymbol{\omega}$ and the usage of $\mathbf{C}$. Depending on these choices one of the two formulas is correct assuming another convention factor, $\alpha_{\boldsymbol{\omega}} \in \{-1, 1\}$ is suitably chosen.[17]

The neutral differences are highlighted in blue. Green vs. red highlights the benefits of $(\mathbf{C}_H, \odot)$ in terms of more similarity between corresponding rotation matrix and quaternion equations. The

---

[15] For instance, active/passive/DCM matrices.

[16] Both active and PBTW are represented by 1 and both PWTB and DCM by -1.

[17] To spell out further details would require the introduction of frames and physical meaning for the involved rotations. Leaving out these details instead stresses the independence of our similarity arguments.

Table 5.2.: Comparing homomorphic QM-conventions

Expression =	**Equal Expression Given QM-Convention**	
$(\mathbf{C}, \star) =$	$(\mathbf{C}_H, \odot)$	$(\mathbf{C}_S, \otimes)$
$\mathbf{C}(\mathbf{p})\mathbf{C}(\mathbf{q}) =$	$\mathbf{C}_H(\mathbf{p} \odot \mathbf{q})$	$\mathbf{C}_S(\mathbf{p} \otimes \mathbf{q})$
$\mathbf{C}(\mathbf{q})\mathbf{x} =$	$\mathscr{I}(\mathbf{q} \star \mathscr{I}^*(\mathbf{x}) \star \mathbf{q}^{-1})$	
$\mathscr{I}(\mathbf{p} \star \mathbf{q}) =$	$p_r \overrightarrow{\mathbf{q}} + q_r \overrightarrow{\mathbf{p}} + \overrightarrow{\mathbf{p}} \times \overrightarrow{\mathbf{q}}$	$p_r \overrightarrow{\mathbf{q}} + q_r \overrightarrow{\mathbf{p}} - \overrightarrow{\mathbf{p}} \times \overrightarrow{\mathbf{q}}$
$\mathbf{i} \star \mathbf{j} =$	$+\mathbf{k}$	$-\mathbf{k}$
$\mathbf{C}(\mathbf{q}) =$	$q_r^2 \mathbf{1} + 2 q_r \overrightarrow{\mathbf{q}}^{\times} + \mathbf{Q}(\mathbf{q})$ [1]	$q_r^2 \mathbf{1} - 2 q_r \overrightarrow{\mathbf{q}}^{\times} + \mathbf{Q}(\mathbf{q})$ [1]
$\mathbf{C}(\boldsymbol{\phi}) =$	$\exp(+\alpha_{\mathbf{C}} \alpha_{\boldsymbol{\phi}} \boldsymbol{\phi}^{\times})$	
$\Rightarrow$[2] $\mathbf{q}(\boldsymbol{\phi}) =$	$\pm \exp(+\alpha_{\mathbf{C}} \alpha_{\boldsymbol{\phi}} \boldsymbol{\phi}/2)$	$\pm \exp(-\alpha_{\mathbf{C}} \alpha_{\boldsymbol{\phi}} \boldsymbol{\phi}/2)$
$\alpha_{\boldsymbol{\omega}} \alpha_{\mathbf{C}} \dot{\mathbf{C}} =$	$+\mathbf{C} \boldsymbol{\omega}_A{}^{\times}$ $+\boldsymbol{\omega}_B{}^{\times} \mathbf{C}$	
$\Rightarrow$[2] $\alpha_{\boldsymbol{\omega}} \alpha_{\mathbf{C}} \dot{\mathbf{q}} =$	$+\frac{1}{2} \mathbf{q} \odot \mathscr{I}^*(\boldsymbol{\omega}_A)$ $+\frac{1}{2} \mathscr{I}^*(\boldsymbol{\omega}_B) \odot \mathbf{q}$	$-\frac{1}{2} \mathbf{q} \otimes \mathscr{I}^*(\boldsymbol{\omega}_A)$ $-\frac{1}{2} \mathscr{I}^*(\boldsymbol{\omega}_B) \otimes \mathbf{q}$

[1] $\mathbf{Q}(\mathbf{q}) := (\overrightarrow{\mathbf{q}}^{\times})^2 + \overrightarrow{\mathbf{q}} \, \overrightarrow{\mathbf{q}}^T$.

[2] $\mathbf{C}(\boldsymbol{\phi}), \mathbf{q}(\boldsymbol{\phi})$ and $\mathbf{C}(t), \mathbf{q}(t)$ are coupled here through $\mathbf{C}(\mathbf{q})$.

table clearly shows that $(\mathbf{C}_H, \odot)$ yields more formal similarity with the rotation matrices when related to rotation vectors or angular velocities: The alternative, $(\mathbf{C}_S, \otimes)$, introduces a difference in sign for the expressions involving $\boldsymbol{\phi}$ or $\boldsymbol{\omega}$ ($-$ instead of $+$) compared to the matrix cases. This difference in similarity with the corresponding rotation matrix formula does not depend on the interpretation/usage (encoded in $\alpha_{\mathbf{C}}$, $\alpha_{\boldsymbol{\phi}}$, A, B) but only on the choice of the homomorphic QM-convention.

Table 5.2 also indicates that additionally flipping the cross product (negating the $\cdot^{\times}$) would change the similarity argument in favour of $(\mathbf{C}_S, \otimes)$. In fact, it seems probable that this lack of formal similarity was one reason to introduce the matrix valued operator $[[\boldsymbol{\phi}]] := -\boldsymbol{\phi}^{\times}$ in [146, p. 445]. Unfortunately, this is hard to verify because it was introduced only on the grounds that this "matrix turns out to be more convenient overall". In [147], this matrix was abandoned again.

Formal Comparison Distinguishing Active and Passive Usage

A further symmetry aspect in favor of Hamilton's multiplication becomes apparent if we compare both active and passive usage of matrices and quaternions (see Section 3 and Figure 5.1). This yields Table 5.3, again using the factor $\alpha_{\boldsymbol{\phi}}$ to encode the usage of a given rotation vector $\boldsymbol{\phi}$. In contrast to Table 5.2, which assumes an arbitrary but single usage for rotation matrices and quaternions, it compares both usages (corresponding values in columns 2 and 3). Each row corresponds to a rotation representation: rotation matrix and three different flavours of rotation quaternions, each yielding homomorphic conversion to the rotation matrix in Row 1. The table shows that neither using Shuster's multiplication for the passive and Hamilton's for active usage (Row 3), nor a full switch to Shuster's multiplication (Row 4) can yield the same level of form-similarity, when compared to the rotation matrices, as our proposed option, using $\odot$ in all cases (Row 2). Similarity flaws are marked in red vs. green. A graphical illustration of the comparison between the pure Hamilton and the mixed multiplication scenarios is depicted in Figure 5.1.

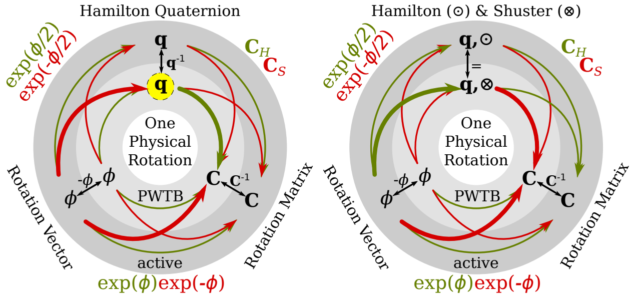


Figure 5.1.: Active (or passive body-to-world; outer ring) and passive world-to-body (=:PWTB; inner ring) representations of a single physical rotation with the three common parametrizations: rotation matrix, **C**, rotation vector $\boldsymbol{\phi}$, and rotation quaternion **q**. Their mutual conversions are represented by arrows pointing to the codomain of the associated expression/function name (nearby and the same color). **On the left, Hamilton multiplication only**: Black arrows represent conversion between active and PWTB usage while keeping the parametrization, green arrows represent conversion between different parametrizations while keeping the usage, and red arrows represent additional switching of the usage. The bold arrows constitute the homomorphic convention suggested in this paper. It is suitable for everybody preferring PWTB for **q** and **C**. The rare users of PWTB rotation vectors would instead entirely remain in the inner circle. Clinging to the green conversion from the common active $\boldsymbol{\phi}$ to **q** even if **q** is intended to be employed PWTB, as done in [146] leads to the situation **on the right, using Shuster's multiplication for quaternions used passively**. The colors are kept for conversion functions that also appear on the left (all except the identity conversion between Hamilton and Shuster quaternions). This conventional branch neglects the option of the highlighted inverted **q** (on the left) and thus requires the flipped multiplication, $\otimes$, and inevitably breaks the symmetry between **C** and **q** on the right compared to the left, i.e., there is no coloring of conversion functions that achieves the same level of symmetry.

7. Recipes

In this section we give some practical advice to a) identify QM-conventions and b) migrate between them.

Table 5.3.: Active vs. passive usage given a rotation vector $\boldsymbol{\phi}$.

Rot. rep.	Active	Passive	Composition
1) $\mathbf{C}(\boldsymbol{\phi}) =$	$\exp(+\alpha_{\boldsymbol{\phi}}\boldsymbol{\phi}^{\times})$	$\exp(-\alpha_{\boldsymbol{\phi}}\boldsymbol{\phi}^{\times})$	$\cdot$
Homomorphic rotation quaternion options:			
2) $\mathbf{q}(\boldsymbol{\phi}) =$	$\exp(+\alpha_{\boldsymbol{\phi}}\boldsymbol{\phi}/2)$	$\exp(-\alpha_{\boldsymbol{\phi}}\boldsymbol{\phi}/2)$	$\odot$
3) $\mathbf{q}(\boldsymbol{\phi}) =$	$\exp(+\alpha_{\boldsymbol{\phi}}\boldsymbol{\phi}/2)$	$\exp(+\alpha_{\boldsymbol{\phi}}\boldsymbol{\phi}/2)$	$\odot$(act), $\otimes$(pass)
4) $\mathbf{q}(\boldsymbol{\phi}) =$	$\exp(-\alpha_{\boldsymbol{\phi}}\boldsymbol{\phi}/2)$	$\exp(+\alpha_{\boldsymbol{\phi}}\boldsymbol{\phi}/2)$	$\otimes$

7.1. How to detect which QM-convention is used

QM-conventions are fully determined by two binary choices: the quaternion to matrix map, $\mathbf{C}$, and the quaternion multiplication, $\star$. These choices can be ascertained as follows:

For the quaternion multiplication

one of the easiest ways is to find out the result of the product $\mathbf{i} \star \mathbf{j}$. If the result is $\mathbf{k}$ then $\star = \odot$ otherwise it is $-\mathbf{k}$ and $\star = \otimes$.

The quaternion to matrix conversion

can be identified by applying it to a suitable test quaternion, e.g. $\mathbf{q}_T := \sqrt{0.5}(\mathbf{1} + \mathbf{k})$. It holds $\mathbf{C}_H(\mathbf{q}_T) = \mathbf{C}_T := \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ and $\mathbf{C}_S(\mathbf{q}_T) = \mathbf{C}_T^T$. This test can also be performed on a **matrix to quaternion conversion**. It maps $\mathbf{C}_T$ to $\pm\mathbf{q}_T$ if $\mathbf{C}_H$ is used and to $\pm\overline{\mathbf{q}_T}$ in case of $\mathbf{C}_S$.

7.2. Migrating from one QM-convention to another

There are several efficient ways to *migrate* a *tool* (e.g. formulas, proofs, publications or algorithms and implementations) between QM-conventions, where migration shall refer to a transformation after which the tool does exactly the same job but with all quaternions going in or out being compatible with the target convention. We introduce two powerful migration procedures, *translate* and *interface*, that are applicable to all types of tools. Both procedures migrate between the two homomorphic QM-conventions[18]. It can be very helpful to first decompose a tool into smaller *sub-tools* and migrate each with the most advantageous procedure.

The two migration procedures

For both procedures a successive mathematical simplification step is recommended.

---

[18] To migrate a non homomorphic QM-convention first replace all multiplications with the other and flip their arguments. This is an equivalence transformation yielding a tool using a homomorphic convention.

Translate Replace all quaternion **constants**, $\mathbf{c} \in \mathbb{H}$, within the tool with their **conjugated** value, $\bar{\mathbf{c}}$, and all quaternion **multiplications** with their **flipped** version ($\otimes \leftrightarrow \odot$).

Interface **Conjugate** all **quaternion** valued **in-** and **outputs** of the tool[19]. Components of a *partial* quaternion (i.e. not constituting a complete quaternion) going in or out should be treated as plain real numbers.

Examples

Next, we provide some examples:

$\mathbf{i}\otimes\mathbf{j} = -\mathbf{k}$ translates into $-\mathbf{i}\odot-\mathbf{j} = -(-\mathbf{k})$, which simplifies to $\mathbf{i}\odot\mathbf{j} = \mathbf{k}$. Interfacing it would not change it since it has no inputs and only a logic output.

$\mathscr{I}(\mathbf{q})$ interfaces into $\mathscr{I}(\bar{\mathbf{q}}) = -\mathscr{I}(\mathbf{q})$. When translating it, its implicit dependence on constants must be respected: The right hand side of $\mathscr{I}(\mathbf{q}) = (\langle\mathbf{q},\mathbf{i}\rangle, \langle\mathbf{q},\mathbf{j}\rangle, \langle\mathbf{q},\mathbf{k}\rangle)$ translates into $(\langle\mathbf{q},-\mathbf{i}\rangle, \langle\mathbf{q},-\mathbf{j}\rangle, \langle\mathbf{q},-\mathbf{k}\rangle) = -\mathscr{I}(\mathbf{q})$.

$\dot{\mathbf{q}} = -\frac{1}{2}\mathbf{q}\otimes\mathscr{I}^*(\boldsymbol{\omega})$ could be first decomposed in $\dot{\mathbf{q}} = -\frac{1}{2}\mathbf{q}\otimes\cdot$ and $\mathscr{I}^*(\boldsymbol{\omega})$. The first translates into $\dot{\mathbf{q}} = -\frac{1}{2}\mathbf{q}\odot\cdot$[20]. The second interfaces into $\overline{\mathscr{I}^*(\boldsymbol{\omega})} = -\mathscr{I}^*(\boldsymbol{\omega})$. Putting together yields the expected $\dot{\mathbf{q}} = \frac{1}{2}\mathbf{q}\odot\mathscr{I}^*(\boldsymbol{\omega})$.

$\mathbf{C}_S(\mathbf{q})$ interfaces into $\mathbf{C}_S(\bar{\mathbf{q}}) = \mathbf{C}_H(\mathbf{q})$[21]. It translates into the same, because $\mathbf{C}_S$ must use coordinates with respect to a basis $(\mathbf{q}_i)_1^4$ (constants) for its input and $\sum\alpha_i\bar{\mathbf{q}}_i = \mathbf{q} \Leftrightarrow \sum\alpha_i\mathbf{q}_i = \bar{\mathbf{q}}$, because $\alpha_i \in \mathbb{R}$ (compare 7.2).

$\mathscr{I}(\mathbf{q}\otimes\mathscr{I}^*(\mathbf{x})\otimes\mathbf{q}^{-1})$ could be first decomposed in $\mathscr{I}(\cdot)$, interfacing into $\mathscr{I}(\bar{\cdot}) = -\mathscr{I}(\cdot)$, and $\mathbf{q}\otimes\mathscr{I}^*(\mathbf{x})\otimes\mathbf{q}^{-1}$, translating into $\mathbf{q}\odot(-$[22]$\mathscr{I}^*(\mathbf{x}))\odot\mathbf{q}^{-1}$. Putting together yields the expected result, $\mathscr{I}(\mathbf{q}\odot\mathscr{I}^*(\mathbf{x})\odot\mathbf{q}^{-1})$.

Correctness of the migration recipes

It remains the question why these two migration concepts work. Unfortunately being intuitively convinced of their correctness is much easier than to proof them. Formal proofs are far beyond this paper. A rather detailed sketch can be found in our extended version [151]. Here we only provide the proof ideas:

---

[19] Also suggested in [122].

[20] The variable $\mathbf{q}$ is an input and no constant.

[21] Hence, the matrix part of the QM-convention is indeed migrated.

[22] From translating constants in $\mathscr{I}^*(\mathbf{x}) = x_1\mathbf{i} + x_2\mathbf{j} + x_3\mathbf{k}$

Interfacing The *interfacing* strategy must yield a corresponding tool, i.e. having the same input output behavior except for quaternion inputs and outputs, which are conjugated, because we can interface it twice and yield the same tool as before (conjugation and identity are self inverse). From this it follows immediately that the interfacing is loss and error free. The correspondence follows the interfacing itself which is designed to achieve corresponding input output behavior.

Translation The *translation* strategy must yield a corresponding tool essentially because it represents precisely the application of an isomorphism, namely quaternion conjugation, between the isomorphic algebraic structures $(\mathbb{H}, +, \odot, \bar{\cdot}, \langle\cdot,\cdot\rangle)$ and $(\mathbb{H}, +, \otimes, \bar{\cdot}, \langle\cdot,\cdot\rangle)$. Since this isomorphism also does not alter the real numbers embedded in $\mathbb{H}$ (as linear span of $\{\mathbf{1}\}$), the input output behavior is affected precisely as hoped. This can be seen as follows: First, there is no other way to exchange information with the "quaternion world" than through quaternions or real numbers because the axiomatic properties do not provide any other possibility. Second, all quaternions get conjugated and real numbers remain the same if we apply conjugation to all constants and inputs, which corresponds to using the translated tool with the target convention.

8. Conclusion

In this paper, we suggested an alternative solution to the antihomomorphy problem that led to the introduction of the flipped quaternion multiplication. We further proposed to discontinue its use in favour of Hamilton's original definition combined with the suggested alternative solution. To argue for this we gave evidence of the cost of maintaining two multiplications and showed that for principal reasons there cannot be any significant capability benefit in using the flipped multiplication for theory or algorithms. Additionally, we demonstrated that the formal similarity between matrices and quaternions, when related to angular velocity or rotation vectors, is in favour of the Hamiltonian multiplication independently of any other conventional decision as long as the definition of the cross product is not additionally questioned. Furthermore, we provided recipes for how to migrate formulas and algorithms from one quaternion multiplication to the other as well as how to detect which convention is used in a given context.

Paper III

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

Hannes Sommer, James Richard Forbes, Roland Siegwart, Paul Furgale

Abstract

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 unit-length quaternions. It extends existing theory of unit-length quaternion B-splines to general Lie groups and arbitrary B-spline order. It is show how to use these curves for continuous-time batch estimation using Gauss-Newton or Levenberg-Marquardt, including efficient curve initialization, a parameter update step that preserves the Lie group constraint within an unconstrained optimization framework, and the derivation of Jacobians of the B-spline's value and its time derivatives with respect to an update of its parameters. For unit-length quaternion splines, the equations for angular velocity, and angular acceleration are derived. An implementation of this algorithm is evaluated on two problems: spacecraft attitude estimation using a three-axis magnetometer, a sun sensor, and (i) a three-axis gyroscope, and (ii) a continuous-time vehicle dynamics model based on Euler's equation. Its performance is compared against a standard multiplicative extended Kalman filter (MEKF) 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.

Published in:
Journal of Guidance, Control, and Dynamics, 2015
doi:10.2514/1.G001149

Nomenclature

$\|\cdot\|$ = the Euclidean 2-norm in $\mathbb{R}^n$, for any $n \in \mathbb{N}$
$\mathscr{A}$ = associative algebra and ambient space of $\mathscr{G}$
$\mathscr{A}_{\text{inv}}$ = the group of multiplicative invertible elements in $\mathscr{A}$
$\boldsymbol{\iota}$ = multiplicative identity element
$\mathscr{G}$ = general Lie group
$\mathfrak{g}$ = $\mathscr{G}$'s Lie algebra
$\mathbf{g}$ = an element of $\mathscr{G}$
$\mathbb{H}$ = the skew field of quaternions
I = the number of knots in the B-spline
K = $I - O$ = the number of control vertices in the B-spline
O = B-spline order
$\mathbf{q}$ = an unit-length quaternion
$\mathbb{R}$ = the field of real numbers
$\mathrm{Rot}_{\mathbf{q}}$ = rotation associated with $\mathbf{q}$
T = the end time
$\mathscr{T}$ = the time range $[0, T]$

1. 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. [153] 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 [97], and incremental batch estimation [82]. The benefits of treating data in batch were clearly shown by Vandersteen et al. [158] 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. [59] 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 [161, pp. 426–428], the QUEST algorithm [148], and the more recent ESOQ [116] and ESOQ2 [117], 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 [51], 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 [24, 41, 95, 145]. The question is no less important in continuous-time, where we must decide how to parameterize a continuously time-varying orientation. Furgale et al. [59] use the simple approach of defining a 3×1 B-spline of Cayley-Gibbs-Rodrigues parameters [30]. The B-spline provides analytical formulas for parameter rates, allowing the computation of angular velocity at any point [77, p. 30–31, Table 2.3].

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. [86] 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. [59] with the one proposed in Kim et al. [86]. To do so, we must further develop the theory presented in Kim et al. [86] 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. [86] 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. [158].

2. 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. [153] 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. [97] 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. [118] 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 [99].

These "online batch" algorithms follow the established formula for continuous-time filtering with discrete measurements first proposed by Moore and Tam [115]. 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 [128] and references therein), the processing of data from rolling shutter cameras [72, 123, 137], or continuously-moving sweeping laser scanners [22, 32, 47, 155], an estimate of the state is required at such a high rate that a discrete-time formulation becomes intractable for all but the smallest problems. 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 [22, 32, 47, 123, 128, 137] or Gaussian process (GP) regression [155]. 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. [156] 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. [59] showed how it relates back to continuous-time maximum likelihood estimation [79, Chapter 5], 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. [123] and Furgale et al. [58]).

Continuous-time representation of attitude has mostly used either curves through parameter space [58, 59, 123, 128] or spherical linear interpolation on SO(3) [47, 72, 137]. 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 [22] 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. [86] 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. [101], 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 [76] 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 [161, pp. 426–428] and QUEST [148], are solutions to Wahba's problem [160]. The q-Method and QUEST, as well as ESOQ [116] and ESOQ2 [117], employ unit-length quaternions when estimating attitude. Other solutions to Wahba's problem, such as Farrel et al. [50], Forbes and de Ruiter [54], Markley [105], estimate the attitude rotation matrix rather than an attitude parameterization. Wahba's problem in SO(n) is considered in de Ruiter and Forbes [45]. There is a vast literature devoted to spacecraft attitude estimation using Kalman-type filters such as the EKF and the UKF [42]. Unit-length quaternions are the preferred attitude description when Kalman filtering due to the absence of singularities; for example, see Bar-Itzhack and Oshman [24], Bar-Itzhack and Reiner [25], Choukroun et al. [37], Crassidis and Markley [41], Lefferts et al. [95], Shuster [145]. Often a three-parameter attitude representation, such as Gibbs parameters, are used together with a quaternion in a multiplicative framework [41, 106, 107]. Rather than trying to circumvent various issues associated the unit-length quaternion constraint within multiplicative filtering structure, Chee and Forbes [35], Zanetti et al. [163] revisit the derivation of the discrete-time Kalman 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. [55] presents a continuous-time generalization of Zanetti et al. [163]. Various authors have also considered estimating the rotation matrix directly in both stochastic and deterministic settings. Bar-Itzhack and Reiner [25], Choukroun et al. [38, 39] consider rotation matrix estimation within a Kalman filtering framework. Firoozi and Namvar [52], Khosravian and Namvar [85] consider rotation matrix estimation of a spacecraft endowed with a rate gyro and one vector measurement, generalizing the SO(3) estimator structure presented in [67, 80, 102, 103]. Similar rotation matrix estimators can be found in Grip et al. [63], Kinsey and Whitcomb [88].

The closest work to our is that of Vandersteen et al. [158] 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.

3. 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 a thorough introduction to B-splines, see Bartels et al. [28, chap. 4]. For a brief overview with a focus on estimation, see Furgale et al. [59]. For a thorough introduction to Lie groups, see Kirillov et al. [89, chap. 2–3], and for a more applied approach, see Hall [66, chap. 1–3].

3.1. Continuous-Time Batch State Estimation

This section will briefly review the continuous-time batch state estimation framework used in this paper and originally presented in Furgale et al. [59]. The probabilistic derivation for estimation of a continuous-time process, $\mathbf{x}(t)$, with discrete measurements seeks an estimate of the joint posterior density, $p\left(\mathbf{x}(t)|\mathbf{u}(t),\mathbf{z}_{1:N}\right)$, over the interval, $\mathscr{T} = [0,T]$, where $\mathbf{u}(t)$ is a control input to the system, and $\mathbf{z}_i$ is a measurement taken at time $t_i \in \mathscr{T}$, $1 \le i \le N$. Assuming the probability density of the initial state, $p\left(\mathbf{x}\left(t_0\right)\right)$, is known, this density is commonly factored as,

$$p\left(\mathbf{x}(t)|\mathbf{u}(t),\mathbf{z}_{1:N}\right) = \frac{p\left(\mathbf{x}(t)|\mathbf{u}(t)\right)\prod_{i=1}^{N} p\left(\mathbf{z}_i|\mathbf{x}(t_i)\right)}{p\left(\mathbf{z}_{1:N}\right)}, \tag{6.1}$$

by using Bayes' rule and assuming that the measurements' distributions given the state (a) do not depend on the control inputs and (b) are mutually independent.

By assuming that the densities involved are Gaussian, we can write the measurement model,

$$\mathbf{z}_i = \mathbf{h}_i\left(\mathbf{x}(t)\right) + \mathbf{n}_i, \quad \mathbf{n}_i \sim \mathscr{N}\left(\mathbf{0},\mathbf{R}_i\right), \tag{6.2}$$

and measurement distribution,

$$p\left(\mathbf{z}_i|\mathbf{x}(t)\right) = \mathcal{N}\left(\mathbf{h}_i\big(\mathbf{x}(t)\big), \mathbf{R}_i\right), \tag{6.3}$$

where i refers to the ith measurement, $\mathbf{h}_i(\cdot)$ is a nonlinear measurement model, typically evaluating $\mathbf{x}$ and its derivatives at a measurement time t_i, and $\mathbf{R}_i$ is the covariance matrix of the measurement noise.

We may also write down the process model as a continuous stochastic dynamical system [79, p. 143] described formally by the differential equation,

$$\dot{\mathbf{x}}(t) = \mathbf{f}\big(\mathbf{x}(t), \mathbf{u}(t)\big) + \mathbf{w}(t), \quad \mathbf{w}(t) \sim \mathcal{GP}\left(\mathbf{0}, \mathbf{Q}\,\delta(t-t')\right), \tag{6.4}$$

where the over-dot represents the time derivative, $\mathbf{f}(\cdot)$ is a deterministic function, $\mathbf{w}(t)$ is a zero-mean, white Gaussian process with power-spectral-density matrix $\mathbf{Q}$, and $\delta(\cdot)$ is Dirac's delta function. The notation $\mathcal{GP}(\cdot)$ denotes a Gaussian process in the same way that $\mathcal{N}(\cdot)$ denotes a Gaussian distribution [134, Chapter 2]. The process distribution may then be written as [79, p. 156]:

$$p\big(\mathbf{x}(t)|\mathbf{u}(t)\big) \propto p\big(\mathbf{x}(t_0)\big) \exp\left(-\frac{1}{2}\int_0^T \mathbf{e}_u(\tau)^T \mathbf{Q}^{-1} \mathbf{e}_u(\tau)\, d\tau\right), \tag{6.5}$$

where

$$\mathbf{e}_u(t) := \dot{\mathbf{x}}(t) - \mathbf{f}\big(\mathbf{x}(t), \mathbf{u}(t)\big). \tag{6.6}$$

A batch estimator for $\mathbf{x}(t)$ may be derived by taking the negative logarithm of the posterior likelihood,

$$\{\mathbf{x}(t)^\star\} = \underset{\mathbf{x}(t)}{\operatorname{argmin}} \left(-\log\left(p(\mathbf{x}(t)|\mathbf{u}(t), \mathbf{z}_{1:N}\right)\right). \tag{6.7}$$

The key innovation presented in Furgale et al. [59] was to leave the problem in continuous-time and make estimation of (6.7) tractable by modeling the process as a weighted sum of a finite number of known analytical basis functions,

$$\boldsymbol{\Phi}(t) := \begin{bmatrix} \boldsymbol{\phi}_1(t) & \dots & \boldsymbol{\phi}_M(t) \end{bmatrix}, \quad \mathbf{x}(t) := \boldsymbol{\Phi}(t)\mathbf{c}, \tag{6.8}$$

where each $\boldsymbol{\phi}_m(t)$ is a $D \times 1$ basis function, $\boldsymbol{\Phi}(t)$ is a $D \times M$ stacked basis matrix, and $\mathbf{c}$ is a $M \times 1$ column of coefficients. Hence, by making the substitution in (6.8), we have turned a difficult problem—estimate a process at the infinite number of points in $\mathcal{T}$—to a simpler problem that we know how to solve: estimate the column of coefficients, $\mathbf{c}$. We call this, the basis function formulation,

$$\{\mathbf{c}^\star\} = \underset{\mathbf{c}}{\operatorname{argmin}} \left(-\log\left(p(\mathbf{x}(t)|\mathbf{u}(t), \mathbf{z}_{1:N}\right)\right) \approx \underset{\mathbf{x}(t)}{\operatorname{argmin}} \left(-\log\left(p(\mathbf{x}(t)|\mathbf{u}(t), \mathbf{z}_{1:N}\right)\right), \tag{6.9}$$

where the approximation sign indicates that we are approximating the process $\mathbf{x}(t)$ with (6.8). This problem is easily solved using standard methods like Gauss-Newton or Levenberg-Marquardt.

The basis function formulation works very well when the process lives in a vector space.

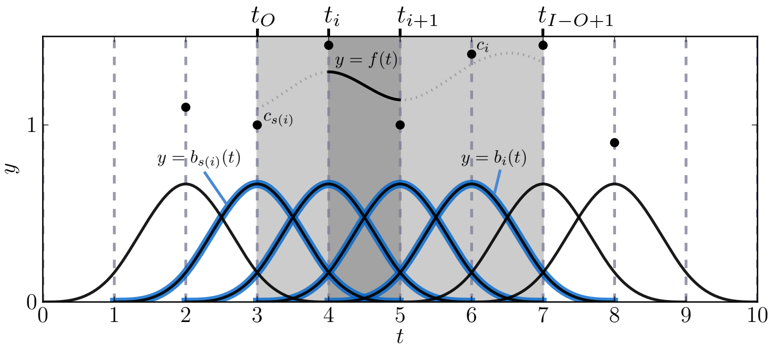


Figure 6.1.: Graph of a B-spline $f : [t_O, t_{I-O+1}) \to \mathbb{R}$ (with $O = 4$, $I = 11$, $t_j := j-1$). It shows basis functions, b_j (solid bumps), knots, t_j, (dashed lines), control vertices, $\mathbf{c}_j$, (black dots). Only the highlighted basis functions contribute to f's value in the dark gray shaded area.

However, this does not cover common cases in spacecraft attitude estimation or robotics, where vehicle states usually are of non-vector space type like the Lie groups SO(3) or SE(3). The next section derives one approach to address this issue.

3.2. Construction of a Lie group-valued B-spline

In this section, we derive a method to construct a B-spline curve on a finite-dimensional Lie group. Our derivation is based on the unit-length quaternion B-spline curve from Kim et al. [86]. 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. The derivation presented below is valid for any nondecreasing knot sequence.

We will quickly motivate that alternative form. Let $\mathbf{f}$ be a B-spline function of order O defined on a nondecreasing sequence of knots, $(t_i)_{i=1}^{I}$ with $I \geq 2O$. At time $t \in [t_i, t_{i+1})$, $\mathbf{f}$, when $i \geq O$ and $i \leq K := I - O$, may be written as

$$\mathbf{f}(t) = \sum_{j=s(i)}^{i} b_j(t)\mathbf{c}_j, \tag{6.10}$$

where $s(i) := i - (O-1)$, $(\mathbf{c}_j)_{j=1}^{K} \in \mathscr{V}$ denotes the control vertices in an $\mathbb{R}$-vector space $\mathscr{V}$, and $b_j := B_{j,O}$ denotes the j-th B-spline basis function, where $B_{j,O}$ is defined as in [86] or de Boor [44, chap. IX]. The B-spline order, O, is precisely the spline's polynomial degree plus one. This construction, 6.10, is illustrated in Figure 6.1 for the case $\dim \mathscr{V} = 1$ and $O = 4$. For $t < t_O$ and $t \geq t_{I-O+1}$ (outside the gray shaded area in Figure 6.1), there are fewer than O nonzero basis functions and we treat the B-spline as undefined and assume for the rest of the paper that

$t \in [t_O, t_{I-O+1})$. This expression may be rearranged into the cumulative form as follows

$$\mathbf{f}(t) = \mathbf{c}_{s(i)} + \sum_{j=s(i)+1}^{i} \left(\sum_{k=j}^{i} b_k(t) \right) (\mathbf{c}_j - \mathbf{c}_{j-1}) = \mathbf{c}_{s(i)} + \sum_{j=1}^{O-1} \beta_{i,j}(t)(\mathbf{c}_{s(i)+j} - \mathbf{c}_{s(i)+j-1}), \quad (6.11)$$

defining the cumulative basis functions $\beta_{i,j}(t) := \sum_{k=s(i)+j}^{i} b_k(t)$ for $1 \leq j \leq O-1$. They are related to the notation used in Kim et al. [86] according to $\beta_{i,j}(t) = \tilde{B}_{s(i)+j,O}(t)$ but only for $t \leq t_{i+1}$. Hence, they share for this t range also their derivatives : $\frac{\mathrm{d}}{\mathrm{d}t}\beta_{i,j}(t) = \frac{O-1}{t_{i+j} - t_{s(i)+j}} B_{s(i)+j,O-1}(t)$.

Note that the assumption $t \in [t_O, t_{I-O+1})$ (or equivalent $O \leq i \leq I-O$) is crucial for (6.10) and (6.11) to hold in the given form. This assumption contrasts Kim et al. [86] where the first summand in (6.11) does not depend on t for any admissible t. For (10) and (11) to hold without this assumption the weight $\sum_{j=1}^{\min(i,I-O)} b_j(t)$ is needed and this weight is 1 if and only if $i \geq O$ given that $t \in [t_i, t_{i+1})$. The fact that this weight would prevent the B-spline from having the bi-equivariance property proven in section 3.5 is an important reason for this assumption.

We will now generalize this form (6.11) to the Lie group, $\mathscr{G}$, just as Kim et al. [86] do it for unit-length quaternions. Let $(\mathscr{G}, \cdot)$ be a connected finite dimensional Lie group, with Lie algebra $\mathfrak{g}$. Let $\boldsymbol{\iota} \in \mathscr{G}$ 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, $\mathbf{g}_i \in \mathscr{G}$. 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 $(\mathbf{g}_{k-1}^{-1}\mathbf{g}_k)$. 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 $\mathscr{G}$. This way one gets the generalized form of (6.11) as

$$\mathbf{g}(t) := \mathbf{g}_{s(i)} \prod_{j=1}^{O-1} \mathbf{r}_{i,j}(t), \text{ where} \quad (6.12)$$

$$\mathbf{r}_{i,j}(t) := \exp\left(\beta_{i,j}(t)\boldsymbol{\varphi}_{s(i)+j}\right), \text{ and } \boldsymbol{\varphi}_k := \log\left(\mathbf{g}_{k-1}^{-1}\mathbf{g}_k\right). \quad (6.13)$$

This construction requires that the log function is defined on $\mathbf{g}_{k-1}^{-1}\mathbf{g}_k$. To guarantee this, one has to make sure that there are enough intermediate points to have all $\mathbf{g}_{k-1}^{-1}\mathbf{g}_k$ close enough to identity. In a connected Lie group, this should always be possible with finite many $\mathbf{g}_i$. Note that the vector space construction, (6.11), is equivalent to the Lie group approach, (3.2), when $(\mathscr{G}, \cdot)$ is chosen as the additive group $(\mathscr{V}, +)$ of the vector space $\mathscr{V}$, 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.

3.3. Theoretical preparations

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

The ambient algebra assumption

Assumption 1. *We assume the Lie group* $(\mathscr{G}, \cdot)$ *to be an embedded differential manifold of an unital associative* $\mathbb{R}$*-algebra* $(\mathscr{A}, +, \cdot)$ [1] *and a sub group of* $(\mathscr{A}_{inv}, \cdot)$*, the group of multiplicative invertible elements in* $\mathscr{A}$*.*

Assumption 1 allows us to take derivatives of curves through $\mathscr{G}$ in the ambient algebra $\mathscr{A}$, where we can easily use the product rule and benefit from a single global tangent space. All real matrix Lie groups, the unit-length quaternions, the unit-magnitude dual quaternions, and the unit-length complex numbers 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 $\mathscr{G}$ and the quaternions as $\mathscr{A}$, or the group SO(3) as $\mathscr{G}$ and $\mathbb{R}^{3\times 3}$ as $\mathscr{A}$.

Matrix isomorphy

Given a vector $\mathbf{a} \in \mathscr{A}$, we will use special operators to retrieve the matrices that represent the left, $\mathbf{a}^L$, or right, $\mathbf{a}^R$, multiplication by $\mathbf{a}$ (with respect to a fixed default basis for $\mathscr{A}$), that is, $\Psi(\mathbf{ab}) = \mathbf{a}^L\Psi(\mathbf{b}) = \mathbf{b}^R\Psi(\mathbf{a})$, for all $\mathbf{b} \in \mathscr{A}$, where $\Psi : \mathscr{A} \to \mathbb{R}^{\dim \mathscr{A}}$ maps the vectors to their coordinate tuples. In the following we will implicitly identify $\mathscr{A}$-vectors with their coordinates. These operators, restricted to $\mathscr{G}$, are injective Lie group-homomorphism, $(\cdot)^L$, and antihomomorphism [2], $(\cdot)^R$, from $\mathscr{G}$ into $\mathrm{GL}(\mathscr{A})$, the General Linear group over $\mathscr{A}$. Hence, these operators give us the opportunity to reduce computational complexity of Jacobian matrix evaluation by choosing to apply the multiplication to elements of $\mathscr{G}$ rather than $\mathscr{G}^L$ or $\mathscr{G}^R$. The mapping $(\cdot)^L$ also leads to the following theorem.

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

We will use this theorem to import knowledge about matrix Lie groups to our general setting. It also tells us that Assumption 1 restricts us—up to isomorphism—precisely to $\mathbb{R}$-matrix Lie groups. 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.

The exponential maps

In both $\mathscr{A}$ and $\mathscr{G}$ there is a canonical definition of an exponential map. For the algebra, $\mathscr{A}$, the exponential map, $\exp_{\mathscr{A}}$, is defined by the usual power series. Applied to matrices, this yields the matrix exponential map. The exponential map, $\exp_{\mathscr{G}}$, of a Lie group, $\mathscr{G}$, is required to be smooth, and, for all $\mathbf{v} \in \mathfrak{g}$, the directional exponential map $\exp_{\mathscr{G}}^{\mathbf{v}} : \mathbb{R} \to \mathscr{G}, t \mapsto \exp_{\mathscr{G}}(t\mathbf{v})$ must fulfill two identities: the classical exponentiation identity, $\exp_{\mathscr{G}}^{\mathbf{v}}(\alpha + \beta) = \exp_{\mathscr{G}}^{\mathbf{v}}(\alpha)\exp_{\mathscr{G}}^{\mathbf{v}}(\beta)$, for all $\alpha, \beta \in \mathbb{R}$, and the differential identity, $\frac{\mathrm{d}}{\mathrm{d}t}\exp_{\mathscr{G}}^{\mathbf{v}}(t)|_{t=0} = \mathbf{v}$.

---

[1] An unital associative algebra is an $\mathbb{R}$-vector space equipped with an $\mathbb{R}$-bi-linear associative vector multiplication and a multiplicative identity element.

[2] 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.

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

We exploit this property to simplify Jacobian calculation by interpreting our B-spline construction (3.2) as construction in $\mathscr{A}$. 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.

3.4. Continuous-Time Batch State Estimation on Lie groups

In the section 3.2 we present a construction for Lie group-valued B-splines. We will now illustrate how they can be exploited to extend continuous-time batch state estimation (see section 3.1) to Lie groups.

To minimize the negative log-likelihood function, L, as given in (6.9), but with $\mathbf{x}(t) := \mathbf{g}(t, (\mathbf{g}_k)_{k=1}^K)$ given as Lie group-valued B-spline, it is necessary to perform a nonlinear minimization involving variables in the Lie group in question, namely the control vertices $(\mathbf{g}_k)_{k=1}^K \in \mathscr{G}$.

In the following formulation we assume L to be given in the form $||\mathbf{e}||^2$, where $\mathbf{e} \in \mathbb{R}^\nu$, $\nu \in \mathbb{N}$. We call $\mathbf{e}$ the error function because it typically consist of the normalized residuals. To implement Gauss-Newton or Levenberg-Marquardt optimization algorithms to minimize the cost function L, we must have access to an expression for the Jacobian of $\mathbf{e}$ with respect to small changes in the estimated variables. For an estimated variable $\mathbf{g} \in \mathscr{G}$ of Lie group type (e.g., one of the control vertices) this requires a generalized Jacobian concept. It is usual to use a *minimal perturbation*, $\boldsymbol{\phi} \in \mathbb{R}^m$ (with $m = \dim \mathscr{G}$), in the Lie algebra, mapped to the Lie group via an exponential chart $\Phi_{\bar{\mathbf{g}}}$ centered at the current guess, $\bar{\mathbf{g}}$.

$$\mathbf{g}(\boldsymbol{\phi}) = \Phi_{\bar{\mathbf{g}}}(\boldsymbol{\phi}) := \exp(\mathscr{B}_{\mathfrak{g}} \boldsymbol{\phi}) \bar{\mathbf{g}}, \tag{6.14}$$

where $\mathscr{B}_{\mathfrak{g}} \boldsymbol{\phi} := \sum_{k=1}^m \boldsymbol{\phi}_k \mathbf{b}_k$ denotes the vector in $\mathfrak{g}$ corresponding to the coordinates, $\boldsymbol{\phi}$, with respect to $\mathscr{B}_{\mathfrak{g}}$, denoting the default basis for $\mathfrak{g}$. This yields the concept of a Jacobian for a function $\mathbf{e} : \mathscr{G} \to \mathbb{R}^\nu$ at $\bar{\mathbf{g}}$ *in the local chart* $\Phi_{\bar{\mathbf{g}}}$, which is defined to be the traditional Jacobian at $\mathbf{0} \in \mathbb{R}^m$ of the composed function $\mathbf{e} \circ \Phi_{\bar{\mathbf{g}}} : \mathbb{R}^m \to \mathbb{R}^\nu$. Unfortunately, while the concept of charts is well known from differential geometry, there seems to be no standard notation for these generalized Jacobians. We will denote it in this paper as $\frac{\partial \mathbf{e}}{\partial \boldsymbol{\phi}}$, while the local chart used is specified based on context, for the sake of brevity.

This generalized Jacobian is used like a normal Jacobian in a Gauss-Newton or Levenberg-Marquardt iteration except that after it produces an answer for $\boldsymbol{\phi}$, the current guess is updated as $\bar{\mathbf{g}} \leftarrow \exp(\mathscr{B}_{\mathfrak{g}} \boldsymbol{\phi}) \bar{\mathbf{g}}$. This update is *constraint sensitive* in that the updated $\bar{\mathbf{g}}$ is guarantied to not leave $\mathscr{G}$, even when a non-minimal representation such as a vector in $\mathscr{A}$ would theoretically allow such a departure.

An outline of the algorithm is provided in Algorithm 1. The umbrella variable $\overline{\mathbf{X}}$ comprises of the current guesses for all the estimated variables. For example the state B-spline's control vertices $(\mathbf{g}_k)_1^K$. In practice it can contain control vertices of multiple splines and other state variables.

Algorithm 1: Continuous-Time Batch State Estimation on Lie groups

```
1 $\mathbf{Z} \leftarrow$ GETMEASUREMENTS
2 $\mathbf{e} \leftarrow$ SETUPERRORFUNCTION($\mathbf{Z}$)
3 $\overline{\mathbf{X}} \leftarrow$ COMPUTEINITIALGUESSES($\mathbf{e}, \mathbf{Z}$)
4 repeat
5     $\mathbf{J} \leftarrow$COMPUTEJACOBIANS($\mathbf{e}, \overline{\mathbf{X}}$)
6     $\mathbf{U} \leftarrow$COMPUTEUPDATES($\mathbf{e}, \overline{\mathbf{X}}, \mathbf{J}, \mathbf{Z}$)    ▷ Gauss-Newton or Levenberg-Marquardt
7     $\overline{\mathbf{X}} \leftarrow \exp(\mathbf{U})\overline{\mathbf{X}}$
8 until HASCONVERGED($\mathbf{e}, \overline{\mathbf{X}}, \mathbf{Z}$)  ▷ Minimize quadratic negative log likelihood function,
    $||\mathbf{e}||^2$
```

The error function $\mathbf{e}(\overline{\mathbf{X}}, \mathbf{Z})$, will typically depend on the state B-spline via its value and derivatives (e.g., via the $\mathbf{h}_i$s in (6.2)), and integrals along the trajectory, (e.g., for (6.5)). Therefor we need analytical expression for the Lie group-valued B-spline's value (section 3.2), its time derivatives (section 3.6), a concept how to evaluate integrals (section 3.8), and generalized Jacobians (line 5 in Algorithm 1), for all involved expressions with respect to changes in its control vertices(section 3.6). To initialize the control vertices (line 3 in Algorithm 1), we also require an initialization strategy (section 3.7).

3.5. Bi-equivariance of this B-spline construction

In this section we will prove that this B-spline construction has an important property for a Lie group-valued parameterized curve when used for physical modeling, called "bi-invariance" by Park and Ravani [124] 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 [124]. 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. First we consider left-equivariance: equivariance with respect to left multiplication by any $\mathbf{u} \in \mathscr{A}_{\text{inv}}$, $L_{\mathbf{u}} : \mathscr{A} \to \mathscr{A}, \mathbf{a} \mapsto \mathbf{ua}$. The expression for $\mathbf{r}_{i,j}$ in (6.13) only depends on the control vertices through expressions like $\mathbf{g}_{k-1}^{-1}\mathbf{g}_k$. Such expressions are invariant with respect to $L_{\mathbf{u}}$, rendering all the $\mathbf{r}$ invariant, too. It follows that

$$\mathbf{g}((L_{\mathbf{u}}(\mathbf{g}_k))_1^K, t) = L_{\mathbf{u}}(\mathbf{g}_{s(i)}) \prod_{j=1}^{O-1} \mathbf{r}_{i,j}(t) = L_{\mathbf{u}}(\mathbf{g}((L_{\mathbf{u}}(\mathbf{g}_k))_1^K, t)) \tag{6.15}$$

Next we consider the right equivariance proof. For $\mathbf{u} \in \mathscr{A}_{\text{inv}}$, the *conjugation* by $\mathbf{u}$, $\pi_{\mathbf{u}} : \mathscr{A} \to \mathscr{A}, \mathbf{a} \mapsto \mathbf{uau}^{-1}$ is an $\mathbb{R}$-algebra automorphism. Therefore any (partial) function of type $\mathscr{A}^k \to \mathscr{A}$ for some $k \in \mathbb{N}$ is automatically equivariant with respect to conjugation by $\mathbf{u}$ if the function is *$\mathscr{A}$-intrinsically defined.* A function from $\mathscr{A}^k \to \mathscr{A}$ is called $\mathscr{A}$-intrinsically defined if and only if its definition is purely based on the structure of $\mathscr{A}$; informally this means that its value and its domain can be defined by only referring to its arguments, real numbers, $\mathbf{0},\mathbf{1} \in \mathscr{A}$, the operations $+$, $*$, the limit in $\mathscr{A}$, the identity relation, set or logical operators, or other $\mathscr{A}$-intrinsically defined functions. This implies that the exp and log mappings on $\mathscr{A}$, and ultimately our whole B-spline construction (3.2), 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, $R_\mathbf{u} : \mathscr{A} \to \mathscr{A}, \mathbf{a} \mapsto \mathbf{au}$, 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 $\mathbf{u} \in \mathscr{A}_{\text{inv}}$, but the transformed spline will remain $\mathscr{G}$-valued if and only if $\mathbf{u} \in \mathscr{G}$. Therefore, in practice, only the equivariances with respect to left or right multiplication by $\mathbf{u} \in \mathscr{G}$ will be of interest.

3.6. Derivatives

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 (3.2) as defined more generally in $\mathscr{A}$.

$$\frac{\mathrm{d}^k}{\mathrm{d}t^k}\mathbf{g}(t) = \frac{\mathrm{d}^k}{\mathrm{d}t^k}\mathbf{g}_{s(i)} \prod_{j=1}^{O-1} \mathbf{r}_{i,j}(t) = k!\, \mathbf{g}_{s(i)} \sum_{\substack{\boldsymbol{\alpha} \in \mathbb{N}^{O-1},\\ \sum \boldsymbol{\alpha} = k}} \prod_{j=1}^{O-1} \frac{1}{\boldsymbol{\alpha}_j!} \frac{\mathrm{d}^{\boldsymbol{\alpha}_j}}{\mathrm{d}t^{\boldsymbol{\alpha}_j}} \mathbf{r}_{i,j}(t) \tag{6.16}$$

To be complete it requires the time derivatives of $\mathbf{r}_{i,j}$ up to order k. We will exploit the directional exponential map's well known differential identity (guaranteed in $\mathscr{A}$ by Theorem 1),

$$\frac{\mathrm{d}}{\mathrm{d}t}\exp^{\mathbf{a}}(t) = \mathbf{a}\exp^{\mathbf{a}}(t), \text{ where } \exp^{\mathbf{a}}(t) := \exp(t\mathbf{a}) \tag{6.17}$$

where $\mathbf{a} \in \mathscr{A}$ and $t \in \mathbb{R}$.

Using Theorem 2 to identify the exp and log in (6.13) with their extensions to $\mathscr{A}$ we get, with $k := s(i) + j$:

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{r}_{i,j}(t) = \frac{\mathrm{d}}{\mathrm{d}t}\exp^{\boldsymbol{\varphi}_k}(\beta_{i,j}(t)) = \beta'_{i,j}(t)\boldsymbol{\varphi}_k \exp^{\boldsymbol{\varphi}_k}(\beta_{i,j}(t)) = \beta'_{i,j}(t)\boldsymbol{\varphi}_k \mathbf{r}_{i,j}(t) \tag{6.18}$$

$$\frac{\mathrm{d}^2}{\mathrm{d}t^2}\mathbf{r}_{i,j}(t) = (\beta'_{i,j}(t)^2\boldsymbol{\varphi}_k + \beta''_{i,j}(t)\boldsymbol{\iota})\boldsymbol{\varphi}_k \mathbf{r}_{i,j}(t) \tag{6.19}$$

$$\frac{\mathrm{d}^3}{\mathrm{d}t^3}\mathbf{r}_{i,j}(t) = \left(\beta'_{i,j}(t)\boldsymbol{\varphi}_k(\beta'_{i,j}(t)^2\boldsymbol{\varphi}_k + \beta''_{i,j}(t)) + 2\beta'_{i,j}(t)\beta''_{i,j}(t)\boldsymbol{\varphi}_k + \beta'''_{i,j}(t)\boldsymbol{\iota}\right)\boldsymbol{\varphi}_k \mathbf{r}_{i,j}(t) \tag{6.20}$$

We will not need higher order derivatives for our experiments. For how to compute the β derivatives see our comment below (6.11) and de Boor [44, chap. X].

Jacobians with respect to changes in the control vertices

In this section we will derive the partial Jacobians of the B-spline's value and derivatives with respect to small changes, $\boldsymbol{\phi}_l$, while considering the control vertices, $\mathbf{g}_l(\boldsymbol{\phi}_l) := \Phi_{\bar{\mathbf{g}}_l}(\boldsymbol{\phi}) = \exp(\mathscr{B}_{\mathfrak{g}}\boldsymbol{\phi})\bar{\mathbf{g}}_l$, as functions of $\boldsymbol{\phi}_k$, as explained in section 3.4. Because the chart map $\Phi_{\bar{\mathbf{g}}}$ is centered at $\bar{\mathbf{g}}$ all Jacobians will be implicitly at $\mathbf{0} \in \mathbb{R}^m$. The value $\in \mathscr{G} \subset \mathscr{A}$ and time derivatives $\in \mathscr{A}$ of the B-spline will be implicitly identified with their coordinates with respect to $\mathscr{A}$'s default basis.

Omitting the time parameter we get for $l \in \{1..K\}$:

$$\frac{\partial \mathbf{g}}{\partial \boldsymbol{\phi}_l} = \frac{\partial}{\partial \boldsymbol{\phi}_l} \mathbf{g}_{s(i)} \prod_{j=1}^{O-1} \mathbf{r}_{i,j} = \left(\prod_{j=1}^{O-1} \mathbf{r}_{i,j}\right)^R \frac{\partial \mathbf{g}_{s(i)}}{\partial \boldsymbol{\phi}_l} + {\mathbf{g}_{s(i)}}^L \sum_{\hat{j}=1}^{O-1} \left(\prod_{j=1}^{\hat{j}-1} \mathbf{r}_{i,j}\right)^L \left(\prod_{j=\hat{j}+1}^{O-1} \mathbf{r}_{i,j}\right)^R \frac{\partial \mathbf{r}_{i,\hat{j}}}{\partial \boldsymbol{\phi}_l} \tag{6.21}$$

For the first summand we get

$$\frac{\partial \mathbf{g}_{s(i)}}{\partial \boldsymbol{\phi}_l} = \frac{\partial}{\partial \boldsymbol{\phi}_l} \exp(\mathscr{B}_{\mathfrak{g}} \boldsymbol{\phi}_{s(i)}) \bar{\mathbf{g}}_{s(i)} = \bar{\mathbf{g}}_{s(i)}^R \frac{\partial}{\partial \boldsymbol{\phi}_l} \exp(\mathscr{B}_{\mathfrak{g}} \boldsymbol{\phi}_{s(i)}) = \delta_{l,s(i)} \bar{\mathbf{g}}_{s(i)}^R \mathbf{V}, \tag{6.22}$$

where δ is the Kronecker delta and $\mathbf{V}$ the Jacobian matrix of $\mathscr{G}$'s exponential map in coordinates at $\mathbf{0}$. Note that, because the differential of the exp map in $\mathscr{A}$ at zero is the identity map, this $\mathbf{V}$-matrix is also the Jacobian of the implicit identification $\mathfrak{g}$ into $\mathscr{A}$. We can use $\mathbf{V}$ to convert coordinates with respect to $\mathscr{B}_{\mathfrak{g}}$ to coordinates in $\mathscr{A}$.

Next, we derive the Jacobian $\frac{\partial}{\partial \boldsymbol{\phi}_l} \mathbf{r}_{i,j}$, for any $1 \leq j < O$ and with $k := s(i) + j$. To begin, we will use the exponential chart to decompose $\mathbf{r}_{i,j}$ into a chain of maps between vector spaces allowing application of the traditional chain rule:

$$\mathbf{r}_{i,j} = \exp(\beta_{i,j} \boldsymbol{\varphi}_k) = \exp(\beta_{i,j} \log(\mathbf{g}_{k-1}^{-1} \mathbf{g}_k)) = \underbrace{\exp}_{\mathrm{I}} \circ (\beta_{i,j} \cdot) \circ \underbrace{(\log \circ \Phi_{\overline{\mathbf{d}}})}_{\mathrm{II}} \circ \underbrace{(\Phi_{\overline{\mathbf{d}}}^{-1} \circ \mathbf{d})}_{\mathrm{III}}, \tag{6.23}$$

with $\mathbf{d} := \mathbf{g}_{k-1}^{-1}(\boldsymbol{\phi}_{k-1}) \mathbf{g}_k(\boldsymbol{\phi}_k)$, and $\overline{\mathbf{d}} = \overline{\mathbf{g}}_{k-1}^{-1} \overline{\mathbf{g}}_k$.

The the Jacobian for I, $\mathscr{G}$'s exponential map at $\overline{\mathbf{v}} := \beta_{i,j} \overline{\boldsymbol{\varphi}}_k$, is usually known for the specific $\mathscr{G}$. We label it with $\mathbf{E}(\overline{\mathbf{v}})$ and assume it to be given. If it is instead available for the exponential map in exponential chart, $\Phi_{\exp(\mathbf{v})}^{-1} \circ \exp$, here denoted with $\mathbf{S}(\overline{\mathbf{v}})$, one can use the identity $\mathbf{E} = \exp(\overline{\mathbf{v}})^R \mathbf{V} \mathbf{S}(\overline{\mathbf{v}})$ to retrieve it. It holds because $\exp(\overline{\mathbf{v}})^R \mathbf{V}$ is the Jacobian of $\Phi_{\exp(\overline{\mathbf{v}})}$ at zero. In case none is available, a general formula for this $\mathbf{S}$-matrix is well known (Hall [66, p. 70]):

$$\mathbf{S}(\mathbf{v}) = \sum_{k=0}^{\infty} \frac{(-1)^k}{(k+1)!} (\mathrm{ad}_{\mathbf{v}})^k, \tag{6.24}$$

where $\mathrm{ad}_{\mathbf{v}}$ denotes the matrix representing the adjoint action, $\mathbf{w} \mapsto [\mathbf{v}, \mathbf{w}]$, where the Lie bracket $[\mathbf{v}, \mathbf{w}]$ equals $\mathbf{vw} - \mathbf{wv}$, using the algebra multiplication, in our assumed setting.

The Jacobian of II, the logarithm in the exponential chart, $\log \circ \Phi_{\overline{\mathbf{d}}}$, we label with $\mathbf{L}(\overline{\mathbf{d}})$ and assume it to be given as well. If not it can be retrieved as the inverse of $\mathbf{S}(\log(\overline{\mathbf{d}}))$, because

$\log \circ \Phi_{\bar{\mathbf{d}}} \circ \Phi^{-1}_{\exp(\log(\bar{\mathbf{d}}))} \circ \exp = \mathrm{Id}$.

The last piece is the Jacobian of III:

$$\frac{\partial}{\partial \boldsymbol{\phi}_l} \Phi^{-1}_{\bar{\mathbf{d}}}(\mathbf{d}) = (\delta_{l,k} - \delta_{l,k-1}) \mathbf{C}(\bar{\mathbf{g}}^{-1}_{k-1}), \tag{6.25}$$

as proven in the appendix, section 6.3, with

$$\mathbf{C}(\mathbf{g}) := \mathbf{W} \mathbf{g}^L \mathbf{g}^R \mathbf{V}, \tag{6.26}$$

the Jacobian at identity of the *adjoint operation* of $\mathbf{g}$ on $\mathfrak{g}$, where $\mathbf{W}$ denotes the Jacobian of the log at $\boldsymbol{\iota}$.

Altogether we have:

$$\frac{\partial \mathbf{r}_{i,j}}{\partial \boldsymbol{\phi}_l} = \mathbf{E}(\beta_{i,j} \overline{\boldsymbol{\varphi}}_k) \beta_{i,j} \frac{\partial \boldsymbol{\varphi}_k}{\partial \boldsymbol{\phi}_l} = \bar{\mathbf{r}}_{i,j}{}^R \mathbf{V} \mathbf{S}(\beta_{i,j} \overline{\boldsymbol{\varphi}}_k) \beta_{i,j} \frac{\partial \boldsymbol{\varphi}_k}{\partial \boldsymbol{\phi}_l}, \tag{6.27}$$

$$\text{with } \frac{\partial \boldsymbol{\varphi}_k}{\partial \boldsymbol{\phi}_l} = (\delta_{l,k} - \delta_{l,k-1}) \mathbf{L} \left(\bar{\mathbf{g}}^{-1}_{k-1} \bar{\mathbf{g}}_k \right) \mathbf{C}(\bar{\mathbf{g}}^{-1}_{k-1}) \tag{6.28}$$

According to section 3.4 we also need the corresponding Jacobians for the time derivatives of the B-spline. Starting with (6.16) and applying the same method as for (6.21), we have

$$\frac{\partial}{\partial \boldsymbol{\phi}_l} \frac{\mathrm{d}^k}{\mathrm{d}t^k} \mathbf{g} = k! \, \frac{\partial}{\partial \boldsymbol{\phi}_l} \mathbf{g}_{s(i)} \mathbf{a}_{k,1,O} \tag{6.29}$$

$$= k! \mathbf{a}_{k,1,O-1}{}^R \frac{\partial \mathbf{g}_{s(i)}}{\partial \boldsymbol{\phi}_l} + k! \mathbf{g}_{s(i)}{}^L \sum_{\hat{j}=1}^{O-1} \sum_{\substack{\boldsymbol{\gamma} \in \mathbb{N}^3, \\ \sum \boldsymbol{\gamma} = k}} \mathbf{a}_{\gamma_1,1,j-1}{}^L \mathbf{a}_{\gamma_2,j+1,O}{}^R \frac{\partial}{\partial \boldsymbol{\phi}_l} D^{\gamma_3} \mathbf{r}_{i,\hat{j}} \tag{6.30}$$

$$= k! \mathbf{a}_{k,1,O-1}{}^R \frac{\partial \mathbf{g}_{s(i)}}{\partial \boldsymbol{\phi}_l} + k! \mathbf{g}_{s(i)}{}^L \sum_{\hat{j}=1}^{O-1} \sum_{\gamma_1=0}^{k} \mathbf{a}_{\gamma_1,1,j-1}{}^L \left(\sum_{\gamma_2=0}^{k-\gamma_1} \mathbf{a}_{\gamma_2,j+1,O}{}^R \frac{\partial}{\partial \boldsymbol{\phi}_l} D^{k-\gamma_1-\gamma_2} \mathbf{r}_{i,\hat{j}} \right), \tag{6.31}$$

$$\text{with } \mathbf{a}_{k,j_F,j_L} := \sum_{\substack{\boldsymbol{\alpha} \in \mathbb{N}^{j_L - j_F}, \\ \sum \boldsymbol{\alpha} = k}} \prod_{j=j_F}^{j_L-1} D^{\boldsymbol{\alpha}_{j-j_F+1}} \mathbf{r}_{i,j}, \text{ if } j_L - 1 > j_F \text{ else } \iota, \text{and } D^{\mu} := \frac{1}{\mu!} \frac{\mathrm{d}^{\mu}}{\mathrm{d}t^{\mu}}. \tag{6.32}$$

The last transformation (6.31) is only to reduce computational complexity by exploiting the distributive law of matrix multiplication. Combining (3.6) and (3.6), we have (not expanding the derivatives of $\boldsymbol{\varphi}_k$ and only up to $k = 2$ for the sake of readability):

$$\frac{\partial}{\partial \boldsymbol{\phi}_l} \frac{\mathrm{d}}{\mathrm{d}t} \mathbf{r}_{i,j}(t) = \beta'_{i,j}(t) \frac{\partial}{\partial \boldsymbol{\phi}_l} (\mathbf{V} \boldsymbol{\varphi}_k) \mathbf{r}_{i,j}(t) = \beta'_{i,j}(t) \left(\mathbf{r}_{i,j}(t)^R \mathbf{V} \frac{\partial \boldsymbol{\varphi}_k}{\partial \boldsymbol{\phi}_l} + (\mathbf{V} \boldsymbol{\varphi}_k)^L \frac{\partial \mathbf{r}_{i,j}(t)}{\partial \boldsymbol{\phi}_l} \right) \tag{6.33}$$

$$\begin{aligned}\frac{\partial}{\partial \boldsymbol{\phi}_l}\frac{\mathrm{d}^2}{\mathrm{d}t^2}\mathbf{r}_{i,j}(t) = \left(\beta'_{i,j}(t)^2\mathbf{V}\boldsymbol{\varphi}_k + \beta''_{i,j}(t)\boldsymbol{\iota}\right)^L \left(\mathbf{r}_{i,j}(t)^R\mathbf{V}\frac{\partial \boldsymbol{\varphi}_k}{\partial \boldsymbol{\phi}_l} + (\mathbf{V}\boldsymbol{\varphi}_k)^L\frac{\partial \mathbf{r}_{i,j}(t)}{\partial \boldsymbol{\phi}_l}\right)\\ + ((\mathbf{V}\boldsymbol{\varphi}_k)\mathbf{r}_{i,j}(t))^R(\beta'_{i,j}(t)^2\mathbf{V}\frac{\partial \boldsymbol{\varphi}_k}{\partial \boldsymbol{\phi}_l}) \qquad (6.34)\end{aligned}$$

3.7. Rough B-spline fitting for initialization

To initialize the nonlinear optimization of a $\mathscr{G}$-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 $\mathscr{T} = [0,T]$ to a time series in $\mathscr{G}$, $\alpha := (\tau_k, \mathbf{g}_k)_{k=1}^K \in \mathscr{T} \times \mathscr{G}$, while somehow penalizing high acceleration, and preventing overfitting. For our experiments we used the following strategy to get a rough fit of our B-spline by making use of Assumption 1. As we have $\mathbf{g}(t, (\mathbf{g}_k)_{k=1}^K) \in \mathscr{G} \subset \mathscr{A}_{\text{inv}} \subset \mathscr{A}$ for all times $t \in \mathscr{T}$ and control vertices $\mathbf{g}_k \in \mathscr{G}$ we can consider the whole $\mathscr{G}$-valued B-spline construct as a Lie group B-spline in the Lie group $(\mathscr{A}_{\text{inv}}, \cdot)$, i.e. based on the multiplication in $\mathscr{A}$. In $\mathscr{A}$ we also have the $\mathbb{R}$-vector space structure and we can define for each multiplicative $\mathbf{g}(t, (\mathbf{g}_k)_{k=1}^K)$ the corresponding additive B-spline $\mathbf{g}^{\text{add}}(t, (\mathbf{g}_k)_{k=1}^K)$ as defined in the additive Lie group, $(\mathscr{A}, +)$. These are now traditional vector space-valued 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 [44]). After the linear fit, we may need to modify the control vertices to make them elements in $\mathscr{G}$. 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 $\mathbf{g}_k \in \mathscr{A}$ by doing a least squares fit of $\mathbf{g}^{\text{add}}$ to α while keeping the integral over the spline's acceleration low (by introducing an appropriate, linear cost scaled with an acceleration penalty factor),

2. project the $\mathbf{g}_k$s into $\mathscr{G}$, and

3. use the modified $\mathbf{g}_k$ as control vertices for a multiplicative spline in $\mathscr{G}$.

Despite being very simple, this method worked extremely well in all of our experiments up to B-spline order 12, where the linear solution for the initialization fell into local likelihood maxima for tight knot spacing when using our default acceleration penalty. This could be solved by rising the acceleration penalty factor from 10^4 to 10^6. Automatic tuning of this parameter is out of the scope of 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. This would leave us no means to penalize high acceleration, which would allow outliers to badly affect the result.

3.8. 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 $\mathscr{T} = [0, T]$ and a cost functional J defined on continuous curves through $\mathscr{G}$, $\gamma : \mathscr{T} \to \mathscr{G}$ such that

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

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

$$\underset{g \in \mathscr{C}(\mathscr{T}, \mathbb{R})}{\forall} \sum_{i=1}^{k} w_i g(\xi_i) \approx \int_0^T g(t)\,\mathrm{d}t$$

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

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

3.9. 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

$$\mathbb{S}^3 := \{\mathbf{q} \in \mathbb{H} \,|\, |\mathbf{q}| = 1\}$$

denote the three dimensional $\mathbb{R}$-Lie group of unit-length quaternions. We can make use of the theory in section 3.3 using the following identification. In this setting, $\mathbb{H}$ is the ambient $\mathbb{R}$-algebra ($\mathscr{A}$) and $\mathbb{S}^3$ is the embedded Lie group ($\mathscr{G}$).

We will rely on the notation presented in Barfoot et al. [26]. Let $\boldsymbol{\iota}$ denote the *identity element* of $\mathbb{S}^3$, 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

$$\mathbf{q} =: \begin{bmatrix} \boldsymbol{\varepsilon} \\ \eta \end{bmatrix}, \tag{6.35}$$

where $\boldsymbol{\varepsilon}$ 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

$$\mathbf{q}^L = \begin{bmatrix} \eta \mathbf{1} + \boldsymbol{\varepsilon}^\times & \boldsymbol{\varepsilon} \\ -\boldsymbol{\varepsilon}^T & \eta \end{bmatrix} \text{ and } \mathbf{q}^R = \begin{bmatrix} \eta \mathbf{1} - \boldsymbol{\varepsilon}^\times & \boldsymbol{\varepsilon} \\ -\boldsymbol{\varepsilon}^T & \eta \end{bmatrix}, \tag{6.36}$$

where $\mathbf{1}$ is the 3×3 identity matrix, and $\boldsymbol{\varepsilon}^\times$ denotes the matrix such that for all $\boldsymbol{x} \in \mathbb{R}^3$, $\boldsymbol{\varepsilon} \times \boldsymbol{x} =$

$\boldsymbol{\varepsilon}^{\times}\boldsymbol{x}$. The quaternion *compound* operators, $(\cdot)^{\oplus}$, $(\cdot)^{+}$ of Barfoot et al. [26], are precisely $(\cdot)^{L}$ and $(\cdot)^{R}$. Given an unit-length quaternion, $\mathbf{q} \in \mathbb{S}^3$, the *complex conjugate* operator coincides with multiplicative inverse operator,

$$\mathbf{q}^{-1} = \begin{bmatrix} -\boldsymbol{\varepsilon} \\ \eta \end{bmatrix}. \tag{6.37}$$

The Lie algebra of $\mathbb{S}^3$ consists exactly of the pure imaginary quaternions. We pick the usual basis, $\mathscr{B}_{\mathfrak{g}} := (\mathbf{i}, \mathbf{j}, \mathbf{k})$ 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

$$\mathbf{V} = \begin{bmatrix} \mathbf{1} \\ \mathbf{0}^T \end{bmatrix}, \text{ implying } \mathbf{V}\boldsymbol{\phi} = \begin{bmatrix} \boldsymbol{\phi} \\ 0 \end{bmatrix}, \text{ and } \mathbf{W} = \mathbf{V}^T, \text{ implying } \mathbf{W}\mathbf{V} = \mathbf{1}, \tag{6.38}$$

which correspond to the matrices introduced after (6.22) and (6.26) respectively as the Jacobians of the exponential map, at zero, and logarithm map, at identity.

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

$$\mathrm{Rot}_{\mathbf{q}} : \mathbb{R}^3 \to \mathbb{R}^3, \mathbf{x} \mapsto \mathbf{W}(\mathbf{q}(\mathbf{V}\mathbf{x})\mathbf{q}^{-1}). \tag{6.39}$$

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

$$\mathbf{C} = \mathbf{W}\mathbf{q}^{L}\mathbf{q}^{-1^{R}}\mathbf{V}. \tag{6.40}$$

This is precisely the matrix $\mathbf{C}$ defined in (6.26).

This unit-length quaternion's log and exp functions can be calculated in the following way

$$\exp(\boldsymbol{\phi}) = \begin{cases} \boldsymbol{\iota} & , \boldsymbol{\phi} = \mathbf{0} \\ \begin{bmatrix} \sin(\phi)\mathbf{a} \\ \cos\phi \end{bmatrix} & , \boldsymbol{\phi} \neq \mathbf{0} \end{cases}, \quad \log(\boldsymbol{q}) = \begin{cases} \mathbf{0} & , \boldsymbol{q} = \boldsymbol{\iota} \\ \frac{\arccos\eta}{\sqrt{1-\eta^2}}\boldsymbol{\varepsilon} & , \boldsymbol{q} \neq \pm\boldsymbol{\iota} \\ \text{undefined} & , \boldsymbol{q} = -\boldsymbol{\iota} \end{cases}, \tag{6.41}$$

where $\boldsymbol{\phi}$ and the log's value are 3×1 coordinate vectors with respect to $\mathscr{B}_{\mathfrak{g}}$, $\phi := \|\boldsymbol{\phi}\|$, and $\mathbf{a} := \boldsymbol{\phi}/\phi$. A derivation of (6.41) can be found in the appendix of Kim and Nam [87].

3.10. Derivatives of quaternion B-splines

In this section we give the time derivative and Jacobian formulas specific to our formulation for curves over $\mathbb{S}^3$. The B-spline time derivatives we calculated for the general case in (3.6) apply immediately to an unit-length quaternion curve, $\mathbf{q}(t) = [\boldsymbol{\varepsilon}(t)^T\ \eta(t)]^T$, giving us directly quantities like $\dot{\boldsymbol{\varepsilon}}(t)$, $\ddot{\boldsymbol{\varepsilon}}(t)$, $\dot{\eta}(t)$, $\ddot{\eta}(t)$, 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 (6.39) in a physical context. We will identify the rotation from frame $\underrightarrow{\mathscr{F}}_a$, an inertial frame, to $\underrightarrow{\mathscr{F}}_b$, one attached to a rigid body, by the unit-length quaternion, $\mathbf{q}_{ba}$, such that

$$\forall_{\mathbf{v}\in\mathbb{E}^3} \mathrm{Rot}_{\mathbf{q}_{ba}}(\mathbf{v}_a) = \mathbf{v}_b, \tag{6.42}$$

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.

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 $\mathbf{q}_{ba}(t) = [\boldsymbol{\varepsilon}(t)^T \eta(t)]^T$ we get (omitting the time parameter for clarity):

$$\boldsymbol{\omega}_a = -2\mathbf{W}(\mathbf{q}_{ba}^{-1}\dot{\mathbf{q}_{ba}}) = -2((\eta\dot{\boldsymbol{\varepsilon}} - \dot{\eta}\boldsymbol{\varepsilon}) - \boldsymbol{\varepsilon}^{\times}\dot{\boldsymbol{\varepsilon}}), \tag{6.43}$$

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

$$\boldsymbol{\omega}_b = -2\mathbf{W}(\dot{\mathbf{q}_{ba}}\mathbf{q}_{ba}^{-1}) = -2((\eta\dot{\boldsymbol{\varepsilon}} - \dot{\eta}\boldsymbol{\varepsilon}) - \dot{\boldsymbol{\varepsilon}}^{\times}\boldsymbol{\varepsilon}), \tag{6.44}$$

when expressed in $\underset{\rightarrow}{\mathcal{F}}_b$. Both equations are proven in the appendix, section 6.1.

Angular acceleration

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

$$\begin{aligned}\boldsymbol{\alpha}_b = \dot{\boldsymbol{\omega}}_b &= -2\mathbf{W}(\frac{\mathrm{d}}{\mathrm{d}t}(\dot{\mathbf{q}}\mathbf{q}^{-1})) = -2\mathbf{W}(\dot{\mathbf{q}^{-1}}\dot{\mathbf{q}} + \ddot{\mathbf{q}}\mathbf{q}^{-1}) = -2\mathbf{W}(\ddot{\mathbf{q}}\mathbf{q}^{-1}) && (6.45)\\ &= -2((\eta\ddot{\boldsymbol{\varepsilon}} - \ddot{\eta}\boldsymbol{\varepsilon}) - \ddot{\boldsymbol{\varepsilon}}^{\times}\boldsymbol{\varepsilon}) && (6.46)\end{aligned}$$

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

To use the results from section 3.6 we will need the specific matrices for the unit-length quaternions. The Jacobian of the exponential map, $\mathbf{S}(\boldsymbol{\phi})$, and of the logarithm map, $\mathbf{L}(\mathbf{q})$, using the same variables as in (6.41), additionally assuming for $\mathbf{L}(\mathbf{q})$ that $\log(\mathbf{q}) = \boldsymbol{\phi}$. Both are $\mathbf{1} \in \mathbb{R}^{3\times 3}$ in case $\boldsymbol{\phi} = \mathbf{0}$ and $\mathbf{q} = \boldsymbol{\iota}$ respectively. Otherwise

$$\mathbf{S}(\boldsymbol{\phi}) = \mathbf{1} - \frac{1}{\phi}\sin^2\phi\,\mathbf{a}^{\times} + (1 - \frac{1}{2\phi}\sin 2\phi)\mathbf{a}^{\times}\mathbf{a}^{\times}, \text{and} \tag{6.47}$$

$$\mathbf{L}(\mathbf{q}) = \mathbf{1} + \boldsymbol{\phi}^{\times} + \left(1 - \frac{\phi}{\tan\phi}\right)\mathbf{a}^{\times}\mathbf{a}^{\times} \text{ for } \mathbf{q} \neq \pm\boldsymbol{\iota}. \tag{6.48}$$

These formulas are proven / derived in the appendix, section 6.2.

Jacobians of angular velocity

The Jacobians of the angular velocity with respect to minimal perturbations of the control vertices are also required. For $1 \leq j < O$ and $l \in \{1..K\}$ we get,

$$\frac{\partial\boldsymbol{\omega}_a}{\partial\boldsymbol{\phi}_l} = -2\left(\frac{\partial\eta}{\partial\boldsymbol{\phi}_l}\dot{\boldsymbol{\varepsilon}} + \eta\frac{\partial\dot{\boldsymbol{\varepsilon}}}{\partial\boldsymbol{\phi}_l} - \frac{\partial\dot{\eta}}{\partial\boldsymbol{\phi}_l}\boldsymbol{\varepsilon} - \dot{\eta}\frac{\partial\boldsymbol{\varepsilon}}{\partial\boldsymbol{\phi}_l} + \dot{\boldsymbol{\varepsilon}}^{\times}\frac{\partial\boldsymbol{\varepsilon}}{\partial\boldsymbol{\phi}_l} - \boldsymbol{\varepsilon}^{\times}\frac{\partial\dot{\boldsymbol{\varepsilon}}}{\partial\boldsymbol{\phi}_l}\right). \tag{6.49}$$

Jacobians of angular acceleration

For the angular acceleration it yields analogously,

$$\frac{\partial \boldsymbol{\alpha}_a}{\partial \boldsymbol{\phi}_l} = -2\left(\frac{\partial \eta}{\partial \boldsymbol{\phi}_l}\ddot{\boldsymbol{\varepsilon}} + \eta\frac{\partial \ddot{\boldsymbol{\varepsilon}}}{\partial \boldsymbol{\phi}_l} - \frac{\partial \ddot{\eta}}{\partial \boldsymbol{\phi}_l}\boldsymbol{\varepsilon} - \ddot{\eta}\frac{\partial \boldsymbol{\varepsilon}}{\partial \boldsymbol{\phi}_l} + \ddot{\boldsymbol{\varepsilon}}^{\times}\frac{\partial \boldsymbol{\varepsilon}}{\partial \boldsymbol{\phi}_l} - \boldsymbol{\varepsilon}^{\times}\frac{\partial \ddot{\boldsymbol{\varepsilon}}}{\partial \boldsymbol{\phi}_l}\right). \tag{6.50}$$

3.11. Associating an unit quaternion time series to a rotation time series

To fit an unit quaternion curve to a time series of noisy rotations can be a challenge. For the fitting procedure described in 3.7 one has to first convert it to an 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 (6.39) 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 iterating over the rotation time series starting from one side and appending in each step to the target quaternion time series the one among the two unit-length quaternions equally well corresponding to the current rotation, which has minimal distance to its predecessor quaternion chosen in the step before. However, this strategy can perform badly in the presence of outliers. Our current implementation iterates through the time series of rotations choosing in each iteration the unit-length quaternion candidate that minimizes the total distance to its (up to) three predecessors. This worked well enough in our experiments, with one exception noted in Section 5.2.

4. 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, $\underrightarrow{\mathscr{F}}_B$, with respect to an inertial frame, $\underrightarrow{\mathscr{F}}_I$, over a time interval of interest, $\mathscr{T} = [0, T]$. We would like to estimate an unit-length quaternion trajectory describing the rotation from the inertial frame to the body frame, $\mathbf{q}_{BI}(t)$, over $\mathscr{T}$.

The vehicle may have bearing sensors (we will consider one or two). A bearing sensor takes measurements at discrete time instances, $(t_m)_{m=1}^{M} \in \mathscr{T}$. The sensor's frame $\underrightarrow{\mathscr{F}}_S$ 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. An instantaneous measurement of this bearing (i.e. a discrete-time measurement), written $\mathbf{y}_{m\ell}$, is modelled as

$$\mathbf{y}_{m\ell} = \mathbf{h}\left(\mathbf{C}_{SB}\mathbf{C}_{\mathbf{q}_{BI}(t_m)}\mathbf{b}_I^{m\ell}\right) + \mathbf{n}_{m\ell}, \qquad \mathbf{n}_{m\ell} \sim \mathscr{N}\left(\mathbf{0}, \mathbf{R}_{m\ell}\right), \tag{6.51}$$

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 (6.40), here acting as the direction cosine matrix from the inertial 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

$$\mathbf{e}_{m\ell} := \mathbf{y}_{m\ell} - \mathbf{h}\left(\mathbf{C}_{SB}\mathbf{C}_{\mathbf{q}_{BI}(t_m)}\mathbf{b}_I^{m\ell}\right). \tag{6.52}$$

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

$$J_y := \frac{1}{2}\sum_{m=1}^{M} \mathbf{e}_{m\ell}^T \mathbf{R}_{m\ell}^{-1} \mathbf{e}_{m\ell}. \tag{6.53}$$

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.

4.1. 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 [83, 114],

$$\boldsymbol{\varpi}_g = \boldsymbol{\omega}(t_g) + \mathbf{b}(t_g) + \mathbf{n}_{g\omega}, \qquad \mathbf{n}_{g\omega} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_{g\omega}), \tag{6.54}$$

$$\dot{\mathbf{b}}(t) = \mathbf{w}_b(t), \qquad \mathbf{w}_b \sim \mathcal{GP}\left(\mathbf{0}, \delta(t-t')\mathbf{Q}_b\right), \tag{6.55}$$

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 (see Section 3.1 for the notation). It is related to the quaternion trajectory $\mathbf{q}_{BI}(t)$ via (6.44). The gyroscope measurements are assumed to be subject to both zero-mean Gaussian measurement noise, $\mathbf{n}_{g\omega}$, and a slowly evolving bias, $\mathbf{b}(t)$. 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

$$\mathbf{e}_{g\omega} := \boldsymbol{\varpi}_g - \boldsymbol{\omega}(t_g) - \mathbf{b}(t_g), \tag{6.56}$$

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

$$J_\omega := \frac{1}{2}\sum_{g=1}^{G} \mathbf{e}_{g\omega}^T \mathbf{R}_{g\omega}^{-1} \mathbf{e}_{g\omega}. \tag{6.57}$$

The bias motion error may be written as

$$\mathbf{e}_b(t) := \dot{\mathbf{b}}(t). \tag{6.58}$$

This error contributes to our objective function as

$$J_b := \frac{1}{2}\int_0^T \mathbf{e}_b^T(t)\mathbf{Q}_b^{-1}\mathbf{e}_b(t)\,\mathrm{d}t. \tag{6.59}$$

Together, (6.53), (6.57), and (6.59) define the combined objective function, $J := J_y + J_\omega + J_b$. Let $\mathscr{Q}$ 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 $\mathbf{b}(t)$, $\mathscr{Q}^\star, {\mathbf{c}_b}^\star$, can then approximately be found by minimizing $J(\mathscr{Q}, \mathbf{c}_b)$,

$$(\mathscr{Q}^\star, {\mathbf{c}_b}^\star) = \underset{\mathscr{Q},\mathbf{c}_b}{\operatorname{argmin}} J(\mathscr{Q}, \mathbf{c}_b)\,. \tag{6.60}$$

4.2. 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,

$$\mathbf{I}\dot{\boldsymbol{\omega}}(t) + \boldsymbol{\omega}^\times(t)\mathbf{I}\boldsymbol{\omega}(t) = \mathbf{u}(t) + \mathbf{w}(t), \tag{6.61}$$

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 $\mathbf{Q}_d$,

$$\mathbf{w} \sim \mathscr{GP}\left(\mathbf{0}, \delta(t-t')\mathbf{Q}_d\right). \tag{6.62}$$

Following Furgale et al. [59] (Section 3.1), we define the error for the dynamics model at time t to be

$$\mathbf{e}_d(t) := \mathbf{I}\dot{\boldsymbol{\omega}}(t) + \boldsymbol{\omega}^\times(t)\mathbf{I}\boldsymbol{\omega}(t) - \mathbf{u}(t). \tag{6.63}$$

The resulting term in our objective function is

$$\mathbf{J}_d := \frac{1}{2}\int_0^T \mathbf{e}_d^T(t)\mathbf{Q}_d^{-1}\mathbf{e}_d(t)\,\mathrm{d}t. \tag{6.64}$$

Together, (6.53) and (6.64) define the combined objective function, $J := J_y + J_d$. Let $\mathscr{Q}$ be, again, the stacked matrix of quaternion control vertices. The trajectory estimate is then given by the J-minimizing quaternion control vertices,

$$\mathscr{Q}^\star = \underset{\mathscr{Q}}{\operatorname{argmin}} J(\mathscr{Q}). \tag{6.65}$$

4.3. 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 [158] and [41] 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,

$$\begin{aligned}
\mathbf{a}_\tau &:= \begin{bmatrix} 0.001 & 0.0005 & 0.00075 \end{bmatrix}^T \text{Nm}, && (6.66) \\
\boldsymbol{\omega}_\tau &:= \begin{bmatrix} 0.1 & 0.05 & 0.075 \end{bmatrix}^T \frac{\text{rad}}{\text{s}}, && (6.67) \\
\boldsymbol{\phi}_\tau &:= \begin{bmatrix} 0 & \frac{\pi}{2} & -\frac{\pi}{4} \end{bmatrix}^T \text{rad}, && (6.68) \\
\boldsymbol{\tau}_0 &:= \begin{bmatrix} 0.005 & -0.005 & 0.000 \end{bmatrix}^T \text{Nm}, && (6.69) \\
\boldsymbol{\tau}(t) &:= f_\tau \left(\mathbf{a}_\tau \sin(t\boldsymbol{\omega}_\tau + \boldsymbol{\phi}_\tau) + \boldsymbol{\tau}_0 \right), && (6.70)
\end{aligned}$$

where the sinus is applied componentwise. This torque is used as the input $\mathbf{u}$ in (6.61). The angular acceleration induced by the thruster is determined by the assumed inertia tensor of the spacecraft in the same frame:

$$\begin{aligned}
\mathbf{I} &:= \text{diag}\left([27\ 17\ 25]\right) \text{kgm}^2 && (6.71) \\
\boldsymbol{\alpha}_\tau(t) &:= \mathbf{I}^{-1} \boldsymbol{\tau}(t) && (6.72)
\end{aligned}$$

We chose our system process noise, described in (6.62), 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 [158] ([41] use 0.1Hz). We used the following parameters for the sensors:

- Gyroscope : $\mathbf{R}_{g\boldsymbol{\omega}} = (0.3\mu \frac{\text{rad}}{\text{s}})^2 \mathbf{1}$, $\mathbf{Q}_b = (3 * 10^{-4} \mu \frac{\text{rad}}{\text{s}^2})^2 \frac{1}{\text{s}} \mathbf{1}$ (see (4.1))

- Magnetometer : $\mathbf{R}_B = (50\text{nT})^2 \mathbf{1}$ (see (6.51))

- Sun sensor: $\mathbf{R}_S = (5\text{m rad})^2 \mathbf{1}$ (see (6.51))

The simulated gyroscope bias was started at 0deg/h or 1000deg/h depending on the experiment. To the best of our knowledge, these parameters are identical to [158] and [41], 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.

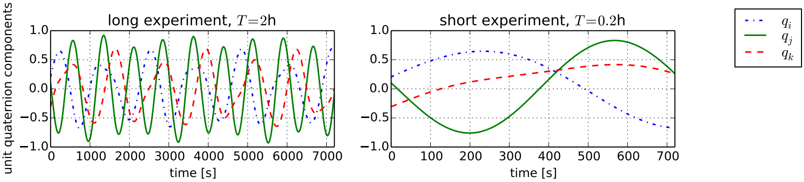


Figure 6.2.: A typical simulated attitude trajectory plotted as the three imaginary components of $\mathbf{q}_{BI}$ over the longest run time $T = 2\text{h}$ and a zoomed plot of the first 0.2h.

The typical simulated attitude trajectory is depicted in Figure 6.2. It depicts the three imaginary components of the attitude quaternion, $\mathbf{q}_{BI}$, over our maximal duration $T = 2\text{h}$ and zooms in to the first $[0, 0.2]$h interval as used in most of the short experiments.

4.4. 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 4.1, and the dynamics model, as described in 4.2 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 [158] 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 [158].

4.5. 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 [161, 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 3.7), 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 3.8) 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 prior bias mean is always $\mathbf{0} \in \mathbb{R}^3$. We use $T = 0.2\text{h}$ 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.

5. 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.

To compute the MADE(ΔT), we evaluate

$$\text{MADE}(\Delta T) := \frac{1}{|\mathscr{U}|}\sum_{t\in\mathscr{U}} d_{\text{angle}}(\hat{\mathbf{q}}(t), \mathbf{q}(t)), \tag{6.73}$$

where d_{angle} denotes the angular distance (in degrees) of two unit quaternions (:= Riemannian distance of the represented rotations), $\hat{\mathbf{q}}(t)$ the estimated attitude quaternion at time t, $\mathbf{q}(t)$ the simulated attitude quaternion, $\mathscr{U} \subset]T-\Delta T, T]$ the set of all time instances for which both $\hat{\mathbf{q}}(t)$ and $\mathbf{q}(t)$ are available and $|\mathscr{U}|$ its cardinality. For all continuous estimators $\mathscr{U}$ are the time instances in $]T-\Delta T, T]$ for which $\mathbf{q}(t)$ was simulated (at 10 Hz). The discrete MEKF always runs on a regular subset (depending on the experiment) of the simulated times and hence defines the $\mathscr{U}$ for its MADE evaluation.

This approximates a temporally averaged angular distance because $\mathscr{U}$ is always evenly spaced which yields for two continuous unit quaternion-valued functions $\mathbf{q}_1, \mathbf{q}_2$ on $[T-\Delta T, T]$

$$\frac{1}{|\mathscr{U}|} \sum_{t \in \mathscr{U}} d_{\text{angle}}(\mathbf{q}_1(t), \mathbf{q}_2(t)) \simeq \frac{1}{\Delta T} \int_{T-\Delta T}^{T} d_{\text{angle}}(\mathbf{q}_1(t), \mathbf{q}_2(t)) \mathrm{d}t. \tag{6.74}$$

We decided to use the Riemannian distance (Euclidean norm of the relative rotation vector) instead of the root sum squared (RSS) of roll, pitch and yaw used in Vandersteen et al. [158] and Crassidis and Markley [41] because the latter is not isotropic and would render the result dependent of the current pose of the space craft. For small errors they are very similar in magnitude and thus the results are still comparable.

Except for this difference in the error metric at a point in time MADE(T) is like the integral cost J defined in Equation (46) of Crassidis and Markley [41].

5.1. Gyroscope based

First the experiments based on gyroscope measurements, corresponding to 4.1.

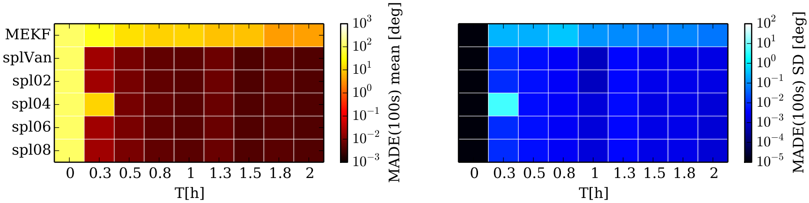


Figure 6.3.: MADE(100s) over the observation time T given a simulated initial bias of 1000deg/h. $T \in [0,2]$h. Sensors: rate gyroscope, magnetometer.

Figure 6.3 shows the results for the extreme initial bias test case of [158]. 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 than the linear ones and the 6th-order. The former is probably mainly an effect of the fact that the linear splines have twice as many knots. Qualitatively these results fit the results of [158] 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 [158].

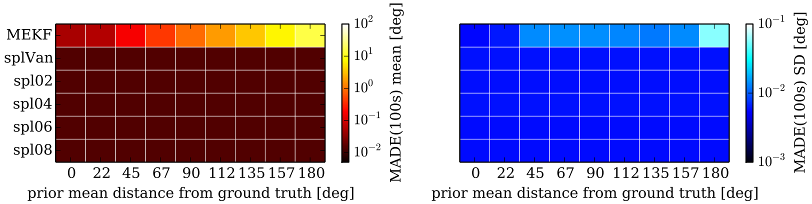


Figure 6.4.: MADE(100s) over the distance of the prior mean to ground truth. $T = 0.2$h. Sensors: rate gyroscope, magnetometer.

In Figure 6.4 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.

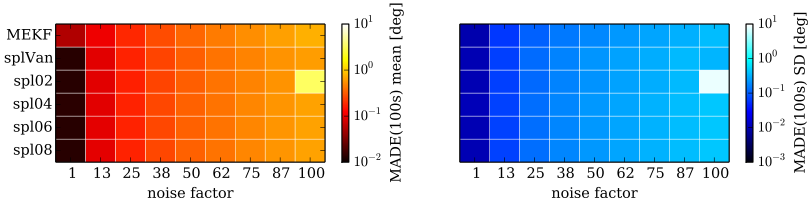


Figure 6.5.: MADE(100s) over a factor multiplying the standard deviation of all sensor noise. $T = 0.2$h. Sensors: rate gyroscope, sun sensor, magnetometer.

In Figure 6.5 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 MEKF converges in every case due to the high sensor rate.

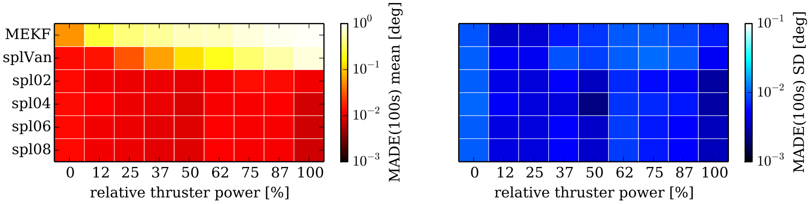


Figure 6.6.: MADE(100s) over the thruster's power factor in percentage. $T = 0.2\text{h}$. Sensors: rate gyroscope, sun sensor, magnetometer.

In Figure 6.6 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.

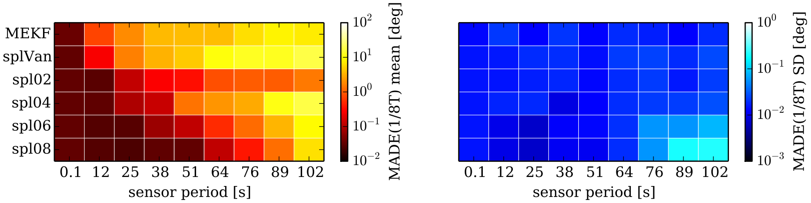


Figure 6.7.: MADE($\frac{1}{8}T$) over the sensors' period. T is determined by a fixed amount of 64 measurements per sensor. Sensors: rate gyroscope, sun sensor, magnetometer.

In Figure 6.7 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 6.7 and Figure 6.6 (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.

5.2. 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 6.3 in the last section), as we do not have an 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 6.8 we show the response of each estimator to a bad prior mean after $T = 0.2\text{h}$ of simulation time. It shows again how all splines perform superior to the MEKF.

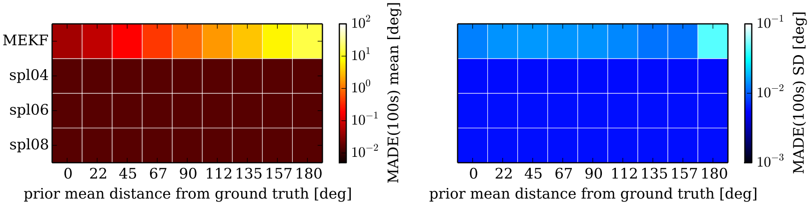


Figure 6.8.: MADE(100s) over the angular distance error of the prior. $T = 0.2\text{h}$. Sensors: magnetometer.

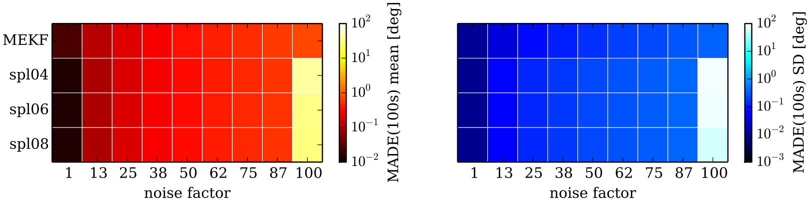


Figure 6.9.: MADE(100s) over a factor multiplying the standard deviation of all sensor noise. $T = 0.2\text{h}$. Sensors: sun sensor, magnetometer.

In Figure 6.9 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 3.11). 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 6.5), the estimator is able to find its way back to the correct solution.

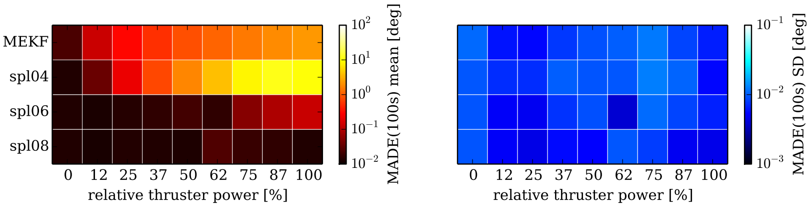


Figure 6.10.: MADE(100s) over the thruster's power factor. $T = 0.2$h. Sensors: sun sensor, magnetometer.

In Figure 6.10 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 (6.61) this difference continuously decreases. We will analyze that effect further in the discussion Section 5.3.

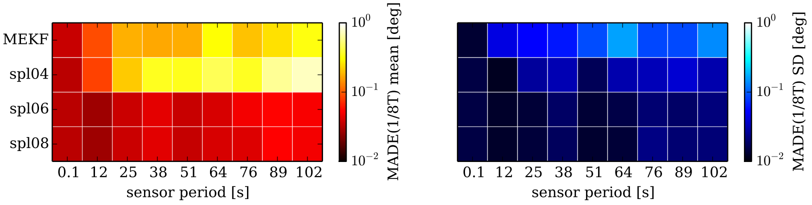


Figure 6.11.: MADE($\frac{1}{8}T$) over the sensors' period. T is determined by a fixed amount of 64 measurements per sensor. Sensors: sun sensor, magnetometer.

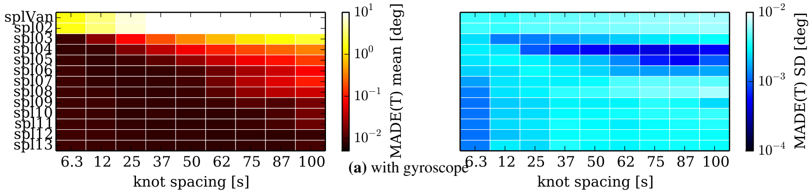
In Figure 6.11 we tested the effect of the sensor rate (as in Figure 6.7). Again, the Kalman filter is much more negatively affected than the high order splines. Figure 6.10 is again quite similar to the first 20 seconds. The bad effect on the 4th-order spline is more severe than in the gyroscope experiment. 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.

5.3. Discussion

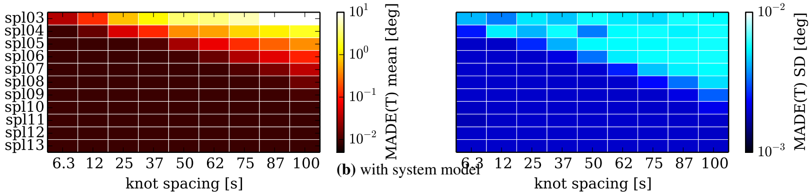
Impact of the B-spline order

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

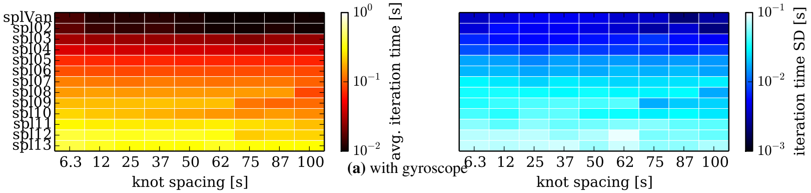
Figure 6.12.: MADE(T) over the time between knots. $T = 0.2$h. Sensors: sun sensor, magnetometer.

In Figure 6.12 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 3.7 and required to raise 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.

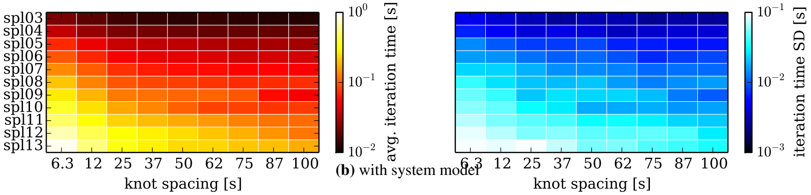
In the high-accuracy range the result is even more extreme. For example, in the gyroscope-based experiment, a 4th-order spline requires at least 8 times more segments to achieve the same accuracy as a 12th-order spline. In the system model experiment it is 16 times more.

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.

Computational cost



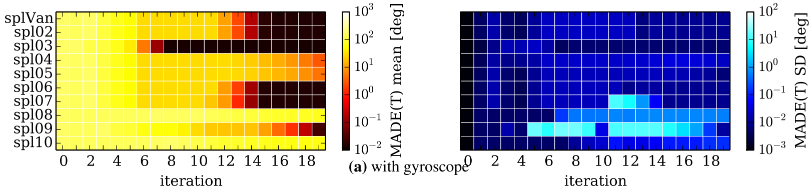
(a) with gyroscope



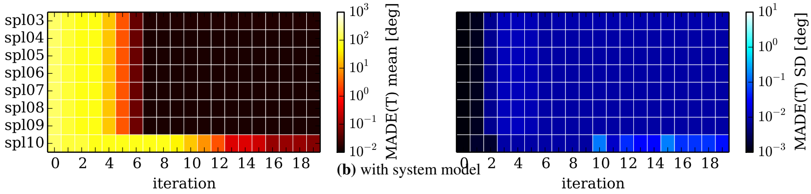
(b) with system model

Figure 6.13.: Time for one iteration over knot spacing on a single Intel(R) Xeon(R) E5 core at 3.6GHz. $T = 0.2$h, yielding 720 readings per sensor. Sensors: sun sensor, magnetometer.

Of course the estimation accuracy 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 6.13. 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



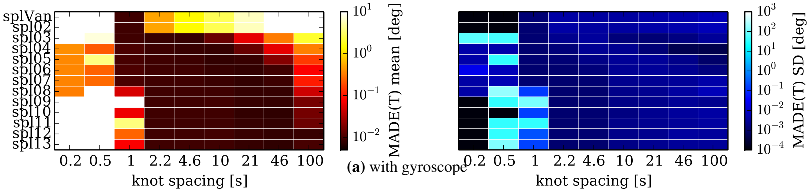
(b) with system model

Figure 6.14.: MADE(T) over iterations ($0 =$ after initialization) after a *very bad* B-spline initialization: constant, at 120deg off the initial attitude. $T = 0.2\text{h}$. Sensors: rate gyroscope, magnetometer.

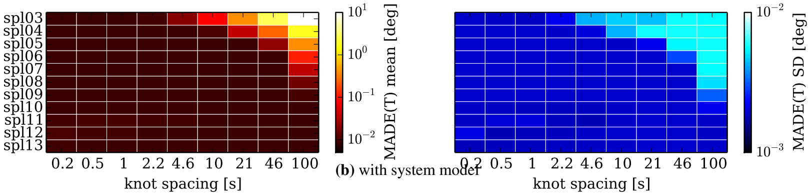
Convergence Behavior

In most of experiments the *convergence behavior* is as follows: there is typically a big step after the first iteration followed by minor improvements in subsequent iterations, except for the linear splines that converge slower and mutually very similar. That is why in Figure 6.14 we show the convergence behavior in a very difficult setup instead. The B-spline gets initialized with a constant value, which is 120deg off the simulated initial attitude, and the estimators are only using the magnetometer and the rate gyroscope. In the gyroscope case the order seems to heavily influence if and how long it takes to recover from the bad initial value but without any obvious rule how. In the system model experiment the typical independence of the order is clearly visible except for very high order B-splines (here 10th order). The sensor rates almost have no impact on the convergence behavior. However, the assumed sensor noise has a surprising big impact. Interestingly, increasing the sensor noise in the gyroscope case will first increase the convergence speed, even in absolute accuracy. Further rising the noise magnitude results in slower convergence and, eventually, the convergence behaviour will become independent of the order.

Under-determined case



(a) with gyroscope



(b) with system model

Figure 6.15.: MADE(T) over the time between knots on a logarithmic scale. The sensor rate is at 1 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 experiment, very similar to Figure 6.12 by starting the knot spacing below the sensor period (1s), and exponentially increasing the number of knots. The result is depicted in Figure 6.15. 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 no longer able to represent the trajectory.

Approximation capacity analysis

In our estimation experiments we do not distinguish between estimation error and approximation error. In a next step we therefore analyze the pure approximation error in a separate experiment.

Figure 6.16 shows the result of this experiment. A sinusoidal attitude trajectory, turning around a fixed axis with amplitude, $A \in \{10, 45, 90, 180\}$deg, and $N = 5$ periods of 1 s each is approximated with evenly spaced until-length quaternion B-splines of various order, $O \in \{2..9\}$, and various number of valid segments $S := I - 2O + 1$. When increasing S, typically there is a big leap to good accuracy between $S = 2N$ and $S = 6N$. Both higher amplitude A and higher order pushes this leap further towards $6N$.

All our estimation experiments have S/N fractions above the maximum in this plot (~ 15) except for the experiments in Figure 6.12 and Figure 6.15 with very high knot spacing, which are hybrids between estimation and approximation experiments. The typical accuracy for $S/N = 15$ is already $< 10^{-3}$deg, except for the linear B-spline case, which therefore possibly have hit the bounds of their expressiveness for some sub experiments. We can conclude that the rest of the estimation experiments show mainly the estimation error, as for those the best accuracy is significantly worse than 10^{-3}deg.

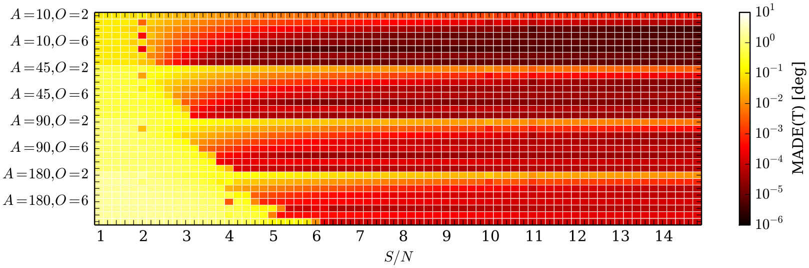


Figure 6.16.: Approximation error when fitting B-splines with order, $O \in \{2..9\}$, and S uniform segments to sinusoidal attitude trajectories with amplitudes, $A \in \{10, 45, 90, 180\}$deg and $N = 5$ periods.

6. Conclusions

In this paper we extended an existing unit-length quaternion B-spline formulation 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 a state-of-the-art approach. 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.

Appendix

6.1. Proofs for angular velocity equations

Let $\mathbf{r}(t,\mathbf{v}) := \mathrm{Rot}_{\mathbf{p}(t)}(\mathbf{v})$ (see (6.39)) describe the rotation by the unit-length quaternion $\mathbf{p}(t)$ of any coordinate triple $\mathbf{v} \in \mathbb{R}^3$ expressed in the frame $\underrightarrow{\mathscr{F}}_a$. Then the following equations defines the angular velocity $\boldsymbol{\omega}_a(t)$ in the same frame $\underrightarrow{\mathscr{F}}_a$ for this rotation of the coordinates $\mathbf{v}$:

$$\frac{\partial \mathbf{r}(t,\mathbf{v})}{\partial t} = \boldsymbol{\omega}_a(t)^{\times}\mathbf{r}(t,\mathbf{v})$$

Please note that we introduced $\mathbf{p}$ to represent the rotation *actively* in $\underrightarrow{\mathscr{F}}_a$ as opposed to the definition of $\mathbf{q}_{ba}$, which *passively* describes the change of coordinates from $\underrightarrow{\mathscr{F}}_a$ to $\underrightarrow{\mathscr{F}}_b$. We start this way because the physical notion of angular velocity is conceptually closer to the active rotation concept. With $\tilde{\mathbf{v}} := \mathbf{V}\mathbf{v}$ we have:

$$\begin{aligned}
\frac{\partial \mathbf{r}(t,\mathbf{v})}{\partial t} &= \frac{\partial}{\partial t}\mathbf{W}(\mathbf{p}(t)\tilde{\mathbf{v}}\mathbf{p}^{-1}(t)) = \mathbf{W}(\dot{\mathbf{p}}(t)\tilde{\mathbf{v}}\mathbf{p}^{-1}(t) + \mathbf{p}(t)\tilde{\mathbf{v}}\dot{\mathbf{p}^{-1}}(t)) && (6.75)\\
&= \mathbf{W}(\dot{\mathbf{p}}(t)\mathbf{p}^{-1}(t)\mathbf{p}(t)\tilde{\mathbf{v}}\mathbf{p}^{-1}(t) - \mathbf{p}(t)\tilde{\mathbf{v}}\mathbf{p}^{-1}(t)\dot{\mathbf{p}}(t)\mathbf{p}^{-1}(t)) && (6.76)\\
&= \mathbf{W}(\dot{\mathbf{p}}(t)\mathbf{p}^{-1}(t)\mathbf{V}\mathbf{r}(t,\mathbf{v})) + \underbrace{\mathbf{W}(-\mathbf{r}(t,\mathbf{v})\dot{\mathbf{p}}(t)\mathbf{p}^{-1}(t))}_{=\mathbf{W}(\dot{\mathbf{p}}(t)\mathbf{p}^{-1}(t)\mathbf{r}(t,\mathbf{v}))} && (6.77)\\
&= 2\mathbf{W}(\dot{\mathbf{p}}(t)\mathbf{p}^{-1}(t)\mathbf{V}\mathbf{r}(t,\mathbf{v})) && (6.78)\\
&= 2\mathbf{W}(\dot{\mathbf{p}}(t)\mathbf{p}^{-1}(t))^{\times}\mathbf{r}(t,\mathbf{v}) && (6.79)
\end{aligned}$$

As the latter equation has to be true for any $\mathbf{v}$ it follows that (omitting the time parameter)

$$\boldsymbol{\omega}_a = 2\mathbf{W}(\dot{\mathbf{p}}\mathbf{p}^{-1}). \tag{6.80}$$

In the case that $\underrightarrow{\mathscr{F}}_b$ is $\underrightarrow{\mathscr{F}}_a$ rotated by $\mathbf{p}$ it holds that $\mathbf{q}_{ba} = \pm\mathbf{p}^{-1}$. Inserting in (6.80) yields, when omitting the subscript of $\mathbf{q}_{ba}$

$$\boldsymbol{\omega}_a = 2\mathbf{W}(\dot{\mathbf{p}}\mathbf{p}^{-1}) = 2\mathbf{W}(\dot{\mathbf{q}^{-1}}\mathbf{q}) = -2\mathbf{W}(\mathbf{q}^{-1}\dot{\mathbf{q}}), \tag{6.81}$$

because $\dot{\mathbf{q}^{-1}} = -\mathbf{q}^{-1}\dot{\mathbf{q}}\mathbf{q}^{-1}$. This proves (6.43).

Angular velocity expressed in $\underrightarrow{\mathscr{F}}_b$ can be acquired by transforming the coordinates using $\mathbf{q}_{ba}$ by applying (6.42):

$$\boldsymbol{\omega}_b = \mathbf{W}(\mathbf{q}(\mathbf{V}\boldsymbol{\omega}_a)\mathbf{q}^{-1}) = -2\mathbf{W}(\mathbf{q}(\mathbf{V}\mathbf{W}(\mathbf{q}^{-1}\dot{\mathbf{q}}))\mathbf{q}^{-1}) = -2\mathbf{W}(\dot{\mathbf{q}}\mathbf{q}^{-1}), \tag{6.82}$$

because $\mathbf{V}\mathbf{W}$ acts as identity on the pure imaginary value $\mathbf{q}^{-1}\dot{\mathbf{q}}$. It must be pure imaginary because complex conjugation - denoted with a hat in the following - negates it as $\mathbf{q}$ is of unit-length ($\Rightarrow \mathbf{q}^{-1} = \hat{\mathbf{q}}$):

$$\widehat{\hat{\mathbf{q}}\dot{\mathbf{q}}} + \hat{\mathbf{q}}\dot{\mathbf{q}} = \hat{\dot{\mathbf{q}}}\mathbf{q} + \hat{\mathbf{q}}\dot{\mathbf{q}} = \dot{\hat{\mathbf{q}}}\mathbf{q} + \hat{\mathbf{q}}\dot{\mathbf{q}} = \frac{\mathrm{d}}{\mathrm{d}t}\hat{\mathbf{q}}\mathbf{q} = \mathbf{0}$$

The second equivalence follows from the fact that taking derivative commutes with complex conjugation. And (6.82) proves (6.44).

6.2. Proofs for unit-length quaternion exponential and logarithm maps' Jacobian formulas

Let $\boldsymbol{\phi} \in \mathbb{R}^3 \setminus \{\mathbf{0}\}$ represent a purely imaginary quaternion and $\phi := \|\boldsymbol{\phi}\|$, $\phi\mathbf{a} := \boldsymbol{\phi}$. First we derive the exponential map's Jacobian, $\mathbf{S}(\boldsymbol{\phi})$, as given in (6.47) by starting with (6.24).

$$\mathbf{S}(\boldsymbol{\phi}) = \sum_{k=0}^{\infty}\frac{(-1)^k}{(k+1)!}(\mathrm{ad}_{\boldsymbol{\phi}})^k \tag{6.83}$$

$$= (\mathrm{ad}_{\boldsymbol{\phi}})^0 + \sum_{k=1}^{\infty} \frac{(-1)^{2k-1}}{(2k-1+1)!} (\mathrm{ad}_{\boldsymbol{\phi}})^{2k-1} + \sum_{k=0}^{\infty} \frac{(-1)^{2k+2}}{(2k+2+1)!} (\mathrm{ad}_{\boldsymbol{\phi}})^{2k+2} \tag{6.84}$$

$$= \mathbf{1} + \sum_{k=1}^{\infty} \frac{-1}{(2k)!} ((-1)^{k-1} 2^{2k-1} \phi^{2k-1} \mathbf{a}^{\times}) + \sum_{k=0}^{\infty} \frac{1}{(2k+2+1)!} ((-1)^k (2\phi)^{2k+2} \mathbf{a}^{\times} \mathbf{a}^{\times}) \tag{6.85}$$

$$= \mathbf{1} - \frac{1}{\phi} \sin^2 \phi \mathbf{a}^{\times} + \frac{-1}{2\phi} \left(\sum_{k=0}^{\infty} \frac{(-1)^k}{(2k+1)!} (2\phi)^{2k+1} - 2\phi \right) \mathbf{a}^{\times} \mathbf{a}^{\times} \tag{6.86}$$

$$= \mathbf{1} - \frac{1}{\phi} \sin^2 \phi \mathbf{a}^{\times} + \left(1 - \frac{1}{2\phi} \sin(2\phi) \right) \mathbf{a}^{\times} \mathbf{a}^{\times} \tag{6.87}$$

We use to get (6.85) that for all $\mathbf{w} \in \mathbb{H}$, also purely imaginary, $\mathrm{ad}_{\boldsymbol{\phi}} \mathbf{w} = [\boldsymbol{\phi}, \mathbf{w}] = \boldsymbol{\phi}\mathbf{w} - \boldsymbol{\phi}\mathbf{w} = 2\boldsymbol{\phi} \times \mathbf{w}$, which implies that $2\boldsymbol{\phi}^{\times}$ is the matrix representation of $\boldsymbol{\phi}$'s adjoint action in $\mathscr{B}_{\mathfrak{g}} = (\mathbf{i}, \mathbf{j}, \mathbf{k})$, and that for the cross product it holds $(\mathbf{a}^{\times})^3 = -\mathbf{a}^{\times}$, which implies $(\boldsymbol{\phi}^{\times})^{2k+1} = (-1)^k \phi^{2k+1} \mathbf{a}^{\times}$ and $(\boldsymbol{\phi}^{\times})^{2k+2} = (-1)^k \phi^{2k+2} \mathbf{a}^{\times} \mathbf{a}^{\times}$ for $k \in \mathbb{N}$. To get (6.86) and (6.87) we only use well known power series expansions for $\sin^2$ and sin respectively.

Next, we prove (6.48) by showing that $\mathbf{S}(\log(\mathbf{q}))\mathbf{L}(\mathbf{q}) = \mathbf{1}$ for $\mathbf{q} \neq \pm j$. Let be $\boldsymbol{\phi} := \log(\mathbf{q})$, then

$$\mathbf{S}(\boldsymbol{\phi})\mathbf{L}(\mathbf{q}) = \left(\mathbf{1} - \frac{1}{\phi} \sin^2 \phi \mathbf{a}^{\times} + \left(1 - \frac{1}{2\phi} \sin(2\phi) \right) \mathbf{a}^{\times} \mathbf{a}^{\times} \right) \left(\mathbf{1} + \boldsymbol{\phi}^{\times} + \left(1 - \frac{\phi}{\tan \phi} \right) \mathbf{a}^{\times} \mathbf{a}^{\times} \right) \tag{6.88}$$

$$= \mathbf{1} + \left(\phi - \frac{1}{\phi} \sin^2 \phi + \frac{1}{\phi} \sin^2 \phi \left(1 - \frac{\phi}{\tan \phi} \right) - \phi \left(1 - \frac{1}{2\phi} \sin(2\phi) \right) \right) \mathbf{a}^{\times} \tag{6.89}$$

$$+ \left(1 - \frac{\phi}{\tan \phi} - \sin^2 \phi + 1 - \frac{1}{2\phi} \sin(2\phi) - \left(1 - \frac{1}{2\phi} \sin(2\phi) \right) \left(1 - \frac{\phi}{\tan \phi} \right) \right) \mathbf{a}^{\times} \mathbf{a}^{\times} \tag{6.90}$$

$$= \mathbf{1} + \left(-\frac{\sin^2 \phi}{\tan \phi} + \frac{1}{2} \sin(2\phi) \right) \mathbf{a}^{\times} + \left(1 - \sin^2 \phi - \frac{1}{2} \sin(2\phi) \frac{1}{\tan \phi} \right) \mathbf{a}^{\times} \mathbf{a}^{\times} \tag{6.91}$$

$$= \mathbf{1} \tag{6.92}$$

In the first step only $(\mathbf{a}^{\times})^3 = -\mathbf{a}^{\times}$ is used. In the last step we used $\frac{1}{2} \sin(2\phi) = \cos \phi \sin \phi$, $\tan = \frac{\sin}{\cos}$ and $1 = \sin^2 + \cos^2$.

6.3. Proof of (6.25)

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

$$= \mathbf{W} \frac{\partial}{\partial \boldsymbol{\phi}_l} (\bar{\mathbf{g}}_{k-1}^{-1} \exp(-\boldsymbol{\phi}_{k-1}) \exp(\boldsymbol{\phi}_k) \bar{\mathbf{g}}_k (\bar{\mathbf{g}}_{k-1}^{-1} \bar{\mathbf{g}}_k)^{-1})$$

$$
\begin{aligned}
&= \mathbf{W}\big(\bar{\mathbf{g}}_{k-1}^{-1}\big)^L (\bar{\mathbf{g}}_{k-1})^R \frac{\partial}{\partial \boldsymbol{\phi}_l} \exp(-\boldsymbol{\phi}_{k-1}) \exp(\boldsymbol{\phi}_k) \\
&= \underbrace{\mathbf{W}\big(\bar{\mathbf{g}}_{k-1}^{-1}\big)^L (\bar{\mathbf{g}}_{k-1})^R \mathbf{V}}_{=\mathbf{C}(\bar{\mathbf{g}}_{k-1}^{-1})} (\delta_{l,k} - \delta_{l,k-1}) \mathbf{1}
\end{aligned}
$$

6.4. Proof of Theorem 1

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

Proof. It is enough to show that $\cdot^L$ is an algebra-monomorphism into $\mathbb{R}^{\dim\mathscr{A}\times\dim\mathscr{A}}$ because its image, $\mathscr{A}^L$, is then a sub algebra and the mapping an algebra-isomorphism onto it. The existence of this isomorphism yields already that $\mathscr{A}$ is isomorphic to $\mathscr{A}^L$, that $\mathscr{G}^L$ is a multiplicative subgroup in it and its restriction to $\mathscr{G}$, $\cdot^L|_{\mathscr{G}}$, 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 $\mathscr{G}$ to be identical with the one induced from $\mathscr{A}$.

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

1. The *injectivity* basically follows from $\mathscr{A}$ having an 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 $\mathbf{v} \in \mathscr{A}$ be a further arbitrary vector.

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

$$(\mathbf{a}+\mathbf{b})^L\Psi(\mathbf{v}) \underset{\text{def}}{=} \Psi((\mathbf{a}+\mathbf{b})\mathbf{v}) = \Psi(\mathbf{av}+\mathbf{bv}) \underset{\text{def}}{=} \mathbf{a}^L\Psi(\mathbf{v}) + \mathbf{b}^L\Psi(\mathbf{v}) = (\mathbf{a}^L + \mathbf{b}^L)\Psi(\mathbf{v}) \tag{6.93}$$

   and

$$(\alpha\mathbf{a})^L\Psi(\mathbf{v}) \underset{\text{def}}{=} \Psi((\alpha\mathbf{a})\mathbf{v}) = \Psi(\mathbf{a}(\alpha\mathbf{v})) \underset{\text{def}}{=} \mathbf{a}^L\Psi(\alpha\mathbf{v}) = \mathbf{a}^L(\alpha\Psi(\mathbf{v})) = (\alpha\mathbf{a}^L)\Psi(\mathbf{v}). \tag{6.94}$$

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

$$(\mathbf{ab})^L\Psi(\mathbf{v}) \underset{\text{def}}{=} \Psi((\mathbf{ab})\mathbf{v}) = \Psi(\mathbf{a}(\mathbf{bv})) \underset{\text{def}}{=} \mathbf{a}^L(\mathbf{b}^L\Psi(\mathbf{v})) = (\mathbf{a}^L\mathbf{b}^L)\Psi(\mathbf{v}). \tag{6.95}$$

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

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

Because $\mathbf{a}$ and $\mathbf{b}$ were arbitrary it follows the homomorphy for the three algebra operations, the addition, the scalar, and vectorial multiplications.

□

6.5. Proof of Theorem 2

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

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

Acknowledgement

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

References

Paper IV

A Low-Cost System for High-Rate, High-Accuracy Temporal Calibration for LIDARs and Cameras

Hannes Sommer, Raghav Khanna, Igor Gilitschenski, Zachary Taylor,
Roland Siegwart, and Juan Nieto

Abstract

Deployment of camera and laser based motion estimation systems for controlling platforms operating at high speeds, such as cars or trains, is posing increasingly challenging precision requirements on the temporal calibration of these sensors. In this work, we demonstrate a simple, low-cost system for calibrating any combination of cameras and time of flight LIDARs with respect to the CPU clock (and therefore, also to each other). The newly proposed device is based on widely available off-the-shelf components, such as the Raspberry Pi 3, which is synchronized using the Precision Time Protocol (PTP) with respect to the CPU of the sensor carrying system. The obtained accuracy can be shown to be below 0.1 ms per measurement for LIDARs and below minimal exposure time per image for cameras. It outperforms state-of-the-art approaches also not relying on hardware synchronization by more than a factor of 10 in precision. Moreover, the entire process can be carried out at a high rate allowing the study of how offsets evolve over time. In our analysis, we demonstrate how each building block of the system contributes to this accuracy and validate the obtained results using real-world data.

Published in:
IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2017
doi:10.1109/IROS.2017.8206042

1. Introduction

Multi-sensor systems that involve cameras and LIDARs, e.g. for motion estimation, are ubiquitous in robotics research. An easy to use, fast and reliable method to acquire accurate and high frequency estimates about when sensors actually take measurements and the reliability of their hardware timestamps, while being independent of any other spatial or intrinsic calibration, has the potential to save a lot of tedious calibration work and errors. This is particularity true for research systems that contain large numbers of different sensors. Furthermore, as the underlying algorithms and concepts for such multi-sensor systems grow more mature, these systems are about to hit the market in diverse end-user products. An example of this is in the automotive field, where a combination of multiple cameras and LIDARs are typically used for autonomous driving and advanced driver assistance systems. Furthermore, recent developments in LIDAR technology and strong interest from industry, promise wider availability of this sensor type and a further reduction in prices. This means that the availability of multi-sensor setups is likely to significantly increase in the future. When these setups are utilized on high speed platforms (such as cars, trains or multi-copters), or used to jointly perceive highly dynamic environments, precise time synchronization becomes a critical aspect. Experience with multi-sensor fusion shows that un-modeled delays can have a strong impact on the performance of motion estimation systems [136]. Unfortunately, most multi-sensor systems are not always hardware synchronized, particularly when low-cost sensors and commodity personal computers are used. And even if hardware synchronization is available, there may still be unknown offsets with respect to the synchronization signal.

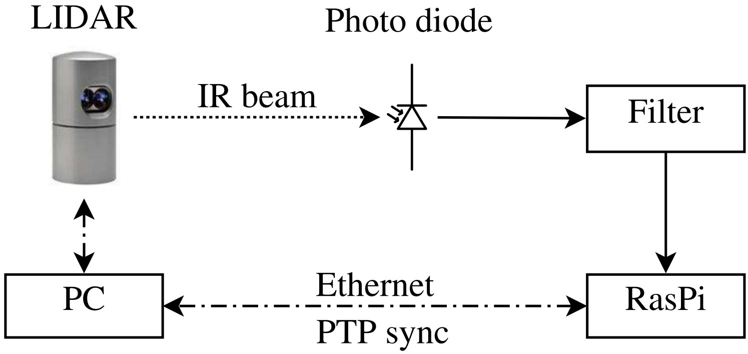


Figure 7.1.: The setup for LIDAR-to-CPU temporal calibration. When a LIDAR beam hits the photo diode a hardware interrupt is triggered on the Raspberry Pi and a low latency timestamp is assigned and sent to the PC.

The current state-of-the-art in multi-sensor timing calibration usually addresses the problem either by exploiting co-observed motion [84, 154] within self-calibration approaches, or using hardware synchronization signals [49, 136]. Motion based methods require a large amount of data to achieve millisecond precision (down to about 0.1 ms with targets such as a checker board, in laboratory environments [136]) and their accuracy is highly dependent on the motion. They also have no means to observe the offset between a sensor and the computer's system time. Hardware signal based methods demand suitable sensors that support this functionality (excluding many low cost camera systems, among others) and additional wiring to every sensor. Our work targets cases where hardware synchronization is not possible or too expensive, or where additional means for validation or inspection are required. This work is motivated by the observation that the use of simple external devices involving LEDs (for cameras) or photo diodes (for LIDARs) can result in significantly improved performance in comparison to motion based approaches. While this comes at the cost of an extra device, the gain in high rate and high accuracy estimates of the timing offsets can be particularly useful to optimize the exploitation of hardware or receive timestamps. Our approach can also serve as an accurate benchmark (ground truth) for better / easier evaluation of self-calibration algorithms. The proposed device is based on a Raspberry Pi 3 which is connected either to several LEDs for temporal camera to CPU calibration or a photo diode for LIDAR to CPU calibration. PTP is used for precise time synchronization between the calibration device and the CPU of the system to which the sensors are attached. Overall, this work makes the following primary contributions

- A system for calibrating the time offset between the actual camera exposure taking place and the corresponding image timestamp (hardware clock and arrival).
- A system for calibrating the time offset between LIDAR timestamps and actual measurements.
- An evaluation of the accuracy and precision obtained using the two systems and 5 popular sensors, and validation for both based on hardware signals.
- A free and open source, ROS compatible C++ library to simplify hardware time related tasks for authors of device drivers, such as translating timestamps, and providing the user with tuning and inspection options. Inspection tools and several modified device drivers are also provided.[1]

2. Related work

To the best of our knowledge very little has been published about temporal sensor calibration using external devices. The only example we are aware of is [140] in which the authors use a single LED flashed at known system time (through a microcontroller) together with a high speed camera recording at 100 Hz This could be considered a simpler version of our LED-signaler device. The fundamental problems with a single LED are first, the achievable accuracy is limited by the maximum of exposure time and LED flash duration - as for most simple approaches with LEDs, and second the smaller this maximum is, the less probable it is that a LED is flashed during the camera exposure.

Two-way synchronization methods like TICSync [69], PTP [48], or NTP perform well but require support by the sensor's firmware, which is rare and expensive.

---

[1] https://github.com/ethz-asl/cuckoo_time_translator/ (BSD-3-Clause)

TriggerSync [49] operates by exploiting co-observation of simultaneous trigger signals. This is a great practical solution for online sensor synchronization if the sensors have trigger or synchronization inputs and very accurate when one of the sensors has very low latency or the main CPU has low latency direct inputs such as a RS-232 port [130]. Unfortunately many setups do not match these requirements. This is especially true in the case of LIDARs. Furthermore, the estimation rate is limited for similar reasons to the single LED approach above, due to the ambiguity that occurs when the trigger rate is high. This problem makes it a) even more restrictive regarding the sensors that can be used, when they are supposed to take measurements at a higher rate than the trigger rate, and b) less useful as a tool to analyze the statistical properties of the individual delays. A further difference is that TriggerSync must rely on the sensors handling hardware synchronization correctly, whereas our approach is directly based on the physical act of taking a measurement.

Target-less self-calibration methods such as described in [154] or [84] rely on the same motion being observed in different sensors to find their relative offsets. This typically require many seconds of sensor data to reach a precision of milliseconds. This limitation means that these approaches cannot analyze the short term stability of the transport and processing delays or accuracy / stability of sensor's hardware timestamps. Furthermore, they cannot provide insight into the offset between CPU and sensor time.

The approach presented in [136] can achieve very high accuracy for temporal calibration LIDAR to camera by supporting the motion estimates with a checkerboard in the camera's field of view. However, it still requires much more data (thousands of LIDAR measurements and hundreds of images) to reach the precision for the *average* delays that our approach can yield for each single image or LIDAR pulse / revolution. Given that much data, the variance of the average offset estimate per measurement can be much smaller, bounded from below by the variance of the physical delays within that data. For example, if we assume 20 s of data from a LIDAR spinning at 50 Hz our method accumulates 1000 delay estimates, each of which has a random error with a standard deviation (SD) of about $\sigma := 1\,\mu\text{s}$. Assuming independence[2] we get a theoretic SD of the estimated average delay of about $\sigma/\sqrt{1000} \simeq 32$ ns. Typically, the true average delays over 20 s are far less stable. However, this instability comes from variations in the actual delays, something our method allows an user to inspect. Please note that this is not about the method's accuracy — only its precision. Nonetheless, is the right quantity to compare with competing methods, for which typically only their precision is published due to fundamental problem of obtaining precise ground-truth data for calibration. Another benefit of the simplicity and immediacy of our method is that we can provide small upper bounds for the actual inaccuracy.

Another typical problem most motion based approaches suffer from is that they do not have the ability to analyze the delays and hardware clocks independently of other calibration parameters such as spatial extrinsics or even sensor intrinsics. Solving the calibration problems jointly does not prevent high precision, in fact it might even improve it [136]. But it typically makes the problem more complex and fragile as well as making it more sensitive to the quality of the model and of the prior calibration.

---

[2] Assuming independence is probably to coarse considering the Raspberry Pi we employ for our experiments. But slightly more expensive hardware can almost diminish this source of inaccuracy — if needed.

3. Design

The core idea of our sensor-to-CPU temporal calibration approach is to create a small, low cost device that allows the CPU to a) detect the event of taking a measurement for active sensors, such as LIDARs, or b) trigger physical events that can be detected and identified in the measurements of a passive sensor, such as a camera. In order for the calibration to be accurate it is necessary that the design of the device yields low unknown latency between an event and its detection. In this paper we present two such devices: a *LIDAR beam detector* (3.1), and a *LED signaler* (3.1) for laser-CPU and camera-CPU temporal calibration respectively. For both devices it is essential to have accurate timestamps for edges of a digital electrical signal. We assume here that the computer receiving the sensor measurements (PC) does not have an appropriate input. We solve this by connecting a Raspberry Pi 3 Model B with 1200MHz ARM CPU (RasPi) via Ethernet, and synchronize it's clock with the PC using the *Precision time protocol* (PTP, see 3.2). We use its general purpose input output (GPIO) pins to directly trigger an interrupt on its ARM CPU. In order to handle these interrupts at low latency, we use a specifically developed kernel module on the RasPi. In our setup Linux is utilized on both machines. In case the PC does have such an input (e.g. the Data Carrier Detect line of a RS-232 port as used in [49]) the approach is even easier to realize because no RasPi or PTP synchronization is required.

3.1. Calibration-devices

In the following sections we outline the two devices developed and used to perform the calibration.

LIDAR beam detector

The *LIDAR beam detector*, is a photo diode based detector of LIDAR pulses. This detector is used as depicted in Figure 7.1 to allow highly accurate and high rate *time of flight* (TOF) LIDAR-to-CPU temporal calibration.

Working principle Whenever a laser pulse of the TOF-LIDAR hits the photo diode the impact gets detected through the filter circuit in the RasPi and a timestamp of the Linux system time, or for this paper *CPU-time* (CT), is recorded and sent via TCP to the PC. As the RaspPi's CT is synchronized via PTP to the PC's CT it is possible to align these timestamps directly with the receive timestamps of the LIDAR for any particular measurement. For this to work it is necessary to place the photo diode at a place where a beam hits it and the corresponding beam's bearing needs to be retrieved. To retrieve the bearing we manually selected a point in the LIDAR's point cloud using the live data visualization capability of RVIZ, a visualization tool provided by the Robot Operating System (ROS)[131]. As the bearing stays constant over time when both the LIDAR and the photo diode are stationary this step only has to be done once per LIDAR for a series of delay estimations.

Circuit The entire detector circuit we used is an IR-photo diode, 900 nm, SFH 203 FA, Osram Semiconductors connected with its anode to the base of a BC547 transistor, whose emitter is grounded, while its cathode is pulled up to 5V through a 10 kΩ resistor. The transistor's collector is pulled up with a 2 kΩ resistor to 5 V. The signal at the collector is sent through a Schmitt trigger (74HCT14) and a 10 kΩ resistor to the GPIO of the RasPi to trigger the interrupt.

Data assignment The periodic nature of the measurements (e.g. per revolution of the spinning LIDAR) yields an assignment problem, as the beginning of the two measurement series (LIDAR and photo diode) may not coincide. This yields an ambiguous latency by multiples of the time for one LIDAR revolution (typically $\gtrsim$ 20 ms). In order to resolve this ambiguity we manually block the path from the LIDAR to the photo diode for a few periods. This blocking event is easy to automatically detect, as it appears as gaps in the photo-diode time-series data along with unusually short ranges measured by the LIDAR. Having detected this in both the photo-diode and laser time series we compute a relative shift for the laser time series which aligns its shorter range data to the gap in the photo diode time series. A shift is enough for the entire data assignment after gaps in both series have been filled with placeholders for missed packages from the LIDAR or undetected events on the photo diode. This can be done for both the laser and photo diode time-series based on the approximate LIDAR revolution period. After applying the correct shift to the dataset containing the blocking event a good first estimate for the transport delay can be retrieved. This first estimate may then be used to resolve the ambiguity for future datasets since it only needs to be accurate to within one revolution of the sensor. For the same reason this step is *unnecessary* in case of sufficiently accurate prior knowledge.

Accuracy The detector itself is not without delay. However, is possible to retrieve bounds for the delay. First an upper bound for the delay from when the diode the diode detects the laser until the RasPi takes a timestamp can be found by employing an IR-LED to periodically trigger the photo diode. This can then be compared to the trigger signal via the use of a GPIO output of the RasPi. This is done by changing the output pins value whenever the RasPi receives an interrupt and after taking the timestamp using a suitable oscilloscope. The oscilloscope is only used here for validating our method, and is not necessary for application scenarios. With our setup the overall delay from the positive edge of the IR-LED current to the RasPi feedback was randomly distributed between 5 and $10\,\mu$s. Comparing the induced photo current in the photo diode between when it was triggered with the IR-LED and when a LIDAR beam hits it (Figure 7.2) using an oscilloscope clearly shows that the first signal edge is even steeper for the laser beam. Therefore, we assume an upper bound for the delayed LIDAR pulse arrival to RasPi timestamp of $10\,\mu$s. The synchronization error between RasPi and the PC is within approximately $\pm 20\,\mu$s according to the internal assessment of the ptpd[3]. In total we expect the absolute error for our setup per **single** laser pulse to be within $[-20, 30]\,\mu$s. The concept has the potential for nanosecond accuracy when used with more expensive hardware.

LED signaler

The *LED signaler*, is comprised of a LED display with 10 large LEDs that are sufficiently bright to be visible in the camera images, even with very short exposure time (e.g. 0.1 ms). These LEDs constitute a digital "clock" employing a special redundant encoding to prevent false readings through the camera images. The LED Signaler is employed as depicted in Figure 7.3 to allow

---

[3] `https://sourceforge.net/projects/ptpd/`

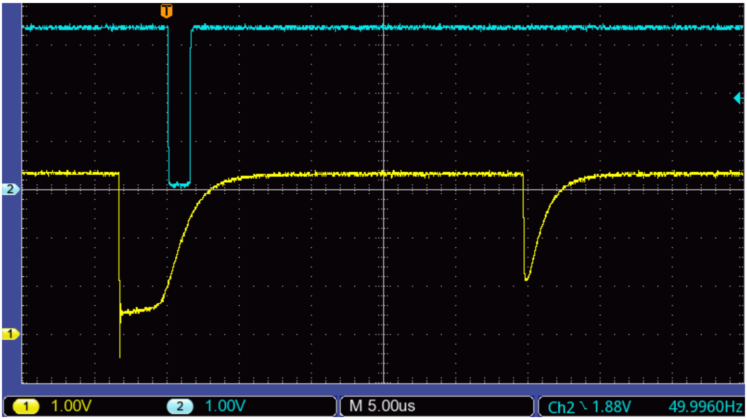


Figure 7.2.: The photo current of the photo diode (yellow, measured as voltage at a 10 kΩ resistor) caused by the impact of two consecutive laser pulses of a LMS151 spinning at nominal 50 Hz and the feedback signal of the RasPi after taking the timestamp (cyan). It shows the typical delay about 3.2 μs of the feedback. The second valley of the yellow curve corresponds to the subsequent laser pulse from the LIDAR. In the RasPi we prevent in software (based on the temporal distance) to take another timestamp at the second (no feedback).

high rate camera-to-CPU temporal calibration with an expected accuracy per measurement of approximately the minimal exposure time of the camera.

Working principle The principle here is to observe the state of the LED “clock” in the camera images and use this state to associate the image timestamps with the RasPi timestamps of a corresponding state. Our setup consists of a LED display comprised of 10 LEDs in a row. The two outermost LEDs are always kept on in order to predict the locations of the inner LEDs. The 8 inner LEDs (0..7) are driven by a digital 6-bit counter at an user defined frequency, and define the state of the display. The 4 most significant bits directly control LEDs 4..7, while the two least significant bits control the other 4 LEDs, such that exactly only one is on at a time and its index corresponds to the binary number encoded with the two bits. Hence, the $(4+4)$ LED setup can display $2^6 = 64$ states. The digital counter that controls the LEDs also triggers an interrupt on the RasPi through one of its GPIO ports, whenever the LED starts showing a “0”. Similar to the LIDAR beam detector setup, the RasPi assigns a timestamp to such events using the Linux CT, which is recorded and sent via TCP to the PC. These “0” timestamps are then used to compute timestamps for every state of the LED display, assuming a constant frequency between two “0”s of the LED display. To generate the input signal for the clock-counter we used our kernel module running on the RasPi, but it could be any source.

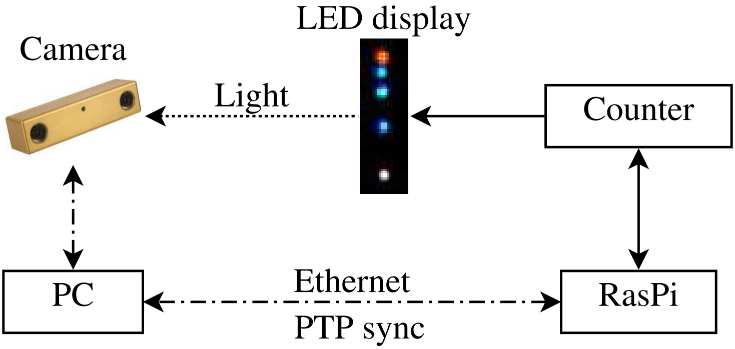


Figure 7.3.: The setup for Camera-to-CPU temporal calibration. Every image taken by the camera can be located in time based on what LED state is displayed. The LED status is controlled by the Raspberry PI with little unknown latency.

Data Assignment To determine the timestamp of an image to a precision of the camera exposure time (τ_e), the LED display must be driven at a frequency of ($\frac{1}{\tau_e}$). However, this leads to images where a transition in the state of the display takes place during an exposure. This leads to a data assignment problem as the state transition with one LED turning OFF and another turning on during an exposure may be incorrectly detected as a state with both LEDs ON. In order to remove such frames from the data association we represent the two least significant bits of our LED clock by 4 LEDs (see above). When 2 of them appear ON in a camera frame there is only one possible transition that can cause this, and the time in the middle between the two states is estimated for the image. In addition, frames showing the transitions between LEDs 0 and 3 are skipped since these imply a change in state of one of the more significant bits, whose observed state might then not be distinguishable from state transitions. The so encoded number on the LED display in any given camera frame is calculated in an automated manner using a simple computer vision blob detector algorithm. This allows us to align the camera timestamps with the clock timestamps with an ambiguity of the period of the entire LED-clock (the duration after which it repeats itself). In order to resolve this ambiguity, a box prior for the average total processing and transmission delay per image is required with an uncertainty smaller than the period of the LED clock. This poses no problem because the period is adjustable through the clock frequency and one can start with a low frequency e.g. 500 Hz which requires the uncertainty to only be below $64/500$ s $= 128$ ms. One pass with this frequency reduces the uncertainty to 2 ms. Such a prior is good enough for our goal frequency of 10 kHz, which can be used in a next pass.

Circuit The schematic of the circuit driving the LED display is too much detail for the paper. But it is very simple to design once the necessary LED pattern is known. For our prototype we used two 74HCT191, 4-bit counters chained together to an 8-bit counter, a 74LS42 to demultiplex the two least significant bits, and a HCF4069 to invert the output signals of the 74LS42. Additionally we use one BC547 transistor per LED to amplify the output current of the digital ICs and resistors to adjust the current through the transistors and LEDs, and some capacitors to stabilize the signals. One input of the RasPi is connected to the 7-th bit of the counter to notify about newly reached “0” display states.

Accuracy Our approach limits the accuracy of the estimated image timestamps to within the exposure time of the camera. Hence, to obtain accurate measurements, the exposure time should be set to a small value and the LEDs chosen should be bright and close enough to be clearly visible in the image. The LED clock’s base frequency (rate of a single clock increment) should be about the inverse of the camera’s exposure time. A higher clock frequency would observe the average effect of multiple LED transitions during one exposure and a lower frequency limits the accuracy achievable with a single image. For our setup we set the camera exposure time to 0.1 ms and the LED clock frequency to 10 kHz leading to an absolute error for each image timestamp to be within $[-0.05, 0.05]$ ms.

3.2. Precision time protocol (PTP)

The Precision time protocol (PTP, [48]) is a protocol for two-way synchronization between computers or other network nodes and from the user perspective similar to NTP. In contrast to NTP it is designed for local area networks and aims at much higher accuracy. Up to few nanoseconds when used with hardware support in the network devices. Even without any hardware support it typically maintains synchronization within a few microseconds. To synchronize the main PC with the RasPi we used ptpd[3] without using any hardware timestamping because the RasPi’s Ethernet adapter does not support hardware timestamping.

3.3. One way clock translation

In order to exploit the *device time* (DT) measured by clocks within the sensors for sensor fusion or control it is necessary to translate it into CPU-time (CT). The same is necessary in order to use the calibration devices presented in this paper because they measure the sensor activity effectively in the CT. This is typically done by a suitable filter / batch algorithm fusing the receive CT-stamps with the DT-stamps in the received packages. Therefore we are referring to the resulting time with *translated device time* (TDT). To filter the DT-stamps we used our free software implementation[4] of the convex hull algorithm presented in [164]. This is identical to what was used in [136] and equivalent to the TICSync implementation used in [49] with the difference that the latter’s authors are using an estimator switching periodically between two of the convex hull filter, which get reseted whenever their turn is over to prevent relying on too old data and therefore be more resilient against long term drifts. For our experiments we also use a switching version of the convex hull algorithm. To retrieve the offset to this TDT with respect to the CT (it is always lagging behind, roughly by the minimal unknown transport delay) is typically the goal of temporal calibration.

With the devices presented here it is possible to go much further due to their high rate high accuracy nature. They enable the user to monitor the performance of the hardware clock filter itself on data generated from the real target hardware clock. This is useful for tuning,

---

[4] Available as part of `https://github.com/ethz-asl/cuckoo_time_translator/`

developing and debugging hardware clock filters directly for a given sensor possibly taking relevant influences such as sensor temperature into account. These tasks are typically hard problems unless one is using hardware synchronization, which we assume to be no option.

In Section 4 we are going to show two versions of TDT for the same raw data for the reader to better appreciate the problem.

4. Experiments and Results

4.1. Setup

We evaluate our method using the following sensors:

- two different Sick LMS151 LIDARs spinning at 50 Hz using a 100 MBit Ethernet connection,
- two Hokuyo LIDARs, UTM-30LX (USB2; 2 hubs) and UTM-30LX-EW (100 MBit Ethernet),
- a Point Grey BB2-08S2C-25 (Firewire; 1 hub),
- a Point Grey Chameleon3 CM3-U3-13Y3C (USB3; 1 hub).

We recorded data from these sensors while estimating the time until the measurements are received on the main PC. All sensors connected via Ethernet communicate through two gigabit Ethernet switches with the PC. All sensors are connected and running in parallel, together with a Velodyne 32E and one more LMS151 within a complex robotic system. However, only one sensor is analyzed at a time. Each LIDAR sensor is recorded for 60 s and each camera for 20 s and the estimated arrival delays and offsets to their TDT are presented. The Chameleon3's strobe signal and the UTM-30LX's "synchronous output" are additionally connected to GPIO ports of the RasPi and the timestamps of their rising edges are recorded in parallel to provide ground truth to validate our methods.

We use ROS [131] as the middleware to interface with the sensors. To our surprise all four required open source ROS-drivers lacked the support for DT-stamps. In order to be able to present the hardware clock offsets as part of our analysis, these drivers were extended to allow the hardware clocks to be used. We made this work also publicly available[1].

4.2. Evaluation

All the figures in the following section follow a common layout:

- The x-axis reflects the CT of the plotted events shifted such that the first measurement is always at zero.
- The y-axis is a time offset between two CT-stamps associated with a common event, of which the reference time is always a time determined with one of the calibration devices by the synchronized RasPi.

- Red points correspond to the overall transport and processing delay of a measurement package (camera image / LIDAR ranges), i.e. the CT of arrival at the main PC (*receive time*) of a sensor measurement minus the corresponding CT of the *first* measurement in the data package.
- Green points correspond to offsets of TDT-stamps, i.e. the one way TDT minus the CT of the corresponding detection.
- Optional blue points correspond to the offset of hardware signals coming directly from the sensor (e.g. strobe) to the RasPi for validation of our method.

4.3. LIDAR-to-CPU

For the LIDAR-to-CPU temporal calibration experiments we use the *LIDAR beam detector* device to generate timestamps in CT (by the means of PTP synchronization between the main PC and RasPi) for the detection of LIDAR pulses on the photo diode. For all the experiments one event is measured per revolution of the spinning sensor as one photo diode is used and only the first detection per revolution is recorded (based on the roughly known sensor period)

Figure 7.4 shows results for the UTM-30LX and the UTM-30LX-EW, each spinning at 40 Hz. The specific pattern of the TDT-stamps (green) is caused by the remainder of the sensor's revolution time modulo the milliseconds counted by their internal clocks. The UTM-30LX data indicates a very stable revolution period. It yields almost a fixed fraction of a millisecond that it looses every revolution until it has lost an $\varepsilon > 0$ more than a millisecond, which corresponds to only ε — modulo the one millisecond resolution. The high stability of the revolution period can be exploited to get timestamps of lower variance than a naive translation of the low precision DT, as suggested by [136]. A very simple way to do this is to use the revolution counter as a "sensor clock" instead of its timestamps. This clock's "time" can be translated with the same type of algorithms into CT. We show in Figure 7.4 (black) the offsets of the revolution counter after translating it with the same convex Hull algorithm we use for the green plots. This method is not reliable enough for real world application but it yields a good way to asses the stability of the revolution time. The UTM-30LX-EM shows similar effects, with the differences coming from a less stable revolution period.

To validate the performance of the LIDAR beam detector we retrieve additionally CT-stamps for the rising edge of the UTM-30LX's "synchronous output" (also in Figure 7.4, blue). This demonstrates high precision. However, the mean (2.607 ms) only roughly corresponds to the 2.75 ms delay specified by the manufacturer (as given in [136]). Because of this significant discrepancy, discrepancy, we must rely upon the oscilloscope (Figure 7.2) and the corresponding camera test in Figure 7.6 for the accuracy. Furthermore, we found that this delay changes over time between 2.56 and 2.62 ms, qualitatively corresponding to the drift in rotation frequency from 40.2 to 39.9 Hz. This indicates that the specification of 2.75 ms is indeed inaccurate.

Figure 7.5 shows the result for 20 s of data from the LMS151(b), spinning at 50 Hz [5]. These sensors emit two timestamps from the same clock with each package. One timestamp for the measurement start and one for the transmission start. The latter can be used to improve the

---

[5] The corresponding plot for the LMS151(a) is not included because it is very similar including the very surprising hardware clock behavior (see caption).

clock translation as it happens very close to the physical receive time. Using this improved translation for the start timestamp yields better accuracy and a CT that is in deed close to the actual measurement start, without relying on assumptions about the measurement and processing duration as can be seen in Figure 7.5 (green). The remaining negative offset comes from the fact that according to the specification, the start timestamp is taken internally at 14 degree before the actual measurement start. This corresponds to -0.77 ms, which is very close to the mean of the translated start time (-0.71). The remaining difference is probably due to the random 1 ms jumping of the green signal. To our great surprise we had to assume here that the start-timestamp delivered along with the measurements from one measurement phase actually marks the beginning of the next measurement phase[6].

Statistical data for all these datasets is presented in Table 7.1. Each is based on 60 s of data per LIDAR, of which only the beginning may be plotted in the corresponding figures for the sake of readability. The two numbers for the receive latency's mean correspond to first and last measurement of one package. Both are inferred based on the measurement for one beam, its index, and the duration of a full measurement phase. Please note that all these numbers will not translate well to other setups and systems because the sensors are only a part of the communication. This holds in particular for the means. Furthermore, they depend on the revolution period, which typically drifts significantly over time and possibly depends on the motion / alignment of the sensor with respect to gravity. Additionally, for the Hokuyo sensors it depends on how the internal bandwidth limit[7] is overcome: skip revolutions or restrict the measurement angles range. We use the latter, which also leads to smaller packages that typically have less random transport delays than larger packages. Nevertheless, comparing the different LIDARs, it becomes apparent that the two LMS151 are much quicker at delivering the data through Ethernet than the UTM-30LX-EW (about 7 ms vs. 14 ms minimal latency). But at the same time their delay is less predictable (0.14 ms vs. 0.04 ms). The UTM-30LX connected with USB2 is even quicker (minimal latency mean 3.6 ms). However, on the USB2 bus we observed quite frequent and rather severe outliers in the latencies, up to ~ 10 ms. Without specialized DT translators all presented LIDARs have more accurate receive time. However, a) more recent / expensive models are likely to have better DT-stamps (e.g. the Velodyne HDL-32E, whose analysis is beyond the scope of this paper), and b) in applications where larger outliers must be avoided it might still be better to utilize the DT-stamps as they can be more reliable, especially in the case of a dedicated data connection.

Our results clearly indicate that to analyze the performance of a LIDAR's hardware clock and to tune, develop and evaluate new filters for it, the LIDAR beam detector is a very useful device because it provides the user with the exact feedback needed: a reference timestamp to compare with for a real device.

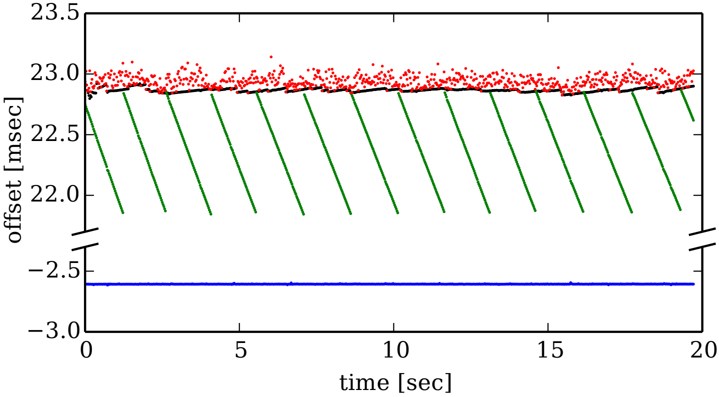
4.4. Camera to CPU

For Camera-to-CPU temporal calibration experiments we use the LED Signaler device to generate timestamps in CT (by means of RasPi-CPU PTP synchronization) of each state of the LED display. We obtain a delay or offset for each image coming from the free running

---

[6] Assumed it to mark the beginning of the same data the transmission of a package containing 15 ms measurements would happen < 3 ms after the first measurement.

[7] The sensors are not able to deliver at full rate over the full angular range.



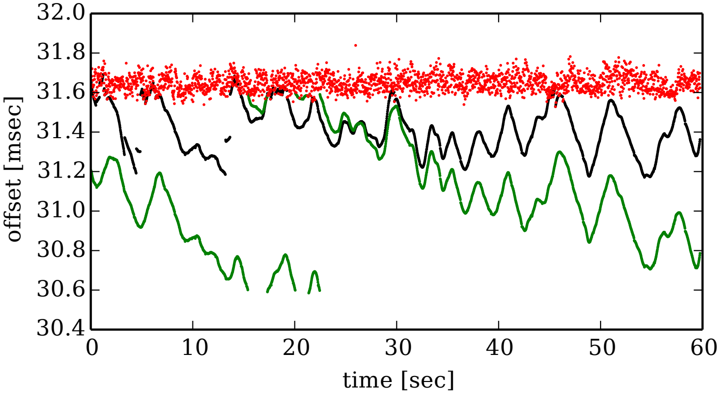


Figure 7.4.: Time offsets for the UTM-30LX (top) and UTM-30LX-EW (bottom). The latency (red) shows the typical pattern of random delays caused on the USB bus and in the OS kernel. The translated hardware clock offset (green) nicely shows the effect of DT taken with too low precision (1 ms; less precise than the revolution period). This lack of precision makes it less accurate than the receive time (SD $\simeq 50\,\mu$s) when used with a simple hardware clock translator (green, SD $\simeq 290\,\mu$). However, it is possible to improve on this by exploiting the precision: black (SD $\simeq 17\,\mu$). The UTM-30LX-EW's corresponding plot (bottom, black) shows the limitations of this "trick". It's higher variance (SD $\simeq 120\,\mu$s) indicates that this sensor's revolution period is less stable. The delay from the rising edge of the "synchronous output" of the UTM-30LX until the first measurement as detected by the photo diode (blue) shows very low variance (SD $\simeq 1\,\mu$s). This demonstrates the high precision of our method. It is not close to 0 because it is not synchronized with the first measurement but with a specific angle before it. Please note the break in the y-axis of the plot.

Sensor	receive latency		trans. hw. clock off.	
	mean [ms]	SD [ms]	mean [ms]	SD [ms]
UTM-30LX	21.9 − 3.9	0.085	21.3	0.289
...-EW	31.6 − 13.6	0.042	31.1	0.257
LMS151(a)	21.6 − 6.6	0.174	−0.7	0.498
LMS151(b)	21.6 − 6.6	0.142	−0.7	0.491
Chameleon3	13.8	0.712	11.6	0.020
Bumblebee2	51.7	0.116	51.3	0.023

Table 7.1.: Latency and Offset statistics for the tested sensors

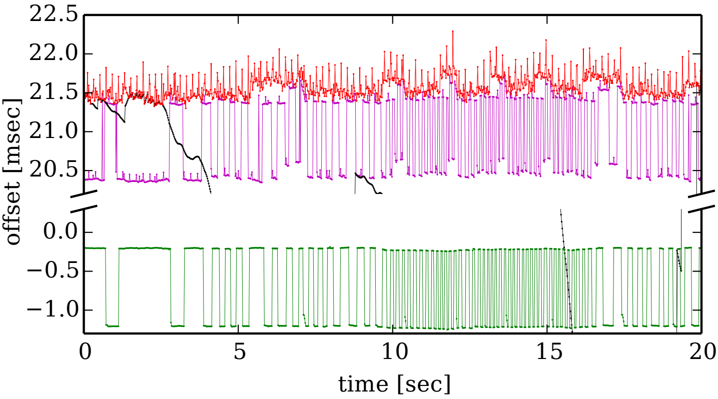


Figure 7.5.: The LMS151(b) receive latency's (red) show the expected random delays (mean $\simeq 21.6$ ms and SD $\simeq 0.14$ ms). The TDT (green) shows very surprising behaviour. It seems to jump between two hardware clocks that are almost precisely 1 ms apart. Each of these apparent clocks is very precise ($< 10\,\mu$s SD between the jumps; the DT resolution is $1\,\mu$s). Because of these unpredictable jumps the overall randomness of the DT-stamps has a standard deviation of approximately 0.5 ms, which is significantly larger than the SD. of the random transport delays. Using the revolution counter as a clock (black) is no alternative. Instead it reveals significant short term instability of the rotation period. The magenta plot shows the translated transmission timestamp. Its existence allows the start timestamp (green) translated with the same model to be close to the actual measurement start. Please note the break in the y-axis of the plot.

camera (Chameleon3 or Bumblebee2) by detecting the state of the LED display in the image and associating it to a corresponding state of the display as described in section 3.1. The camera exposure times are set to 0.1 ms, which should limit absolute errors to within $[-0.05, 0.05]$ ms for each measurement. Thanks to the short exposure time normal office lighting does not pose any problem for the procedure. Typically only the LEDs are bright enough to be visible in the images[8].

Figure 7.6 shows the delay and offset plots for a 20 s dataset collected using the Chameleon3. In addition to recording the image arrival and TDT-stamps, we also recorded the timestamp for the strobe signal for each image directly in RasPi time. As seen from the figure, the **difference** between estimated image timestamps using our method and the strobe signals (blue) has a mean of only $10\,\mu$s (SD $19\,\mu$s), clearly validating our method. Furthermore, we can observe that the variance of the image arrival timestamps (red) over the dataset is much greater than the translated image DT-stamps (green), reflecting the variable delay in image transmission. Table 7.1 shows statistics providing a comparison between image arrival timestamps and TDT-stamps for both cameras studied during this work. Figure 7.7 shows similar plots for the delays and offsets of the Bumblebee2 camera but without strobe signal. The offsets plotted in purple show the typical bad performance of a freshly started convex hull filter. Only after some time it obtains steady values for the delays (green), hence the particular switching concept described above, which allows a filter to mature before being used. Long term clock drifts (acceleration / deceleration) are not captured by the employed convex hull filter, and hence a gradual drift in the offset values can sometimes be observed in longer datasets, such as the one shown in Figure 7.8. Against this problem the reset part of the switching concept is a good remedy. Switching was disabled for this last experiment.

Since our method provides an offset almost for each image in contrast to batch processing based approaches [136, 154], it is an useful tool to study such effects for a variety of sensors.

5. Conclusions

We presented a novel approach for temporal calibration of LIDARs and cameras with respect to a connected computer's system time that offers high accuracies at a high rate. And we showed with 5 different sensors that it can easily provide insights into the nature of the sensor delays that are difficult to retrieve otherwise. The presented solution promises to be relevant to all applications that require low timing uncertainty when utilizing these sensors, such as high speed state estimation and control. The proposed methodology is particularly useful in the detailed analysis of the temporal offsets to leverage optimal hardware time filters or investigate causes for random transfer delays. We expect the contribution of this work to be most useful for robotic researchers and industrial users relying on these sensor types without having the resources for custom hardware development or hardware synchronization. To validate our methods we compared our results with the hardware synchronization signals of a camera and a LIDAR and measured the delays of the LIDAR beam detector using an artificial IR source to simulate laser pulses from LIDARs.

---

[8] The LED display image in Figure 7.3 was taken in a bright office and behind the LEDs there was white plastic.

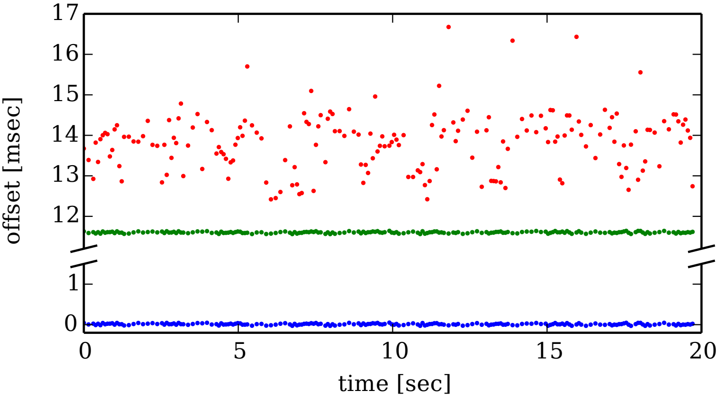


Figure 7.6.: Chameleon3 time offsets estimated using the LED signaler. As hoped, the strobe signal time offsets (blue) have almost zero mean (10 μs; SD 19 μs; both below exposure time, 0.1 ms). Hence they validate the method. The standard deviation of the receive latencies (red; SD 0.7 ms) is observed to be much higher than the offsets of the translated image timestamps from the camera hardware clock (green; SD 20 μs). Data collected over 20 s, 192 images (+66 skipped). Please note the break in the y-axis of the plot.

Acknowledgments

This work was supported by the European Union's Seventh Framework Programme for research, technological development and demonstration under the EUROPA2 project No. FP7-610603, the European Union's Horizon 2020 research and innovation programme under grant agreement No 644227 (FLOURISH) and from the Swiss State Secretariat for Education, Research and Innovation (SERI) under contract number 15.0029. The authors would also like to thank Jörn Rehder for the very helpful discussions and Mark Pfeiffer for the code contributions!

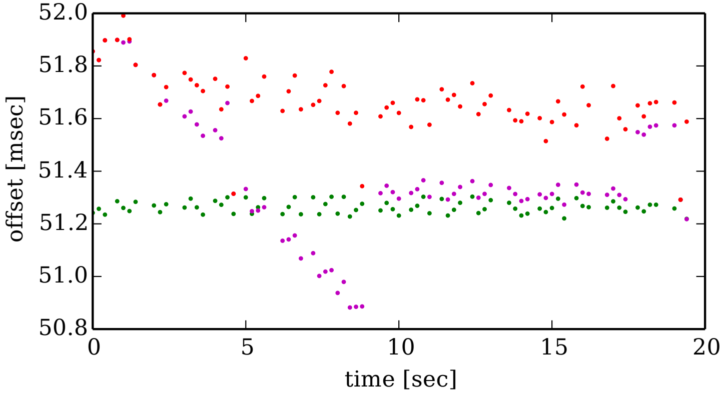


Figure 7.7.: Bumblebee2 time offsets estimated using our method for received (red), and hardware (green) timestamps. The statistics for this 20 sec dataset are tabulated in Table 7.1. The offsets shown in purple are also obtained using DT-stamps. However, they are translated to CT using a filter (Section 3.3) initiated at the beginning of the dataset as opposed to the green offsets, which are translated using a filter which has been running for a while. This highlights the effect of using one way clock translation, to learn the higher order terms of clock skew and offset for the sensor hardware clock with respect to the CPU clock.

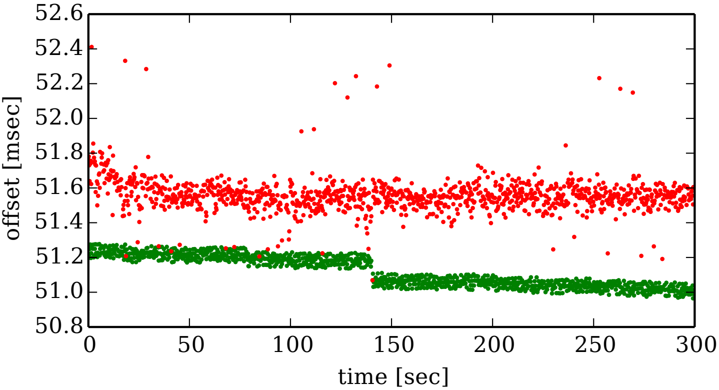


Figure 7.8.: Bumblebee2 time offsets over a longer, 300 s dataset (1124 images, +374 skipped due to LED transitions). The red points show the offsets of the image arrival timestamps. One can also see the slow, drift of the camera hardware clock with respect to the CPU clock over time (green). This is due to imperfection of the one way clock translation applied here (green; convex hull without switching). Our setup is precise enough to highlight these effects, and provides a stepping stone towards addressing them in application scenarios.

Part B

APPLICATIONS

Life-Long Self-Calibration and Fault Detection

Hannes Sommer, Zachary Taylor

Abstract

The EUROPA2 platform contains a large number of sensors and actuators to allow it to perceive and interact with the surrounding world. However, to allow the combination of readings from multiple sources, an accurate calibration of the sensors intrinsics and extrinsic parameters are required. To allow the goal of long-life autonomy these parameters also require continuous automatic calibration over the life of the robot while it goes about normal operations in an unconstrained environment.

In working towards this goal of continuous lifelong calibration, ETH has developed solutions for several of the major issues that limit current calibration approaches. The major contributions were the development of a continuous-time method for obtaining sensor information including orientation, a method for detecting the observability of a system's parameters that is robust to noise, a method that utilizes ICP in combination with the measurement models of point cloud sensors to perform spatiotemporal calibration of point-cloud sensors. All of these methods were finally combined in a batch maximum likelihood estimator that finds the most likely configuration for the system given its sensor readings.

Two different forms of this estimator were created, an offline and an online module. The offline module incorporated additional external pose information and would be used when first calibrating a new system. The online module then operates continuously only using internal readings and would be used to correct for any small errors that accumulate over the life of the EUROPA2 system. Experimental results of both system demonstrate that the system can achieve calibration precision to within a range of around 10 mm, 0.5 degrees and 20 ms for a large range of the given sensors.

Published in:
EU 7th framework programme deliverables, 2016
`https://cordis.europa.eu/`

1. Introduction

Within the EUROPA2 project, one major focus is lifelong operation in urban environments (WP5). In order to be able to achieve this, the capability to continuously self-calibrate the robot sensors and detect failures during operation is required. 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 deliverable D5.1 on the lifelong self-calibration approach for the EUROPA robot.

To accomplish the goal of lifelong self-calibration ETH has developed a new calibration framework. Unlike most sensor calibration approaches, the technique requires no special markers or other calibration aids to be placed in the scene, allowing the process to operate in environments the robot encounters during its normal operation (self-calibration). At the heart of this framework is a batch optimization method that is able to estimate the relevant calibration parameters of a wide range of sensors and actuators. The approach is able to simultaneously consider, and solve for all of the modeled extrinsic and intrinsic parameters of the system, as well as estimating any temporal offsets present in the setup. To accomplish this task, the method utilizes the observations of each sensor in combination with the confidence that can be placed in the measurements, to formulate the calibration as a maximum likelihood problem. This principled estimation of the most likely platform configuration including systematic delays requires that accurate estimates of all sensor's state can be formed at any point in time, not just when readings are observed. To allow this we employ B-spline based continuous-time representation for the entire system state, including 3d-orientation components for which we extended the necessary theory to employ Lie-group valued B-splines in a Jacobian based nonlinear optimization context.

A second major contribution of the presented calibration approach is the exploration of methods for handling unobservable and nearly unobservable parameters. Under a traditional maximum likelihood approach, these parameters would obtain values unrelated to the actual properties of the system, depending almost entirely on the noise present in the utilized measurements or numeric errors. By considering the observability of properties in our approach, it is possible to only update parameters when sensor measurements convey actionable information about the underlying calibration of this component of the system. While the current method we have developed for performing this observability identification cannot prevent all situations that would result in erroneous updates, it acts to robustify the system in many cases.

The final element of research performed for this system was the development of a probabilistic process via which a wide range of depth sensors could be accurately aligned. The approach utilizes an ICP (iterative closest point) method that allows the formation of maximum likelihood error terms from the point matches generated. This allows these sensors to be integrated into the overall ML batch estimation.

From this theoretical framework, a calibration software library has been developed. The library has been programmed in C++, with modularity and extendibility in mind. This means that in many cases an user will be able to use the library to greatly reduce the time required to generate a representation of a generic robotic setup, such as the EUROPA2 platform that contains a large number and range of sensors and actuators. Sensor readings can then be provided to this representation to assist in the calibration of a new and complex systems. This greatly simplifies the daunting task of configuring a new platform. The library is now in its final stages

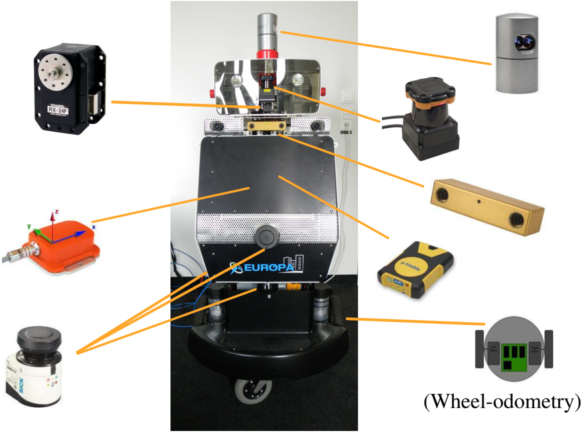


Figure 8.1.: EUROPA2 sensor and actuator setup. The green checks indicate which sensor / actuator is already running and can be calibrated in the EUROPA2 configuration here shown on the ETHZ platform. The question marks indicate sensors or sensor placement, which have not been specified yet.

of development and an open-source version of it will be released shortly. This library has been utilized in the development of offline and online self-calibration modules for the EUROPA2 platform.

Thorough evaluation of both the offline and online calibration procedures has been conducted, with the reliability of their calibrations assessed through measuring the precision of the generated calibration parameters.

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 utilizes a wide variety of sensors, as depicted in Figure 8.1. It comprises a Velodyne HDL-32E, a Bumblebee stereo camera, three Sick 2d-LIDARs, LMS-151, and a Hokuyo, R314, actuated with a Dynamixel RX24F servo. To allow the platform to obtain a clear representation of its state and that of the surrounding environment, the output of multiple sensors will often need to be combined in an accurate manner. To facilitate this multi-modal sensor fusion accurate intrinsic and extrinsic calibration of all the sensors must be performed. However, the large number and verity of sensors the system possess makes this calibration a complex and challenging problem.

2.1. Required calibration parameters

This section gives an overview of the calibration parameters that are present in the EUROPA2 system and how their accurate estimation can be broken down into a series of subproblems.

Many of the sensors used by the EUROPA2 platform come with a pre-set factory calibration for their intrinsic properties. These properties tend to be accurately estimated and do not change significantly over the life of a well maintained system. This means that for these sensors no further intrinsic calibration is required. On the EUROPA2 platform this was found to be the case for the mounted GPS, IMU, Stereo-Camera and LIDARs. Of the sensors used, only the wheel odometry sensor has intrinsics that require calibration. This is because its accurate operation depends on parameters such as the wheel radius which will change due to tire pressure and wear.

In addition to this, the extrinsic pose of all of the sensors as well as their delays must be estimated. To accomplish this we make use of a batch maximum likelihood estimator that operates on the calibration parameters jointly with a time continuous state trajectory. This is combined with highly accurate external position tracks of the platform and sensor specific markers provided by a Vicon system. This procedure allows precise offline calibration while keeping the redundant effort for the online implementation low.

This calibration problem can be broken down into four main challenges:

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

In the following sections each of these areas will be examined in greater detail in the context of both offline and online calibration.

2.2. Sensor delay calibration

One common loss of accuracy is due to the unknown delays that occur between a sensor taking a reading and the data being timestamped. In order for readings from multiple sensors to be accurately combined, precise knowledge of the relative time between these readings is required [113]. This is especially important for accurately estimating the position of points observed while the platform is turning and in SLAM systems where even a small offset can cause significant drift in a systems position estimation.

Sensor readings are often acquired over unreliable channels (for example Ethernet), which introduces unknown and random delays in the order of milliseconds. This renders the simple solution of time-stamping sensor data when it is received by a central computer too inaccurate. This is especially true for sweeping sensors or for sensors, such as IMUs that can perform hundreds or even thousands of readings per second. This fundamental problem has led to the introduction of hardware timestamps committed by the sensors alongside their measurements that are based on an internal clock. This method of time-stamping drastically reduces the problem of random delays. However, the internal clocks of different sensors are typically not synchronized and thus the problem of unknown sensor timing offsets remains. In both cases this problem introduces important calibrations parameters to accurately associate sensor timestamps.

All discrete-time state representations have significant issues with the estimation of these timing related parameters, as associating measurements with a certain time is fundamental to how these discrete methods operate. For that reason we have chosen to employ continuous-time state representations. In these continuous state trajectories, the estimated state given by the sensor measurements is represented by a parametric family of differentiable functions, rather than by a discrete time series of states. This method has been demonstrated to allow accurate estimation of time delays [57]. While these families are well known and often easy to implement for state space components in Euclidean spaces, difficulties can 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 a crucial component if the attitude and extrinsic calibration of the EUROPA2 platform are to be correctly modeled.

2.3. Observability aware calibration

When attempting to estimate the state and calibration parameters of a robot during normal operation, some aspects of 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, [111], is the first published self-calibration algorithm able to cope with this issue.

2.4. Incremental calibration

Incremental calibration is a vital step for reaching our goal of a long lived calibration system. This is because the more time a system spends operating, the more data it will have from which it can calibrate. For a system equipped with a large array of sensors even a few minutes of data can be many gigabytes in size, this means that any long lived system will not be able to store all the raw information relevant for calibration, let alone process it in a reasonable manner. However, a calibration system that only utilities short sections of the newest data would ignore the information gained from all the previous observations of the system. In many cases, parameters which were observable in some previous section of data could be unobservable in the information the platform is currently limited to processing.

Incremental calibration overcomes these issues. In this process each calibration stage takes in the previous calibration as well as the new sensor data when estimating the desired parameters. Estimates of the confidence in each parameter are also calculated and stored to allow an update to have appropriate influence over each parameter. This required knowledge of the confidence that can be placed in a reading and its influence on the system results in a strong coupling between incremental calibration and the previously mentioned observability awareness.

2.5. Sensor data associations

A very different kind of challenge is to associate the sensor readings. This can either take the form of cross-sensor associations or self associations, where the readings from one sensor at different times are compared. Cross-sensor data associations are required 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 associations utilizing nothing but the structure of the scene that the sensors are observing. Due to the large verity of possible objects, configurations and potential sources of error this method of calibration poses a greater challenge then the exploitation of known features for well chosen markers.

The extensiveness of the current literature covering these approaches is largely dependent on the type of sensors being calibrated. The best studied case is that of camera-camera calibration followed by LIDAR-LIDAR. All forms of data association between sensors of differing modalities are more difficult and in general have not been thoroughly explored.

3. Our Approach

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

3.1. Sensor data associations for self-calibration

Our concept concerning sensor data associations for the EUROPA2 self-calibration is as depicted in Figure 8.2. The connecting edges in the Figure represent data associations found by the point-to-plane variant of the Iterative Closest Point (ICP) algorithm (Section 3.4). The tip of the arrow indicates to which side the plane role is assigned to.

For the extrinsic calibration of the IMU and Odometry relative to the 3D sensors (stereo camera and LIDAR) the algorithm builds on the fact that all of them are able to at least partially observe their own 3D motion due to their self associations. Assuming a suitable motion of the platform, this yields the possibility to infer (partially) their mutual extrinsic calibration.

3.2. Estimating sensor delays using Lie-group valued B-splines

Our solution to the problem of time continuous estimation (Section 2.2) for non-vector-space variables, is to generalize the concept of B-splines, so that it may be applied to Lie groups [86]. This method allows finite dimensional parametric representation of $SO(3)$-trajectories

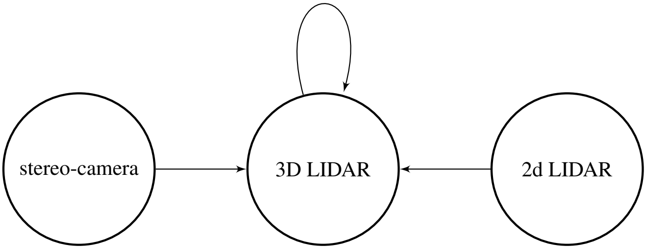


Figure 8.2.: Sensor data associations for the EUROPA platform.

that remain, by construction within the manifold. This representation also ensures that there are no singularities or degenerated points in the parameter space. Furthermore, this method has the added advantage of allowing a lower bound on the degree of differentiability (in time) to be selected for an entire trajectory. This is especially important for physical modelling, where velocities and accelerations need to be available for the state trajectories. We have also developed the necessary analytical expressions to employ these generalized B-splines within nonlinear optimization processes that rely on first order derivatives, such as Gauss-Newton or Levenberg-Marquardt optimization. The accuracy of this approach has been evaluated for the case of attitude estimation. The only comparable work we are aware of employing this for state estimation is [101]. This work makes use of scalar based auto differentiation (part of Google's Ceres) to retrieve the necessary Jacobians. While effective, this approach 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, [58] and [135] uses traditional B-splines to represent a minimal parametrization of $SO(3)$, and is therefore limited by not being rotation invariant.

Further details can be found in the paper "Continuous-Time Estimation of attitude using B-splines on Lie groups" [2], funded by the EUROPA2 project.

3.3. Observability aware and incremental self-calibration

To tackle the problem of unobservable variables that appear observable due to noise in the measurements, we build upon and improve our former work as published in [111]. A journal paper [4], partially funded by the EUROPA2 project, is 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. We achieve this by constraining each Gauss-Newton step to the eigenspaces of the inverse Fisher information

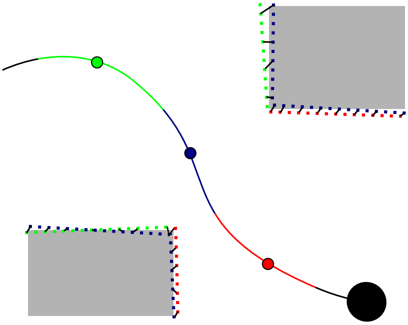


Figure 8.3.: A top view illustration of the grouping (illustrated via coloring) of LIDAR point measurements base on their timestamp along the trajectory of a moving platform (big black dot). The small colored rectangles represent the LIDAR measurements taken from the correspondingly colored segments 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.

matrix, that corresponds to eigenvalues of sufficient magnitude. This is done as these values roughly correspond to how locally observable a given direction is.

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.

For further details behind the algorithm see the attached paper "Online Self-Calibration for Robotic Systems" [4].

3.4. Novel extrinsic batch self-calibration algorithm for LIDARs and stereo-cameras

Our LIDAR-LIDAR, LIDAR-stereo-camera, IMU, and odometry extrinsic calibration concept was specifically developed for the EUROPA2. One of the most challenging problems inherent to spinning LIDAR measurements, especially for multi-beam LIDAR like the Velodyne 32E, is the fact that every point is measured at a different time, this is further complicated by the exceptionally large number of points (up to 700,000 per second for the Velodyne) produced. This results in a moving platform having a different pose for each laser point. An exact ML estimator jointly estimating the platform's trajectory is thus intractable for any practical system, especially

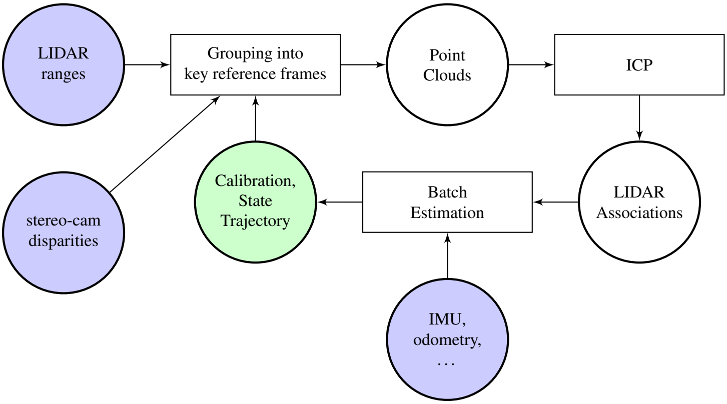


Figure 8.4.: Sketch of the algorithm of iterative refinement of LIDAR-LIDAR and LIDAR-stereo-camera associations to jointly calibrate IMU, odometry, the LIDARs and the stereo-camera

when the goal of online operation is considered. To overcome this limitation we approximate the system in an iterative batch optimization.

The LIDAR measurements are first subsampled over a short duration (about a second each) into consecutive point clouds, as depicted in Figure 8.3, 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 same process is then performed for the stereo imaging data after transforming it into 3d-point clouds using the semi-global stereo-matching algorithm [74]. The current best guess for the platforms state-trajectory is initialized based on IMU and odometry only. The corresponding points between these point clouds are then found by using a point-to-plain ICP approach. These associations are passed 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 8.4.

This process allows us to simultaneously calibrate the for the extrinsic transformations of all the sensors (LIDARs, stereo cameras, IMUs and odometry) as well as the odometry intrinsic calibration (wheel diameters and distance). This system is already functional and working well, with the exception of some failure cases caused by the unforeseen problem of high frequency shaking induced into the system by a passive base joint. See Section 3.4 for an explanation. However, in spite of this issue we believe this method to be suitable and extendible with further sensor data associations up to the full featured online self extrinsic and intrinsic calibration system.

Platform specific challenges and limitations

The EUROPA2 platform contains several properties specific to the system that detrimentally affected the overall accuracy of the calibration. The issues caused by these parameters are especially apparent in the context of online calibration, where no external sensing can be utilized to provide additional information about the platform's state.

The EUROPA2 platform rests upon two large drive wheels and two small caster wheels. The caster wheels are passive, unobservable and reasonably elastic. The lower body is then connected to the upper body through a passive revolute joint that allows the upper body to rotate relative to the lower body. In combination these caster wheels and the revolute joint induce motions into the system that are challenging to predict with any reliability for the following reasons:

- The caster wheels induce nonholonomic, non-linear dynamics into the system as some maneuvers will cause one or both of them to flick directions, inducing a significant shock on the system (can turn it rapidly by several degrees).

- The elasticity of the caster wheels in combination with the revolute joint induces high frequency osculations on the upper body. This occurs whenever the vehicle moves. This shaking is clearly viable to an observer and may displace sensors located near the top of the body by several centimeters.

- The revolute joint will rock the system for several seconds after any acceleration, braking maneuver or driving over slightly uneven ground.

These issues are further compounded by the difficulties attempting to observe them presents. The only sensor located below the revolute joint on the EUROPA2 platform is the odometry sensor and this provides measurements in terms of average rotational and translational velocities at a rate of 10Hz with random delays and lacks timestamp information. There is also a brake that an independent and unmonitored system activates and releases.

In order to overcome these challenges two possibilities were examined. The first was to attempt to model the dynamics of the lower body and its connection to the upper body, however, after careful consideration it was decided that this course should not be pursued for the following reasons:

- The lack of accurate high speed sensing on the lower body would severely limit the observability of any model proposed.

- Any model that could capture the dynamics and interaction of the system with sufficient accuracy to predict its current state and motion would contain a large number of parameters and would add significant complexity to the overall system.

- This model would likely be highly specific to the EUROPA2 platform and be of little use in calibrating other systems.

Instead a second solution was implemented. During offline calibration the top and bottom sections of the platform are calibrated allowing for independent trajectories due to the unknown state of the link between them. External tracking via a Vicon motion capture system is used to ensure that the state of the bottom section remains observable. Then, in online calibration simple constraints are placed on the link between these two sections. This reduces the number of parameters that can be estimated in an online fashion, but ensures the system is observable and can reasonably be expected to converge to a correct calibration in many environments.

3.5. In-laboratory offline calibration

Accurate state estimation is crucial for accurate self-calibration. However, as previously discussed (Section 3.4), the limited sensor information on the systems lower body reduces the observability of its pose. This means that any estimation of the full system state from onboard sensors alone may contain significant uncertainties that will detrimentally impact the accuracy obtainable in the sensor calibration stage.

To overcome this, in the case of in-laboratory offline calibration we employed an external motion capture system (Vicon). This system separately tracks the lower and upper body's motion. This process assists the accurate offline calibration process, even though it introduces further unknowns, namely the pose of the tracking markers on the platform and the alignment of the motion capture reference frame with gravity. These unknown parameters can be estimated jointly with the parameters for the platform itself. Aside from the setup of the motion tracking system's markers no special structure is required for the surrounding environment, though some environments will result in poor observability of some parameters. The most significant issue is that many man made environments have most of their surfaces either parallel or perpendicular to the ground. This results in observability issues in predicting the position of the 2D LIDAR. This was overcome in our laboratory tests by placing several surfaces (boards and cardboard boxes) at a verity of angles around the room. These surfaces also contained sufficient texture to ensure that stereo matching could be performed. Figure 8.5 shows the setup we used for the automatic offline calibration.

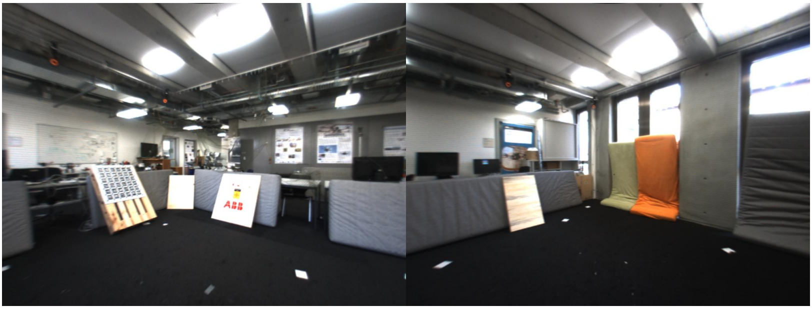


Figure 8.5.: The calibration arena with Vicon motion capture system as seen from the onboard Bumblebee camera

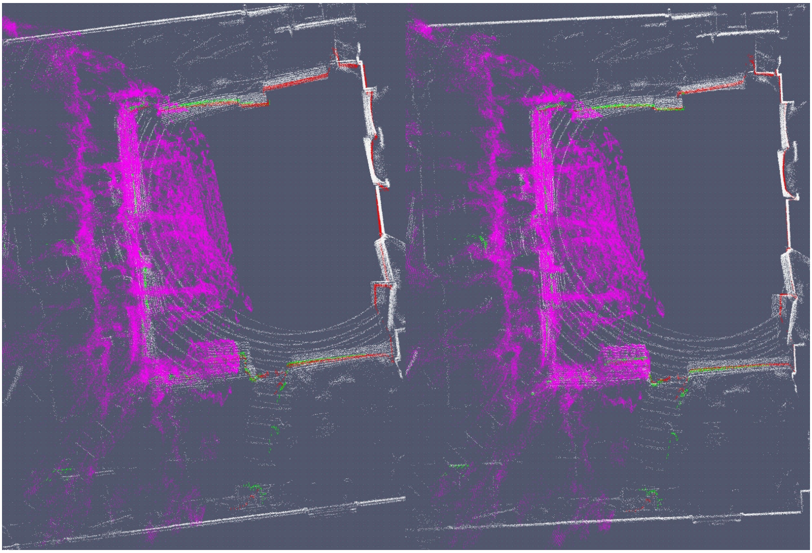


Figure 8.6.: Two top views, before and after low quality self-calibration (25s), showing subsampled point clouds of two horizontal 2d-Sick (green and red), Bumblebee stereo-camera (magenta) and 3D-Velodyne (white) of 1 second of sensor data taken over about 10s of a slow forward movement in a the offline calibration arena. A comparison shows the effect of our self-calibration implementation: in the left image the clouds are less crisp and badly aligned, after being transformed to the image coordinate frame based only on manual extrinsic calibration; the clouds in the right image, after calibration, are crisper and much better aligned. The yellow circles mark some areas where the improvements are particularly easy to spot.

4. Software

While working on the EUROPA2's calibration task ETH put great effort in extending and improving their calibration software.

Based on earlier state-of-the-art research that was continued for the EUROPA2project we released, Kalibr, a camera-to-camera and IMU calibration toolbox as open source. While being well suited for the tasks of offline, target based N-Camera to IMU calibration, Kalibr is lacking any support for online or self-calibration or LIDARs. Furthermore, its design and operating principles did not allow it to be easily extended to incorporate the required extensions and tasks.

Therefore it was decided to start, for the EUROPA2, the development of a much more general, modular calibration framework entirely in c++ that would cover all use cases that were envisioned for a platform like EUROPA2, namely automatic and continuous online and offline self-calibration.

The system currently comprises the following general features:

- Support observability aware and incremental or full batch calibration
- Modular and object oriented design that supports easy assembly of a specific setup and extension with new sensors, joints, trajectories and data association algorithms
- Support for external sensors such as motion capture systems for higher accuracy or model debugging
- Block automatic differentiation (BAD) for rapid differentiable model prototyping
- Continuous time state representation to support delay estimation and simplify modeling
- Inspection and analysis tools to support model debugging and evaluation
- Calibration-server mode in which the calibration algorithms can be triggered and configured from a separate software module while it conducts specific maneuvers with the platform

This work currently contains some coupling with EUROPA2 specific code, however it is planned to remove this and release the code to the public as open source library.

On the EUROPA2 we integrated both the offline and online calibration modules as well as an interactive calibration viewer (Figure 8.7) that allows for changing the calibration parameters while showing a live image where LIDAR points are projected into the Bumblebee images such that a human can easily inspect or manually tune the calibration.

5. Evaluation

The evaluation of the calibration modules with the ETH EUROPA2 platform were performed as follows. The offline calibration was first evaluated by measuring its precision. The online calibration was then conducted and the results compared to those obtained in the offline calibration process. All results are presented as figures showing box-plots for the extrinsic parameters of a given frame pair. We specify each spatiotemporal extrinsic calibration for evaluation with a set of 7 parameters as follows:

- x, y, z for the three translations
- roll, R, pitch, P, and yaw, Y, for the rotation[1],
- d for the delay (the temporal offset)

---

[1] Internally all estimation and analysis is done with unit quaternions; only for human readable plots and numbers we switch to these Euler angles.

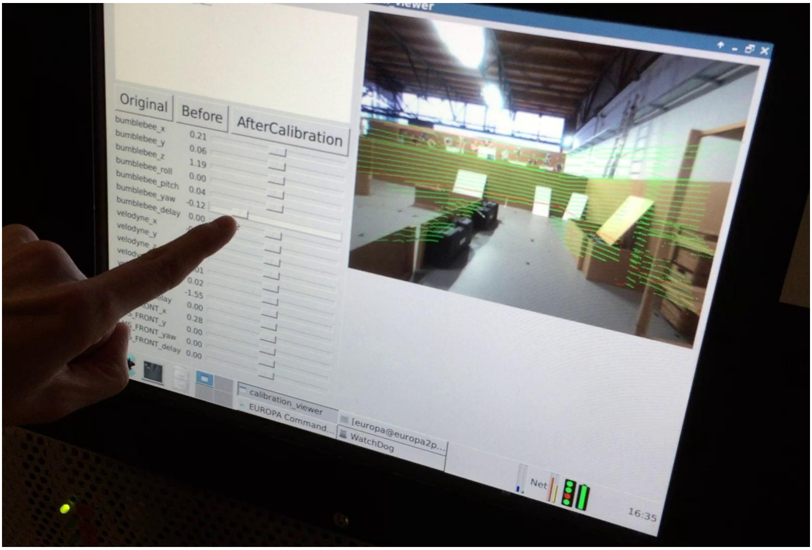


Figure 8.7.: The interactive calibration viewer running on the ALU-FR platform. The sliders on the left allow for changing calibration parameters while the image on the right immediately shows the effects by overlaying live laser data on camera images. The spoiled Bumblebee yaw (by -0.12 rad $\simeq -6.9$ deg) is easy to detect for a human in the image.

We typically present the extrinsic calibration with respect to the Velodyne 32E's frame as it is in some sense the central sensor of the EUROPA2 platform. All estimations are done without using any prior distribution. The hand calibration is only used as initial guess. This was done so that the processes actual estimation precision could be evaluated without being artificially influenced or reduced by a prior.

For a productive online estimation one would provide a prior together with the initial guess up to some point where enough information is accumulated to increase the initial robustness.

5.1. Offline

For offline calibration, we repeatedly conducted procedures for specific calibration sub-goals (typically 10 each) while the platform was located within a motion capture system (Vicon). To achieve this we defined *calibration plans* for specific *calibration goals*. To ensure that the parameters of interest would be observable some flat structures with visual texture were placed in a verity of poses around the room in which calibration was performed.

Each calibration plan consists of a maneuver the platform has to execute and specifications for when each sensor should record data. This helps increasing accuracy as some sensor (LIDARs, cameras) readings contain more sources of uncertainty when made from a moving platform, while other sensors require motion (IMU, Odometry).

The offline calibration module for the EUROPA2 is able to repeatedly execute these calibration plans, provided by humans using a script like specifically tailored c++ mini-language, in a safe way without human intervention. This is possible because it can lean on EUROPA2's local planner with its collision preventing safety module and localization within the motion capture system.

In the following we will report on different calibration plans used and the corresponding results. These results can be merged together to an offline calibration of all pairwise extrinsic calibrations as they are all connected through the Vicon marker poses. Only the odometry is slightly detached because it is estimated w.r.t. the Vicon markers on the lower body while all the other sensors are found w.r.t. the markers on the upper body. However, the default transformation between the two marker sets (upper and lower body), when resting on flat ground orthogonal to gravity direction we extracted directly from Vicon measurements while the robot was resting.

IMU offline calibration plan

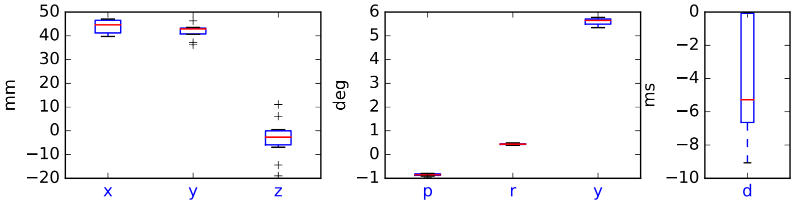
To calibrate the IMU extrinsics w.r.t. to the upper body Vicon markers we conducted a separate calibration routine. It consists of first rotating autonomously on the spot for about 15 seconds. After this has been performed a human operator grabs and shakes the upper body back and forth for 15 seconds. This was the only maneuver that required manual work. It was necessary as exciting the IMU to this extent with driving is not feasible indoors on flat ground. A full autonomous alternative could have been to build a ramp for it to drive on and excite the accelerometers through changes in the gravity direction.

Figure 8.8 shows the result of 10 independent calibrations of the aforementioned 30 second long datasets. Most of the calibration runs for this sensor fall within a range of around 20 mm and 0.5 degrees. We consider this to be a highly accurate result given the length of the dataset and the size and flexibility of the platforms body.

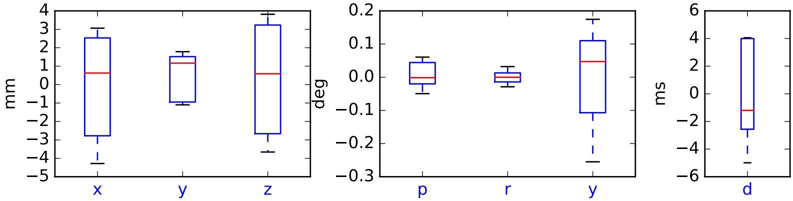
Static and dynamic offline calibration plans

In the *static* calibration plan the robot drives through several positions and headings in a stop and go pattern and only considers sensor readings while the robot is not moving and not shaking significantly. This allows estimating the spatial extrinsics without them being coupled to the temporal extrinsics (delays). We conducted 10 experiments, with each experiment taking roughly 60s. However, for the point cloud (LIDARs, and the stereo camera) sensors only about 11×0.3 seconds of data are used while the robot is resting.

In the *dynamic* calibration plan the robot drives through several positions and records data while slowly moving (forward turns). This allows observing delays between sensors while inducing minimal shaking in the system. Furthermore, it yields more three dimensional point clouds for the 2d LIDARs that are not scanning horizontally. This can deliver better matching with the 3d point clouds from the Velodyne. We conducted 10 experiments of about 60s in total. However, for the point cloud sensors only about 7×1 seconds of data are used while the robot is performing the special slow motion for recording data.



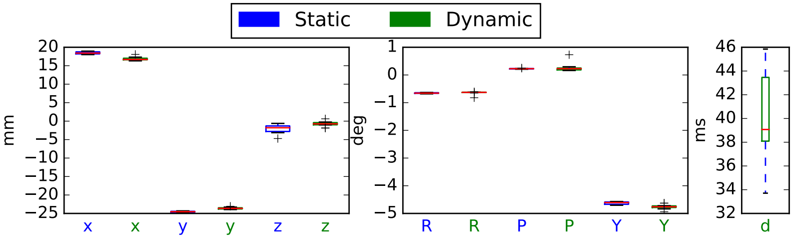
(a) With respect to hand calibration (corresponds to 0)



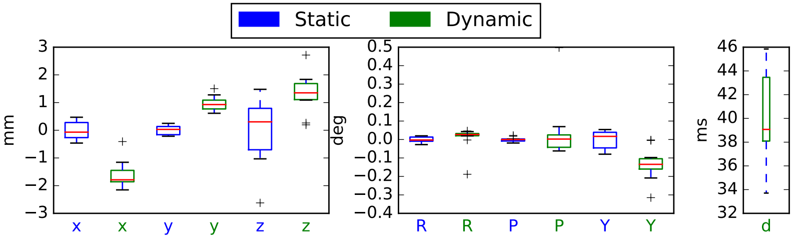
(b) zoomed in and mean centered

Figure 8.8.: The extrinsic calibration IMU to Vicon markers on the upper body based on 10 × 30 seconds IMU offline calibration maneuver.

Figure 8.9 shows the results for extrinsic calibration Velodyne to VICON markers on the upper body and both, static and dynamic experiments. The delay is w.r.t. the Vicon system's clock. Figure 8.10 shows the corresponding results for Velodyne to the Bumblebee stereo camera. The EUROPA2 platform has three 2d-Sick-LIDARs as indicated in Figure 8.1. Two scanning roughly in the horizontal plane - one scanning in the front direction of the platform, one scanning in the back direction and one located on the robot's belly scanning downwards. Their offline calibration results are shown in Figure 8.11 for the rear scanner, in Figure 8.12 for the front scanner and in Figure 8.13 for the down facing scanner. It is quite satisfying that all three Sick-LIDARs are found by the offline calibration to have similar delays (around 30 ms earlier than the Velodyne).

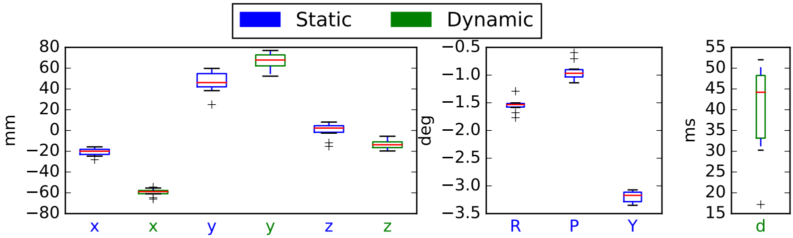


(a) With respect to hand calibration (corresponds to 0)

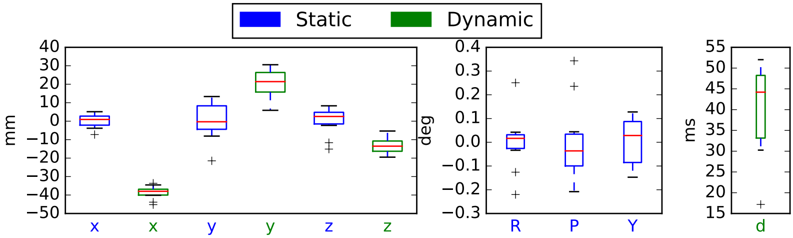


(b) zoomed in and mean centered

Figure 8.9.: The extrinsic calibration Velodyne to Vicon markers on the upper body based on 10×60 seconds static and dynamic offline calibration maneuver with quite satisfying precision.



(a) With respect to hand calibration (corresponds to 0)



(b) zoomed in and mean centered

Figure 8.10.: The extrinsic calibration Velodyne to the Bumblebee stereo camera based on 10×60 seconds static and dynamic offline calibration maneuver. The precision is as expected due to the challenging alignment task. The rotation parameters for the dynamic experiments are fixed here for Velodyne and Bumblebee as otherwise the delay would have lower precision because the experiments do not allow perfect distinction.

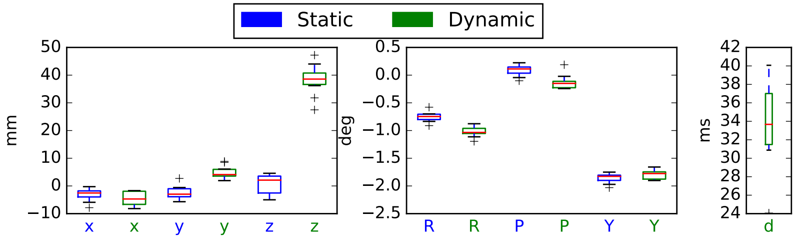


Figure 8.11.: The extrinsic calibration Velodyne to the rear 2d-Sick-LIDAR based on 10 × 60 seconds static and dynamic offline calibration maneuvers. The height of a horizontally scanning 2d-LIDAR is typically not very well constrained especially when moving and therefore slightly shaking. However, to explain this large and consistent deviation would require further investigation.

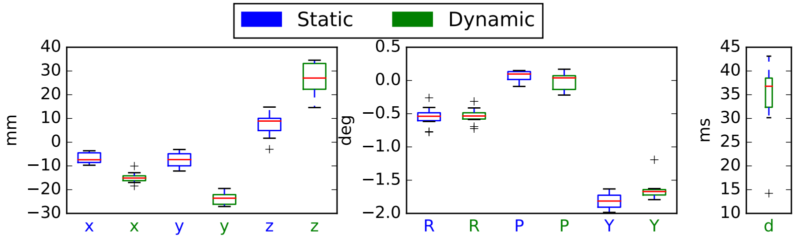


Figure 8.12.: The extrinsic calibration Velodyne to the front 2d-Sick-LIDAR based on 10 × 60 seconds static and dynamic offline calibration maneuvers. The precision is slightly worse than for the rear Sick (Figure 8.11). This is likely due to the smaller field of view of this sensor.

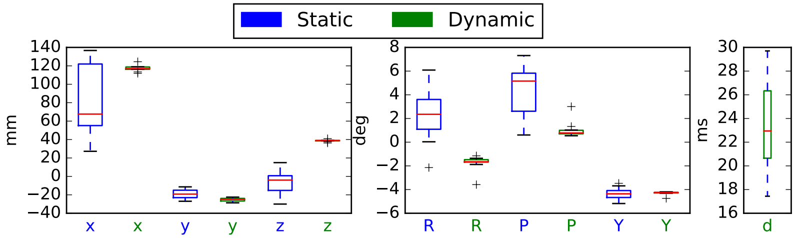


Figure 8.13.: The extrinsic calibration Velodyne to the down facing 2d-Sick-LIDAR based on 10 × 60 seconds static and dynamic offline calibration maneuvers. The precision based on the dynamic experiments is quite surprising especially considering the apparently quite bad hand calibration. The bad performance when based on the static experiments nicely shows how beneficial for this down facing sensors it is to record in motion - at least with our method.

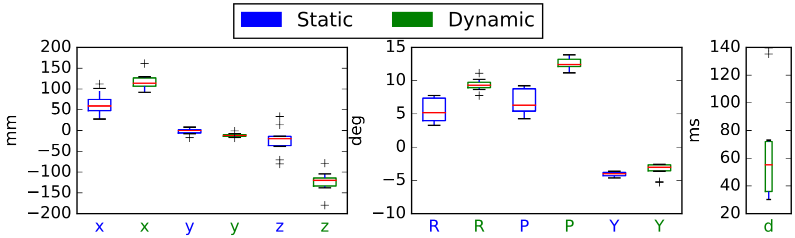


Figure 8.14.: The extrinsic calibration Velodyne to the 2d-Hokuyo-LIDAR based on 10 × 60 seconds static and dynamic offline calibration maneuvers. Like for the down facing Sick (Figure 8.13) the dynamic offline experiment yields more precise results. The large variance of the estimates is most likely due to the strong vibrations the Hokuyo-Nose is suffering from in its current setup. The metal plate on which the Servo is mounted on is flexible and allows for an unmodeled swinging motion - easily visible to a human.

5.2. Online

The main difference of the evaluated online module to the offline calibration above is that no motion capture system is used. Therefore the platform's state is estimated purely by fusing onboard sensors, namely Odometry, IMU and Velodyne. Because the calibration is initialized with the same coarse hand calibration, the state estimates will initially suffer from the inaccurate calibration. Only through joint estimation of state and calibration parameters this can be and is solved in the EUROPA2 online calibration module. Unfortunately, the platforms shaking behaviour makes state estimation very inaccurate and we believe this is the main limitation that reduces the calibration precision.

The estimation algorithm is performing incremental calibration. Here based on mini-batches of about 10 seconds data each. Figure 8.15 shows a former result for the online module in an office environment.

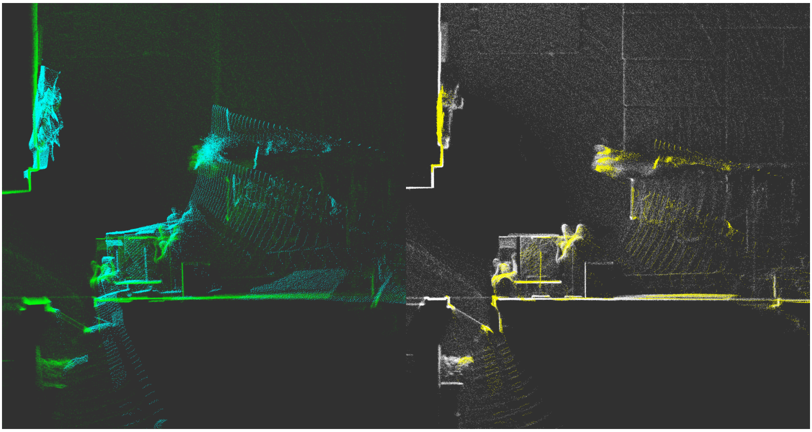


Figure 8.15.: Two top views, before and after LIDAR extrinsic self-calibration, showing sub-sampled point clouds of the almost-vertical front 2d-Sick (cyan and yellow) and 3D-Velodyne (green and white) taken over about 10s of a slow left turning forward movement in a room. A comparison shows the effect of the current state of our self-calibration implementation: in the left image the two clouds are less crisp and badly aligned, 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 aligned.

Figure 8.17 shows the extrinsic calibration Velodyne to the Bumblebee stereo camera.

As previously mentioned, the EUROPA2 platform has three 2d-Sick-LIDARs as indicated in Figure 8.1. Their online calibration results are shown in Figure 8.20 for the rear scanner, in Figure 8.19 for the front scanner and in Figure 8.18 for the down facing scanner.

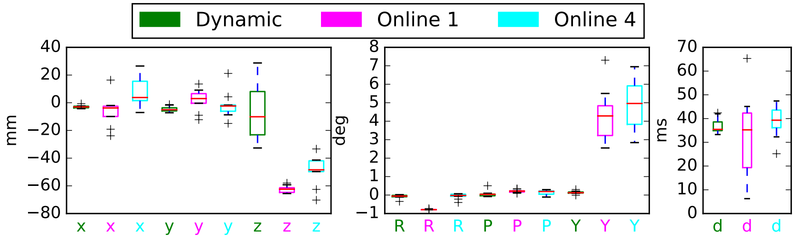


Figure 8.16.: The extrinsic calibration of the Velodyne to IMU, comparing the offline and online approach against the hand calibration. Online 1 refers to the calibration after one accepted mini-batch. Online 4 refers to the calibration after four accepted mini-batches. The y, d nicely show the effect of accumulated information in the incremental calibrator through the reduction in their variance. Parameters like R that seem to have an opposite trend are weekly observable based on the first mini-batch. The big difference in z and yaw are due to the shaking of the upper body. Without a good representation of the kinematics of the joint connecting upper body and base the accelerometer measurements cannot be explained and typically push the IMU much higher than it is.

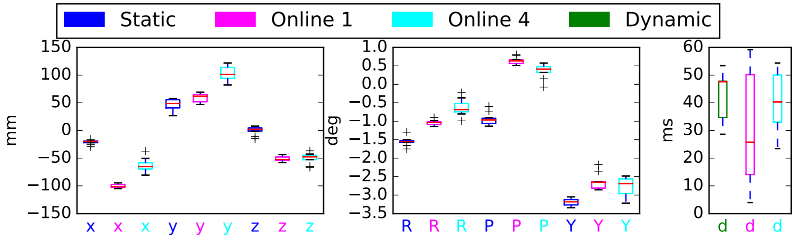


Figure 8.17.: The extrinsic calibration of the Velodyne to the Bumblebee. Offline and online calibration are compared based on 10×60 seconds data from the dynamic offline calibration maneuver. The accuracy for the translational extrinsics shows some significant variation due to the difficulty of estimating these parameters solely from the onboard sensor information. The rotation parameters for the dynamic experiments are manually locked here for Velodyne and Bumblebee to the estimates from the static experiment because otherwise the delay estimate would have lower precision because this experiment cannot identify the relative rotation as good and a bad rotation estimate will degrade the delay estimate for which the dynamic experiment is mostly for.

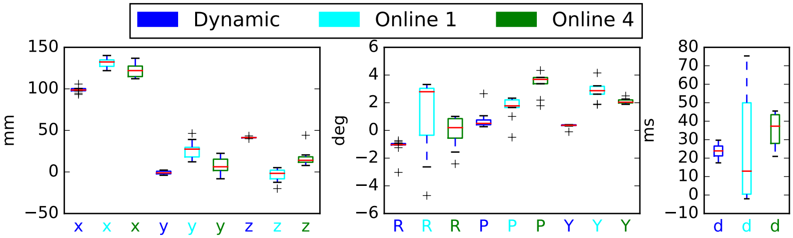


Figure 8.18.: The extrinsic calibration Velodyne to the down facing 2d-Sick-LIDAR. Offline and online calibration compared both based on 10 × 60 seconds data from the dynamic offline calibration maneuver. An okay accuracy given the short amount of data. The variance reduction due to more information is clearly visible.

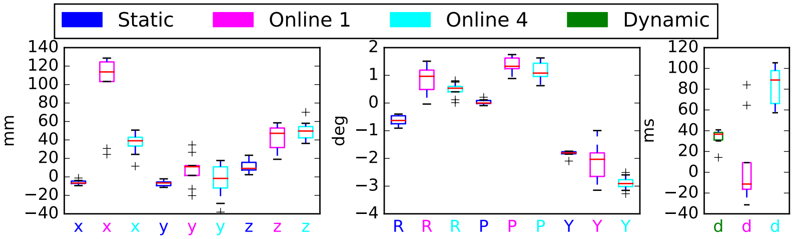


Figure 8.19.: The extrinsic calibration of the Velodyne to the front 2d-Sick-LIDAR. Offline and online calibration are compared based on 10 × 60 seconds data from the dynamic offline calibration maneuver. The horizontal 2d-LIDARs are the sensors that are most detrimentally effected by the shaking upper body. The seeming consistent deviations are likely due to the fact that the maneuvers were executed in a rather similar way causing quite similar shaking and observability.

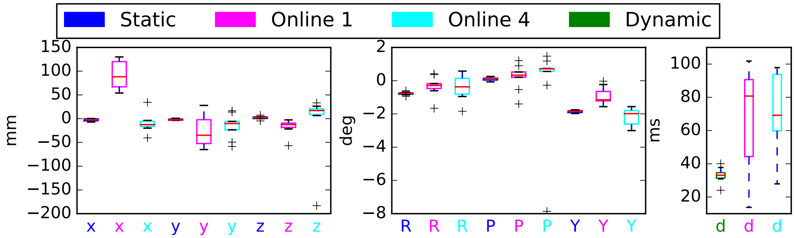


Figure 8.20.: The extrinsic calibration of the Velodyne to the rear 2d-Sick-LIDAR. Offline and online calibration are compared based on 10 × 60 seconds data from the dynamic offline calibration maneuver. The same explanations as for Figure 8.19 apply to the apparent inaccuracies.

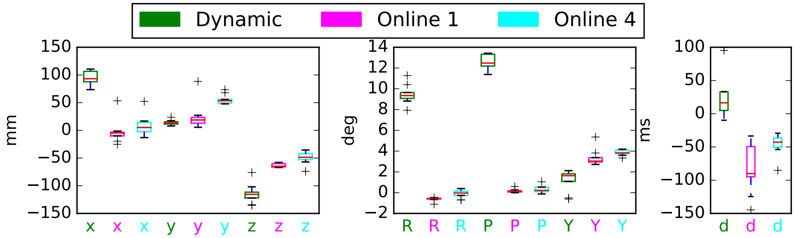


Figure 8.21.: The extrinsic calibration Velodyne to the Hokuyo-Nose. Offline and online calibration compared both based on 10 × 60 seconds data from the dynamic offline calibration maneuver. The deviations between offline and online are due to too conservative observability assessment in the online algorithm. This shows the limitations of the current observability analysis method.

5.3. Odometry

To make the intrinsic odometry parameters, namely the two wheel diameters, R_l and R_r, and the wheels distance L, observable the calibration plan comprised multiple right and left circles to drive and several back and forth straight motions. Here, the rather limited area within the Vicon motion capture system comes as a drawback. Figure 8.22 shows the results of 10 experiments of around 60s in length for odometry calibration, as well as the corresponding results from the online calibration experiment.

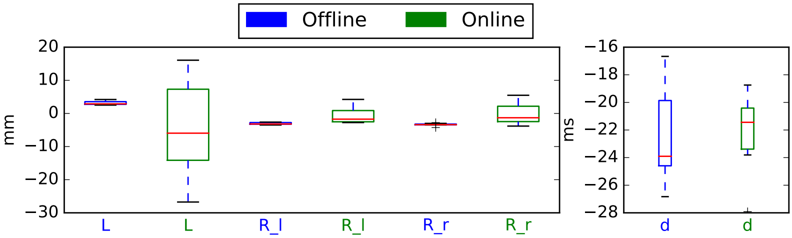


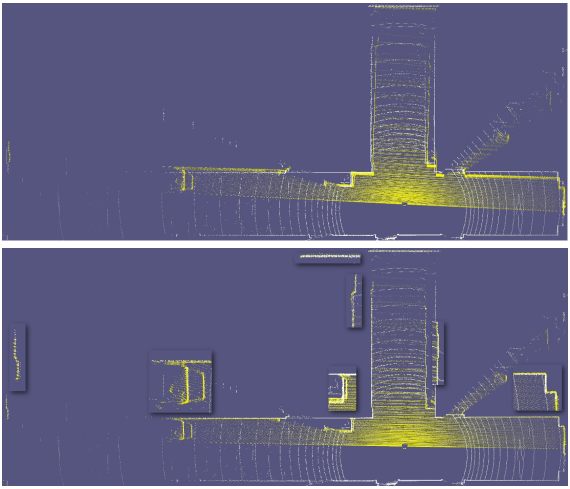
Figure 8.22.: Odometry intrinsic parameters estimated based on an Odometry specific offline maneuver (about 60 seconds each) and with incremental online calibration.

5.4. Update: improvements for the Hokuyo-Velodyne calibration

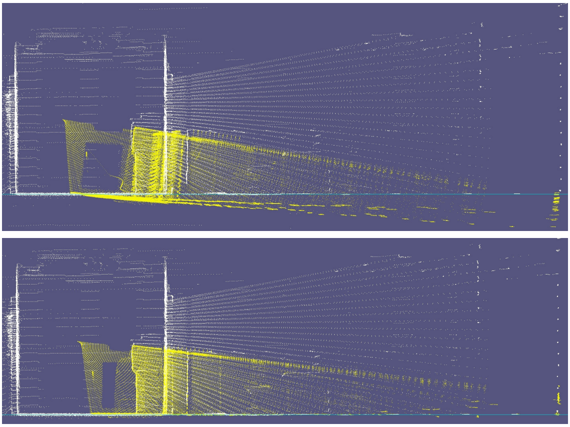
Here, we provide an update that shows evaluations of where we improved the system since the state of the above sections. We investigated and improved on the offline calibration of the tilting Hokuyo, which showed the least satisfying results back then. For that we added a specific calibration plan for the Hokuyo with respect to the Velodyne and the tilting unit: In a suitable environment, i.e. containing larger planar structures in proximity to the robot, such as a wide corridors, the robot is standing still and only tilting the Hokuyo up and down with constant velocity (except around the turning points). This introduces fewer vibrations, while still allowing to observe the delay between the tilt angle encoders and the Hokuyo measurements. Experiments with sinusoidal motions patterns for the tilting unit did not reduce the vibrations visibly and yielded no significant improvements for the calibration stability.

Figure 8.23 shows the typical qualitative results of the self-calibration in a T-shaped corridor environment. The automatic self-calibration routine (bottom) produces crisper and substantially better aligned point clouds than manual calibration (top). Hereby, the self-calibration-algorithm is initialized with the manual calibration routine.

We evaluated the accuracy of the approach by repeating the self-calibration 5 times for each of three different locations in an office building environment. Each dataset is about 12 seconds long. Figure 8.24 shows the statistics over the 15 spatiotemporal extrinsic calibrations of the tilting unit with respect to the Velodyne sensor, which—for this type of stationary experiment—is considered



(a) Manual and self-calibration, top view



(b) Manual and self-calibration, side view

Figure 8.23.: Top and side view in a T-shaped corridor environment before (top) and after (below) self-calibration between Velodyne (white) and the tilting Hokuyo (yellow). The cyan line represents the projection of the ground plane, used as a reference for the absolute Velodyne pose. The side view, (b), also shows the improved time delay estimate: without the correction there are two ground lines visible in the yellow cloud. The severe vibrations in the Hokuyo mounting are easy to observe as deviations from ideal straight ground lines.

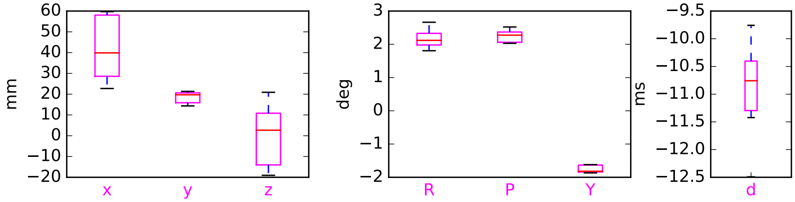


Figure 8.24.: *Left, middle*: The spatial extrinsic calibration of the coordinate frame, in which the Servo is tilting the Hokuyo, with respect to the Velodyne coordinate frame. *Right*: The delay of the angle encoder output with respect to the Hokuyo measurements. A value of zero corresponds to the initial calibration. The sample standard deviation of the delay estimate (about 0.6ms) is quite good considering the precision of the internal Hokuyo timestamping (1ms).

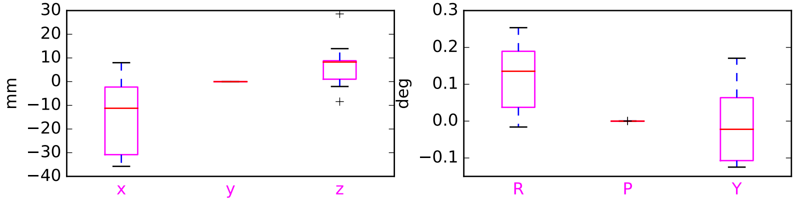


Figure 8.25.: The spatial extrinsic calibration of the tilted Hokuyo sensor with respect to the coordinate frame, which it is tilted in.

to have no delay with respect to the Hokuyo. Figure 8.25 depicts the corresponding results for the spatiotemporal extrinsic calibration of the Hokuyo measurement frame with respect to its tilting unit. Despite the clearly improved accuracy compared to the state at deliverable 5.1, the remaining variance is not fully satisfactory. In fact, the sample variances for each of the three different locations are significantly smaller than for all locations together. This indicates that the incorporated model does not generalize perfectly. We believe that the limiting factor here is the consideration of only extrinsic parameters.

In order to further improve accuracy, we implemented two new features into the system:

i) Intrinsic calibration for the Velodyne 32E: beam-wise distance offsets and vertical angle corrections

ii) Automatic extraction of larger planar patches from the environment

The motivation for i) is that a bad intrinsic calibration of the 32E basically prevents good alignment with the 3d point clouds measured by the tilting Hokuyo. More precisely, this happens because the individual beams of the Velodyne would lead to contradicting extrinsic calibrations between the tilting Hokuyo and the Velodyne. The amount of contradiction corresponds directly to the inconsistency between the individual beams. The reason for ii) is that when aiming at intrinsic calibration for the Velodyne, it is no longer sufficient to rely on very local planar assumptions. The local normal estimates would depend too much on very few individual beams (typically two to three), and only the assumption of larger structures can lead to a consistent correction for all beams. Of course, working with planar hypotheses only requires a suitable environment that provides these planes. Luckily, this mostly holds for indoor environments, and in principle one can reject bad planar patches as long as there are enough good planar patches in an iterative procedure while estimating the intrinsic parameters.

Both features are not fully mature at the current stage, and still fail to improve the repeatability of the latter calibration task (Hokuyo to Velodyne) in a reliable manner. However, they improve the precision of the estimates for the Velodyne height (z) with respect to a gravity aligned ground plane for an upright robot position and improve the generalization over different environments. Figure 8.26 shows the improvement of jointly calibrating the intrinsics on the statistics over five datasets for each of five different locations. With bad intrinsic calibration, the maximum likelihood (ML) height estimate depends much more on how the beams touch the ground, because the individual beams disagree about the distance to the ground. With good intrinsic calibration, the beams tend to agree better on the distance to the ground, and the compromise gets less dependent on which beams had to be considered. The remaining variance is presumably due to the imperfection of the observed ground planes and to the unobservability of the total scale of all beams. Being unobservable, the scale is determined by the initial calibration. But again, the compromise for the scale is dependent on how much the individual beams are involved because they disagree.

Figure 8.27 visualizes the point clouds of the two Sick LIDARs in the horizontal plane in addition to the Velodyne and the Hokuyo after a calibration with intrinsic Velodyne calibration and automatic planar patches extraction, ii). Here it becomes apparent that a more precise alignment of the Velodyne especially with respect to the rear Sick LIDAR (red) is impossible to achieve with the current model, because they disagree strongly on how straight the walls are. Since the Hokuyo and the Velodyne quite nicely agree, we suspect the larger systematic error at the Sick LIDAR. To correct for that, we would need to introduce further intrinsic parameters for the Sick LIDARs.

6. Unresolved issues

While significant work has been performed towards the goal of lifelong calibration, several significant challenges must still be overcome before a fully autonomous life long calibration approach could be reliably implemented in a robotic system such as the EUROPA2 platform.

The current implementation has no high level failure detection implement. While a simple calibration approach could be implemented by detecting severe changes in calibration parameters with short periodic non incremental calibration runs, this method would have a lag of several seconds.

Another shortcoming of the current implementation is that the incremental online calibration

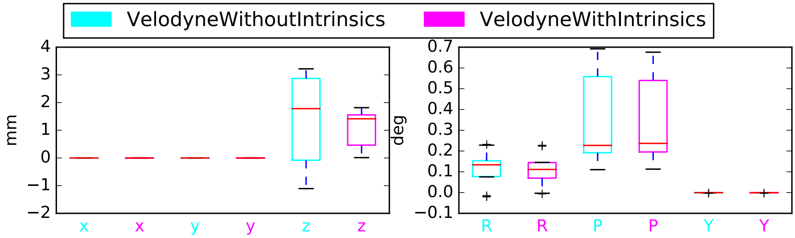


Figure 8.26.: The sample statistics of the extrinsic parameters of the Velodyne with respect to a gravity aligned ground plane on which the platform rests. Activating the current capabilities of intrinsic self-calibration reduces the sample standard deviation to almost a third. The parameters with no visible spread are not observable and properly detected as such. The yaw angle (Y) has very little spread because the statistics is not given in asymmetric Euler-angle parameters, but rather based on the geometric logarithm around the Fréchet-sample-mean and yaw (Y), pitch (P), and roll (R) only refer to the corresponding components.

module does not ever forget any informative data. The implemented information gain criterion drastically limits the rate of accumulated data, however it will slowly increase over time increasing the calibration processing time. There are some simple strategies that could be applied to correct this issue (such as out-dating), however these were not implemented as we attempted to research more sophisticated methods.

The evaluation in this report does not include the calibration of the two actuators present on the platform, a locomotion system and a nose-tilting servo. Both are integrated in the estimators and can be calibrated. However, the inclusion of the locomotion system currently jeopardize the incremental online calibration due to random delays in its operation. These unmodelled delays adversely impact the estimated trajectory. The second actuator also adversely impacts the calibration as the large flexibility of its mounting violates the assumption of a rigid connection. We could overcome this problem only partially for offline calibration with a special calibration plan (motion) for the Hokuyo sensor as described in Section 5.4.

7. Conclusion

The EUROPA2 platform contains a large number of sensors and actuators to allow it to perceive and interact with the surrounding world. However, to allow the combination of readings from multiple sources, an accurate calibration of the sensors intrinsics and extrinsic parameters are required. To allow the goal of long-life autonomy these parameters also require continuous automatic calibration over the life of the robot while it goes about normal operations in an unconstrained environment.

In working towards this goal of continuous lifelong calibration, ETH has developed solutions

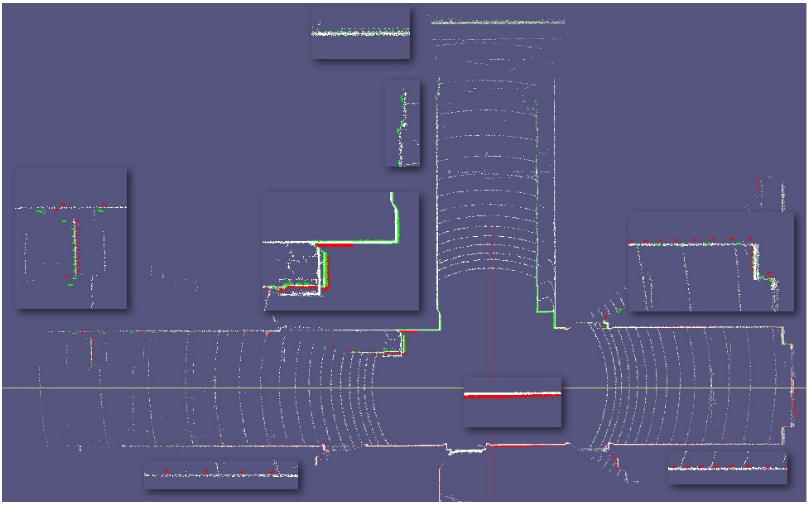


Figure 8.27.: The alignment of the two planar Sick LIDARs (red=rear, green=front) with the Velodyne sensor (white) reveals the limitations of the currently supported model (not including any intrinsics for the Sick LIDARs): The red cloud's lower wall line, for instance, is bent such that the center is significantly lower than the Velodyne's white wall line. At the same time, however, the red line is above the white at the far left and right. The distance from right to left amounts to approximately 15 meters here. This constitutes a problem that cannot be corrected solely through extrinsic calibration.

for several of the major issues that limit current calibration approaches. The first was the development of a continuous-time method for obtaining sensor information including orientation. This is required to allow readings from asynchronous sensors to be compared and allow the accurate estimation of timing delays. The second major contribution was a process for detecting the observability of a system's parameters that is robust to noise. This process assists in preventing many erroneous updates to system parameters. A method was then developed that utilizes ICP in combination with the sensor poses and estimated noise to estimate the joint likelihood of a given pose of a pair of 3D sensors. All of these stages were finally combined in a batch maximum likelihood estimator that finds the most likely configuration for the system given its sensor readings.

Two different forms of this estimator were created, an offline and an online module. The offline module incorporated additional external pose information and would be used when first calibrating a new system. The online module then operates continuously only using internal readings and would be used to correct for any small errors that accumulate over the life of the EUROPA2 system.

Experimental results of both system demonstrate that the system can achieve calibration precision to within a range of around 10 mm, 0.5 degrees and 20 ms for a large range of the given sensors. However, unmodelled high frequency shaking induced into the system by its drive system was found to detrimentally impact some results. This impact often took the form of increased uncertainty or slight biasing.

Overall however, in-spite of this issue the system is able to achieve accurate calibrations in a wide verity of situations. In addition to these results, an open source library that encompasses all of the above functionality is currently in the final stages of development and will soon be provided to the public to simplify the calibration of similar robotic systems.

Part C

APPENDICES

Appendix A

Supplemental work on rotation quaternions

In this chapter we present some of the additinal arguments, proofs, and derivations we developed while working on PaperII and which could only fit into its extended version [151] and are therefore also missing from the version included in this thesis.

A.1. Proof sketch: Correctness of migration recipe

A full technical proof is beyond the scope of this paper and we only provide a sketch of a proof: To show the claim we assume the opposite, i.e. that we have a *counter example*, i.e. a tool, T, that after some migration, M, as in Section 7.2 into $M(T)$ is not doing the same job when used with quaternions of the target convention. Formally, $M(T)$ does not yield the corresponding, i.e. quaternions in the target convention and otherwise equal, outputs when given corresponding inputs (denoted with $T \not\simeq M(T)$).

1. T can be assumed migrated as a whole by M (=: M `doesMigAsAWhole` T):
   If not we take the first sub tool, T', on the way from input to output, for which $T' \not\simeq M(T')$, as new T until M `doesMigAsAWhole` T.

2. T can be additionally assumed having no quaternion in- or outputs and being translated by M ($\neg\mathcal{Q}(T) \wedge M$ `doesTranslate` T):
   To ensure that we inflate T into $\hat{T}$ by adding bijective conversions per quaternion in- or output, $\mathbf{q}$, into a pair $\in \mathbb{R} \times \mathbb{R}^{3\times 3}$ through $\mathbf{C}$[1] (from the source convention). $\hat{T}$ can then be used for exactly the same jobs by employing inverse conversions wherever quaternions were exchanged with T. We migrate $\hat{T}$ by $\hat{M}$ into $\hat{M}(\hat{T})$ by migrating the contained T using M and translating the inflation layer (effectively translating all source $\mathbf{C}$ into the target $\mathbf{C}$; see Example 7.2). Since $\hat{M}(\hat{T})$ would be used through conversions using the target $\mathbf{C}$, which would cancel out the translated inflation, causing the same effective input

---

[1] E.g. $(\operatorname{sign}(q_r)\|\mathbf{q}\|, \mathbf{C}(\mathbf{q}\|\mathbf{q}\|^{-1}))$ if $\mathbf{q} \neq 0$, otherwise $(0, \mathbf{0})$

output behaviour as $M(T)$ and therefore different from $\hat{T}$. It follows $\hat{T} \not\simeq \hat{M}(\hat{T})$, while $\neg\mathscr{Q}(\hat{T})$. Furthermore T must have been already translated: If it had been interfaced (as a whole) $\hat{M}(\hat{T})$ would exactly behave as $\hat{T}$, because the interfacing conjugation effectively revokes the translation of the inflation, rendering $\hat{M}$ into an equivalence transformation, contradicting $\hat{T} \not\simeq \hat{M}(\hat{T})$. It follows $\hat{M}$ `doesTranslate` $\hat{T}$ and, since component-wise translating is equivalent to translating as a whole also $\hat{M}$ `doesMigAsAWhole` $\hat{T}$.

3. T is no counter example:
   It is well-known and straight forward to verify that conjugation is an isomorphism $(\mathbb{H}, +, \odot, \bar{\cdot}, \langle\cdot,\cdot\rangle) \simeq (\mathbb{H}, +, \otimes, \bar{\cdot}, \langle\cdot,\cdot\rangle)$ that lets the embedding $\mathbb{R} \subset \mathbb{H}$ invariant. Since T can only use the quaternions by means of their structure and their relation to $\mathbb{R}$ (nothing more is defined about them after all), replacing all quaternion constants according to the isomorphism and all operations with their isomorphic partners, as precisely done by the translation procedure, cannot change its input-output behavior for non-quaternions. Therefore $\neg\mathscr{Q}(T) \wedge M$ `doesTranslate` T contradicts $T \not\simeq M(T)$ and proves the claim.

A.2. Proof: Hamilton QM-convention always yields more similarity

Let C be a consistent partial convention that is a homomorphic QM-convention and additionally determines a mapping, $C.\mathbf{q} : \mathbb{R}^3 \to \mathscr{U}$ from rotation vectors to the unit quaternions.

A.2.1. Rotation vector to matrix and rotation quaternion conversions

As C is consistent it holds for all $\boldsymbol{\phi} \in \mathbb{R}^3$ that $C.\mathbf{C}(C.\mathbf{q}(\boldsymbol{\phi})) = C.\mathbf{C}(\boldsymbol{\phi})$, with $C.\mathbf{C}$ denoting the rotation vector to matrix conversion determined by C. Assuming the right side has the form

$$C.\mathbf{C}(\boldsymbol{\phi}) = \exp(\mathbf{a}(\boldsymbol{\phi})^{\times}), \tag{A.1}$$

with some $\mathbf{a} : \mathbb{R}^3 \to \mathbb{R}^3$ [2] it holds necessarily

$$\begin{aligned} \mathbf{C}_H(\exp(\mathscr{I}^*(\mathbf{a}(\boldsymbol{\phi})/2))) = \exp(\mathbf{a}(\boldsymbol{\phi})^{\times}) &= C.\mathbf{C}(\boldsymbol{\phi}) \\ &= C.\mathbf{C}(C.\mathbf{q}(\boldsymbol{\phi})) \end{aligned} \tag{A.2}$$

using the exponential map of quaternions, exp [3], because for all $\mathbf{x} \in \mathbb{R}^3$ it holds, independently of C, that

$$\mathbf{C}_H(\exp(\mathscr{I}^*(\mathbf{x}/2))) = \exp(\mathbf{x}^{\times}).$$

Due to the definition of $\mathbf{C}_H$ it follows, for the case that $(\mathbf{C}_H, \odot)$ is sub convention of C, (and therefor $C.\mathbf{C}|_{\mathscr{U}} = \mathbf{C}_H$):

$$C.\mathbf{q}(\boldsymbol{\phi}) = \alpha \exp(\mathscr{I}^*(\mathbf{a}(\boldsymbol{\phi})/2)), \tag{A.3}$$

---

[2] We are not aware of any other existing convention. Also, otherwise it should be hard to justify $\boldsymbol{\phi}$ as rotation vector unless the rotation matrix does not act through the matrix product.

[3] The exponential map is the same for both, Hamilton's and Shuster's multiplication.

with $\alpha \in \{-1, 1\}$. To get the analogous result for the second case, $(\mathbf{C}_S, \otimes)$ is sub convention of C, we can translate (see Section 7.2) the left hand side of (A.2) into the other QM-convention and get

$$\mathbf{C}_S(\exp(-\mathscr{I}^*(\mathbf{a}(\boldsymbol{\phi})/2))) = C.\mathbf{C}(C.\mathbf{q}(\boldsymbol{\phi})), \tag{A.4}$$

which yields analogously for this case

$$C.\mathbf{q}(\boldsymbol{\phi}) = \alpha \exp(-\mathscr{I}^*(\mathbf{a}(\boldsymbol{\phi})/2)). \tag{A.5}$$

Comparing (A.3) and (A.5) with (A.1) it becomes evident that the second case, $(\mathbf{C}_S, \otimes)$ being part of C, always yields an extra $-$.

A.2.2. Angular velocity kinematic equation

All common conventions and frame assignments regarding angular velocities yield one of the two forms of kinematic equation — to the best of our knowledge:

$$\boldsymbol{\omega}^\times = \beta \mathbf{C}^{-1}\dot{\mathbf{C}}, \text{ or } \boldsymbol{\omega}^\times = \beta \dot{\mathbf{C}}\mathbf{C}^{-1}, \tag{A.6}$$

with $\beta \in \{-1, 1\}$ and $\mathbf{C}(t)$ a rotation matrix trajectory.

For the following we assume the first form. The claim can be shown for the second analogously.

Firs we observe that a QM-convention, $\mathbf{q} \mapsto \mathbf{C}(\mathbf{q})$, completely determines the corresponding relation to the quaternion trajectory, $\mathbf{q}(t)$ representing the same trajectory as $\mathbf{C}$:

$$\boldsymbol{\omega}^\times = \beta \mathbf{C}^{-1}\dot{\mathbf{C}} = \beta \mathbf{C}^{-1}(\mathbf{q})\mathbf{C}\dot{(}\mathbf{q}) = \gamma \frac{1}{2} \beta \mathbf{q}^{-1}\dot{\mathbf{q}} \tag{A.7}$$

with a convention-factor $\gamma := 1$ for $(\mathbf{C}_H, \odot)$ and $\gamma := -1$ for $(\mathbf{C}_S, \otimes)$.

The last equality of (A.7) is not trivial but can be extracted from known identities as follows. Table 5.2 contains a well-known special case of (A.7) for $\beta = -1$ and $\boldsymbol{\omega} = \boldsymbol{\omega}_A$:

$$-\mathbf{C}^{-1}(\mathbf{q})\mathbf{C}\dot{(}\mathbf{q}) = \boldsymbol{\omega}_A^\times = -\frac{1}{2}\gamma \mathbf{q}^{-1}\dot{\mathbf{q}}$$

Multiplication with $-\beta$ proofs the last equality of (A.7). And comparing (A.7) with the assumed $\boldsymbol{\omega}^\times = \beta \mathbf{C}^{-1}\dot{\mathbf{C}}$, shows that $\gamma = 1$ yields more similarity independently of β and therefore any conventional decision beyond the QM-convention.

A.3. Representing 3d proper rotations with matrices

In this section we aim at clarifying the different kinds of "rotation matrices" and the distinction about active and passive rotations as far as needed for the arguments presented in the rest of the paper. For that we first we briefly define the setup and some notions we need in order to define rotation matrices. Second we define rotation matrices in three different ways. After that we explain their relation to the usages, active and passive, and how those are related to the composition of rotations.

A.3.1. Rotations in 3d-Euclidean space

First we need a general setup in which to define rotations in three dimensions. The Euclidean three dimensional vector space E_3 with the scalar-product $\langle\cdot,\cdot\rangle$ is sufficient. A linear map $\Phi : \mathrm{E}_3 \to \mathrm{E}_3$ is a proper[4] rotation iff it preserves the scalar product and the orientation. It is important to note, that Φ is not a matrix, but just a map mapping abstract vectors to abstract vectors. A basis, $\mathscr{B} = (\boldsymbol{b}_i)_{i=1}^3$, we call a *positively oriented orthonormal basis (PONB)* iff it is of positive orientation and orthonormal, i.e.

$$\langle \boldsymbol{b}_i, \boldsymbol{b}_j \rangle = \delta_{ij}$$

And we write $[\boldsymbol{v}]^{\mathscr{B}} \in \mathbb{R}^3$ to denote the coordinates of a vector $\boldsymbol{v} \in \mathrm{E}_3$ with respect to $\mathscr{B}$, such that $\boldsymbol{v} = \sum_{i=1}^3 [\boldsymbol{v}]_i^{\mathscr{B}} \boldsymbol{b}_i$ or equivalently, because $\mathscr{B}$ is orthonormal,

$$[\boldsymbol{v}]_i^{\mathscr{B}} = \langle \boldsymbol{v}, \boldsymbol{b}_i \rangle$$

A.3.2. Three competing major approaches to represent a rotation with a matrix

Given a PONB, $\mathscr{B} = (\boldsymbol{b}_i)_{i=1}^3$ there are three major approaches of representing a rotation Φ with a matrix:

(a) The *representing matrix*[5] of the rotation with respect to a PONB $\mathscr{B}$, such that for all vectors $\boldsymbol{v} \in \mathrm{E}_3$:

$$[\Phi(\boldsymbol{v})]^{\mathscr{B}} = \mathbf{R}_{\Phi}^{\mathscr{B}} [\boldsymbol{v}]^{\mathscr{B}} \tag{A.8}$$

This is typically used in mathematical literature. But one could have flipped $\mathbf{R}_{\Phi}^{\mathscr{B}}$ and $[\boldsymbol{v}]^{\mathscr{B}}$ interpreting the latter as row-vector [6] and define this way the transposed matrix.

(b) The *change-of-basis matrix* that transforms the coordinates of all fixed vectors $\boldsymbol{v} \in \mathrm{E}_3$ when the PONB $\mathscr{B}$ is rotated, with respect to which the coordinates of $\boldsymbol{v}$ are given before and after the rotation:

$$[\boldsymbol{v}]^{\Phi(\mathscr{B})} = \mathbf{B}_{\Phi}^{\mathscr{B}} [\boldsymbol{v}]^{\mathscr{B}}$$

One could have exchanged $[\boldsymbol{v}]^{\Phi(\mathscr{B})}$ and $[\boldsymbol{v}]^{\mathscr{B}}$, ending up with the inverse matrix ${\mathbf{B}_{\Phi}^{\mathscr{B}}}^{-1}$, as used e.g. in [121, p. 25].

(c) The *direction cosine matrix* that contains the coordinates of a rotated PONB $\mathscr{B}$ with respect to itself before rotation - one rotated basis vector per row:

$$(\mathbf{C}_{\Phi}^{\mathscr{B}})_{i\cdot} = [\Phi(\boldsymbol{b}_i)]^{\mathscr{B}} \text{ or } {\mathbf{C}_{\Phi}^{\mathscr{B}}}_{ij} := \langle \Phi(\boldsymbol{b}_i), \boldsymbol{b}_j \rangle$$

---

[4] We will often skip the "proper" for brevity. We are not interested here in improper rotations.

[5] Yes, this name seems a bit unfair - compared to the others, since we are talking about how to represent a rotation with a matrix. But the other common name for that matrix connected to a general linear map is "transformation matrix". Unfortunately this name is often specialized to the matrix representing elements of the special euclidean group — at least in the robotics community.

[6] The usual identification of $\mathbb{R}^3$ with $\mathbb{R}^{3\times 1}$ is purely conventional.

Here $\mathbf{M}_{i\cdot}$ denotes the ith row of a matrix $\mathbf{M}$.
This is used e.g. in [77], p. 8 and [146], p. 447.
One could have chosen columns instead of rows. This would have yielded the transposed matrix ${\mathbf{C}_{\Phi}^{\mathscr{B}}}^{T}$.

In all three cases the resulting matrix is itself orthogonal (so its transposed is its inverse) and it has determinant 1 (this corresponds to Φ preserving the orientation). They all depend on the choice of the basis $\mathscr{B}$ but not one to one. In general given a second PONB $\mathscr{C}$ which results from applying the rotation Ψ on $\mathscr{B}$ the representing matrices are all connected by the following conjugation, while $\mathbf{M}$ denotes one of $\mathbf{R}, \mathbf{B}, \mathbf{C}$:

$$\mathbf{M}_{\Phi}^{\mathscr{C}} = \mathbf{B}_{\Psi}^{\mathscr{B}} \mathbf{M}_{\Phi}^{\mathscr{B}} {\mathbf{B}_{\Psi}^{\mathscr{B}}}^{-1} \tag{A.9}$$

But they are not all the same in value (in definition they are clearly different). In fact, the following always holds:

$${\mathbf{R}_{\Phi}^{\mathscr{B}}}^{-1} = \mathbf{B}_{\Phi}^{\mathscr{B}} = \mathbf{C}_{\Phi}^{\mathscr{B}} \tag{A.10}$$

Hence, considering only the values of $\mathbf{R}, \mathbf{B}, \mathbf{C}$ leads to two different rotation matrices (for a given basis and a given rotation) always being a mutually inverse pair.

A.3.3. Usage and Composition

The value of $\mathbf{R}$ is called the *active* rotation matrix and the value of $\mathbf{B}, \mathbf{C}$ the *passive* rotation matrix for the rotation Φ with respect to $\mathscr{B}$. The motivation behind active is the fact that the definition of $\mathbf{R}$ is based on how Φ rotates $\mathscr{B}$ while the idea behind passive is that $\mathbf{B}$ is based on how the coordinates of a fixed $\mathbf{v}$ change when switching basis. We call these different ways to use a rotation matrix its *usage*. However, as we mentioned before, all the definitions include an arbitrary binary choice that taken differently would lead to the transposed / inverse matrix. Therefore the assignment of active and passive to the values of rotation matrices seems arbitrary or purely conventional, too. Especially it is unclear why the definitions are not aligned such that at least they all lead to the same matrix. The reason is that for various applications the important difference between the available matrices is not their definition but what the standard matrix product corresponds to in terms of rotations. Because for every pair of invertible matrices, $\mathbf{X}, \mathbf{Y}$ it holds $(\mathbf{XY})^{-1} = \mathbf{Y}^{-1}\mathbf{X}^{-1}$ it actually makes a difference whether one uses the orthogonal matrix $\mathbf{R}$ or its inverse, e.g. $\mathbf{C}$. And switching between the two options has the same effect on the result as flipping the two represented rotations or the two matrices. Both resulting matrices correspond to a composition of two rotations, Φ, Ψ with a significant difference. Using $\mathbf{R}$ as active and $\mathbf{C}$ as the passive matrix:

$$\begin{aligned}
\mathbf{R}_{\Phi\circ\Psi}^{\mathscr{B}} &\underset{(A.8)}{=} \mathbf{R}_{\Phi}^{\mathscr{B}} \mathbf{R}_{\Psi}^{\mathscr{B}} && \text{(A.11)} \\
&= (\mathbf{R}_{\Psi}^{\mathscr{B}} {\mathbf{R}_{\Psi}^{\mathscr{B}}}^{-1}) \mathbf{R}_{\Phi}^{\mathscr{B}} \mathbf{R}_{\Psi}^{\mathscr{B}} \\
&= \mathbf{R}_{\Psi}^{\mathscr{B}} ({\mathbf{R}_{\Psi}^{\mathscr{B}}}^{-1} \mathbf{R}_{\Phi}^{\mathscr{B}} \mathbf{R}_{\Psi}^{\mathscr{B}})
\end{aligned}$$

$$\begin{aligned} & \underset{(A.10)}{=} && \mathbf{R}_{\Psi}^{\mathscr{B}}\left(\mathbf{B}_{\Psi}^{\mathscr{B}}\mathbf{R}_{\Phi}^{\mathscr{B}}\mathbf{B}_{\Psi}^{\mathscr{B}^{-1}}\right) \\ & \underset{(A.9)}{=} && \mathbf{R}_{\Psi}^{\mathscr{B}}\mathbf{R}_{\Phi}^{\Psi(\mathscr{B})} && \text{(A.12)} \\ \mathbf{C}_{\Phi\circ\Psi}^{\mathscr{B}} & \underset{(A.11),(A.10)}{=} && \mathbf{C}_{\Psi}^{\mathscr{B}}\mathbf{C}_{\Phi}^{\mathscr{B}} \\ & \underset{(A.12),(A.10)}{=} && \mathbf{C}_{\Phi}^{\Psi(\mathscr{B})}\mathbf{C}_{\Psi}^{\mathscr{B}} && \text{(A.13)} \end{aligned}$$

The active matrix, **R**, matches the composition order, Φ after Ψ, with its multiplication order when one keeps the basis of representation (A.11). The inverse, passive matrix, **C** does the same iff one represents the second rotation with respect to the second (rotated first) basis (A.13). In other words the passive is better suited when from the applications domain it is preferable to switch the reference basis with each rotation (e.g. attitude estimation and control) and the active is preferable whenever it is preferable to keep the reference frame fixed for all the composed rotations. In contrast to the definitions above this difference is not arbitrary or conventional. And therefore the labels active and passive, i.e. the usage of a rotation matrix, would be more robustly defined using these properties with respect to composition of rotations.

The big confusion concerning rotation matrices is caused — in our opinion — by the fact that papers and textbooks from different disciplines seemingly fight for the authority to define which definition deserves the label “rotation matrix” instead of mentioning the two fundamental options and explicitly choosing the one that is more suitable for a given use case. Of course somebody who only knows about one definition and usage of rotation matrices will be confused when reading texts that call the inverted matrix “the rotation matrix”.

It is one plausible but hard to verify theory that this habit of ignorance ultimately led to the mistake to use active rotation quaternions in attitude estimation and control (see Section 6.4) — a domain where passive rotation matrices were and still are predominant used. This mistake inevitably led to the problem represented by (5.1), because — as we have seen — the essential difference between the two usages is precisely how they relate to the compositions of rotations.

Appendix B

Further unpublished proofs

B.1. Correctness proof for the hierarchical task optimization algorithm

The correctness and unique optimality proof for the hierarchical optimization algorithm I was contributing to [1] was removed from the final version and was instead published as appendix, A.1, of the PhD thesis [78]. It follows a different and unpublished proof that is less compact but better for understanding the underlying reasons. The original publisher of the approach (in a less generic form) [149] did provide a compact proof that their solution honors the hierarchy and that all feasible tasks are satisfied but neither that all infeasible task are at the same time in fact achieved in a least squares optimal sense nor that this optimum is unique.

B.1.1. On optimization of prioritized objectives in totally preordered sets

We start with more general notions and observations regarding the optimization of prioritized objectives in totally preordered sets.

Let
X be a set
$n \in \mathbb{N}^+$
$(Y_i, \lesssim_i)$, $1 \leq i \leq n$ a family of totally preordered sets
$f_i : X \to Y_i$, $1 \leq i \leq n$ a family of objective functions on X.

We now want to formalize the task to minimize first f_1 then f_2, without changing f_1's value and so on. To achieve this we first introduce some further symbols :

Definition 1 (Notation)**.**
$\tilde{Y} := \bigtimes_{i=1}^{n} Y_i$ to be the Cartesian product of the Y_i
$\lesssim :=$ the lexicographic total preorder on $\tilde{Y}$ induced by the $\lesssim_i$

$$\begin{array}{rccc} \tilde{f}: & X & \to & \tilde{Y} \\ & x & \mapsto & (f_i(x))_{i=1}^{n} \end{array}$$

To finally define our task to be a prioritized minimization problem of the family f_i in the following sense:

Definition 2 (prioritized minimization problem)**.** The task of finding a minimum in $\mathbf{img}(\tilde{f}) \subseteq \tilde{Y}$ with respect to $\lesssim$ is called a *prioritized minimization problem* of the family f_i.

Definition 3 (value uniquely solvable minimization problem in a preordered set)**.** A minimization problem $f \in X \to (Y, \lesssim)$, with $(Y, \lesssim)$ a preordered set, is called *value uniquely solvable* iff $\mathbf{img}(f)$ has an unique minimum.

Remark. *If $\lesssim$ is also an order (i.e. antisymmetric $:\Leftrightarrow (a \lesssim b \land b \lesssim a \Rightarrow a = b)$) then the uniqueness is implied by the antisymmetry.*

Definition 4 (the i first f-components)**. Let** $0 \leq i \leq n$.

$$\boxed{\begin{array}{rccc} f^{\leq i}: & X & \to & Y^{i} \\ & x & \mapsto & (f_j(x))_{j=1}^{i} \end{array}}$$

Definition 5 (componentwise value uniquely solvable)**.** This minimization problem is called *componentwise value uniquely solvable* iff for every $1 \leq i \leq n$ the minimization problem of f_i constrained by an arbitrary fix value of $f^{\leq i-1}$ is value uniquely solvable.

$$\boxed{\forall_{1\leq i\leq n} \forall_{y\in \mathbf{img}(f^{\leq i-1})} \| \min_{f^{\leq i-1}(x)=y} f_i(x) \| = 1}$$

Remark. *For $i = 1$ this only requires $\mathbf{img}(f_1)$ to have an unique minimum, as $f^{\leq 0}$ is the constant empty tuple and thus the constraint is effectively no constraint.*

Theorem 3 (value uniqueness and solution of a prioritized optimization problem, which is componentwise value uniquely solvable)**.**

The prioritized minimization problem of f_i (i.e. the lexicographic minimization problem of $\tilde{f}$) is value uniquely solvable as a whole if it is componentwise value uniquely solvable and the unique value can be acquired as $\tilde{f}(x_n)$ by the following iteration :
Start with an arbitrary $x_1 \in \underset{x\in X}{\operatorname{argmin}} f_1(x)$ and set x_i in the i-th step ($i > 1$) to an arbitrary element of $\underset{f^{\leq i-1}(x)=f^{\leq i-1}(x_{i-1})}{\operatorname{argmin}} f_i(x)$.

Remark. *The choice of x_i essentially assures that for every $1 \leq j \leq i : f_j(x_i) = f_j(x_j)$, which means that the values for the higher prioritized tasks are not touch when choosing the next ones.*

Proof. Let the prioritized minimization problem f_i be componentwise value uniquely solvable. The iteration can proceed up to the n-th step because the problem is solvable by component (i.e. there will always be a x_i to choose).
Assume now the opposite of the claim, i.e. that the $\tilde{f}(x_n)$ is not an unique $\lesssim$-minimum of $\mathbf{img}(\tilde{f})$.

Let $\hat{x} \in X$ be mapped by $\tilde{f}$ to a different value as x_n but also be at least as good as x_n with respect to the minimization problem,

$$\tilde{f}(\hat{x}) \neq \tilde{f}(x_n) \wedge \tilde{f}(\hat{x}) \lesssim \tilde{f}(x_n)$$

and $1 \leq i \leq n$ be the smallest index at which $f_i(\hat{x}) \neq f_i(x_n)$.

Since i is minimal it follows that $f^{\leq i-1}(\hat{x}) = f^{\leq i-1}(x_n)$. From $\tilde{f}(\hat{x}) \lesssim \tilde{f}(x_n)$ it follows additionally that $f_i(\hat{x}) \lesssim_i f_i(x_n)$. Together this violates the componentwise value uniquely solvable property of the f family, because the ith subproblem has at least two different minima, $f_i(x_n) = f_i(x_i)$ by construction and $f_i(\hat{x})$ because it is $\lesssim_i f_i(x_n)$. So the assumption of the claim's opposite was wrong. ☐

B.1.2. Correctness of the hierarchical optimization algorithm

In this part we directly refer to the algorithm and symbols of [1], Section "3. Hierarchical least squares optimization" and proof that the provided algorithm, Equation (10) and (11) finds a value unique global optimum and respect the task's hierarchy.

Theorem 4 (the algorithm finds a value unique optimal solution)**.**

Let $k \in \mathbb{N}^+$, *the input dimension*
$X := \mathbb{R}^k$, *the input space*
$n \in \mathbb{N}^+$, *the number of tasks*
$k_i \in \mathbb{N}^+, 0 < i \leq n$, *the i-th task space dimension*
$Y_i = \mathbb{R}^{k_i}, 0 < i \leq n$, *the i-th task space*
$\mathbf{b}_i \in Y_i, 0 < i \leq n$, *the i-th target*
$\mathbf{y}_1 \lesssim_i \mathbf{y}_2 := \|\mathbf{y}_1 - \mathbf{b}_i\|_2 \leq \|\mathbf{y}_2 - \mathbf{b}_i\|_2$ *for* $\mathbf{y}_1, \mathbf{y}_2 \in Y_i$

$$\begin{array}{llll} f_i: & X & \to & Y_i \\ & \mathbf{x} & \mapsto & \mathbf{A}_i\mathbf{x} \end{array}$$

Remark. *Fulfilling the hierarchical task in a LS optimal sense for the problem in [1] is equivalent to finding a solution to a corresponding prioritized minimization problem f_i. The transition from a hierarchy of tasks to a prioritized task list must and can always be performed as a first step by combining all tasks of a given level set of the distance to root (priority) in the task tree (hierarchy) into components of complex task. For instance, a pair $(f,a),(g,b)$ of tasks on X with the same priority, such that the goal is to minimize $\|f(x)-a\|_2^2 + \|g(x)-b\|_2^2$, get combined into $(h,(a,b))$ with the goal to minimize $\|h(x)-(a,b)\|_2^2 = \|(f(x),g(x))-(a,b)\|_2^2 = \|f(x)-a\|_2^2 + \|g(x)-b\|_2^2$*

The $\mathbf{x}_i$ in [1] are identical to the $\mathbf{x}_i'$ and <u>not</u> the $\mathbf{x}_i$ below, However, the final solutions, each denoted with $\mathbf{x}$, are identical.

The following recursive algorithm outputs a $\mathbf{x} \in X$ such that $\tilde{f}(\mathbf{x})$ is uniquely minimal in $\tilde{Y}$:

$$\begin{aligned} \mathbf{x}_1 &:= \mathbf{A}_1{}^+\mathbf{b}_1, \\ \mathbf{x}_i &:= \mathbf{N}_i \underbrace{(\mathbf{A}_i\mathbf{N}_i)^+ (\mathbf{b}_i - \mathbf{A}_i\mathbf{x}_{i-1})}_{\mathbf{x}_i'} + \mathbf{x}_{i-1} \text{ for } 1 < i \leq n \\ \mathbf{x} &:= \mathbf{x}_n, \end{aligned}$$

where $(\cdot)^+$ *denotes the Moore-Penrose-Inverse and* $\mathbf{N}_i$ *is the orthogonal projector onto the null space,* $\mathscr{N}([\mathbf{A}_1, .., \mathbf{A}_{i-1}]) := \{\mathbf{x} \in X | \forall_{k=1}^{i-1} \mathbf{A}_k \mathbf{x} = 0\}$.

Proof. We'll apply Theorem 3. To prepare that we have only to check its condition for the f_i and show, that the algorithm executes in deed the described iteration.

- the problem is componentwise value uniquely solvable

  Let $1 \leq i \leq n$
  $\mathbf{y} \in \mathbf{img}(f^{\leq i-1}) \subset \bigtimes_{l=1}^{i-1} Y_l$.

  The minimization of f_i on $V := (f^{\leq i-1})^{-1}(\mathbf{y})$ is value uniquely solvable with respect to $\lesssim_i$ because: $V \ni \mathbf{y}$ is non empty and the intersection of the convex solution sets of the linear equations $\mathbf{A}_j x = \mathbf{y}_j$ for $1 \leq j < i$ and thus convex itself. Hence, $f_i(V)$ is a linear mapping's image of a nonempty convex set and thus nonempty and convex itself. Such a set always contains uniquely a minimizer of the distance to any point, such as D_i according to Lemma 5, which implies the unique solvability of the ith subproblem.

- the given algorithm finds the optimal solution to the minimization problem
  It is sufficient to show that the given algorithm performs the iteration from Theorem 3. Let therefore $1 \leq i \leq n$.

  - $i = 1$
    $\mathbf{x}_1 = \mathbf{A}_1{}^+\mathbf{b}_1$ and with a well known property of the Moore-Penrose-Inverse it follows $\mathbf{x}_1 \in \underset{\mathbf{x} \in X}{\operatorname{argmin}} \|\mathbf{A}_1\mathbf{x} - \mathbf{b}_1\| = \underset{\mathbf{x} \in X}{\operatorname{argmin}} f_1(\mathbf{x})$

  - $i > 1$
    It is to be shown that

    $$\boxed{\mathbf{x}_i \in \underset{f^{\leq i-1}(\mathbf{x}) = f^{\leq i-1}(\mathbf{x}_{i-1})}{\operatorname{argmin}} f_i(\mathbf{x})}$$

    Firstly we take a closer look at the argmin constraint for a $\mathbf{x} \in X$:

$$\begin{aligned}
& f^{\leq i-1}(\mathbf{x}) = f^{\leq i-1}(\mathbf{x}_{i-1}) \\
& \quad \Leftrightarrow f^{\leq i-1}(\mathbf{x}) = f^{\leq i-1}(\mathbf{x}_{i-1}) \\
& \quad \Leftrightarrow \forall_{1 \leq j \leq i-1} f_j(\mathbf{x}) = f_j(\mathbf{x}_{i-1}) \\
& \quad \Leftrightarrow \forall_{1 \leq j \leq i-1} \mathbf{A}_j \mathbf{x} = \mathbf{A}_j \mathbf{x}_{i-1} \\
& \quad \Leftrightarrow \forall_{1 \leq j \leq i-1} \mathbf{A}_j(\mathbf{x} - \mathbf{x}_{i-1}) = 0 \\
& \quad \Leftrightarrow \mathbf{x} - \mathbf{x}_{i-1} \in \mathscr{N}([\mathbf{A}_1, .., \mathbf{A}_{i-1}]) \\
& \quad \Leftrightarrow \exists_{\mathbf{x}' \in X} \mathbf{x} - \mathbf{x}_{i-1} = \mathbf{N}_i \mathbf{x}'
\end{aligned}$$

This suggests a variable substitution $\mathbf{x} = \mathbf{N}_i\mathbf{x}' + \mathbf{x}_{i-1}$ simplifying the argmin expression as follows :

$$\begin{aligned}
&\underset{f^{\leq i-1}(\mathbf{x})=f^{\leq i-1}(\mathbf{x}_{i-1})}{\operatorname{argmin}} f_i(\mathbf{x}) \\
&= \mathbf{N}_i(\underset{\mathbf{x}'\in X}{\operatorname{argmin}} f_i(\mathbf{N}_i\mathbf{x}' + \mathbf{x}_{i-1})) + \mathbf{x}_{i-1} \\
&\overset{\text{def}}{=} \mathbf{N}_i(\underset{\mathbf{x}'\in X}{\operatorname{argmin}} \mathbf{A}_i\mathbf{N}_i\mathbf{x}' + \mathbf{A}_i\mathbf{x}_{i-1}) + \mathbf{x}_{i-1} \\
&\overset{\text{def}}{=} \mathbf{N}_i(\underset{\mathbf{x}'\in X}{\operatorname{argmin}} \|\mathbf{A}_i\mathbf{N}_i\mathbf{x}' + \mathbf{A}_i\mathbf{x}_{i-1} - \mathbf{b}_i\|) + \mathbf{x}_{i-1} \\
&= \mathbf{N}_i\underbrace{(\underset{\mathbf{x}'\in X}{\operatorname{argmin}} \|\mathbf{A}_i\mathbf{N}_i\mathbf{x}' - (\mathbf{b}_i - \mathbf{A}_i\mathbf{x}_{i-1})\|)}_{=:M} + \mathbf{x}_{i-1}
\end{aligned}$$

One element of M is easy to find again due to a well known property of the Moore-Penrose-Inverse :

$$\mathbf{x}' := (\mathbf{A}_i\mathbf{N}_i)^+(\mathbf{b}_i - \mathbf{A}_i\mathbf{x}_{i-1}) \in M$$

Thus it holds that $\mathbf{x}_i = \mathbf{N}_i\mathbf{x}' + \mathbf{x}_{i-1} \in \underset{f^{\leq i-1}(\mathbf{x})=f^{\leq i-1}(\mathbf{x}_{i-1})}{\operatorname{argmin}} f_i(\mathbf{x})$.

□

Lemma 5 (In a nonempty convex subset of a vector space there is always exactly one element minimizing the Euclidean distance to a given vector)**.**

Proof. Assume the claim is wrong.
Let $n \in \mathbb{N}^+$,
$a \in \mathbb{R}^n$,
$\mathrm{d}_a : \mathbb{R}^n \to \mathbb{R}, x \mapsto := \|x - a\|_2$ the Euclidean distance to a,
$M \subseteq \mathbb{R}^n$ convex,
$x, y \in M$, $x \neq y$, both minimizing d_a in M,
$w : [0,1] \to \mathbb{R}; \lambda \mapsto \lambda x + (1-\lambda)y$, and
$\rho : [0,1] \to \mathbb{R}; \lambda \mapsto \mathrm{d}_a(w(\lambda))$
Since M is convex, it follows that $\forall \lambda \in [0,1] : w(\lambda) \in M$. Since x, y have minimal distance to a in M, it follows that $\forall \lambda \in [0,1] : \rho(\lambda) \geq \|x - a\| = \rho(0)$. From the triangle inequality it follows further for $\lambda \in [0,1]$,

$$\begin{aligned}
\rho(\lambda) &= \|\lambda x + (1-\lambda)y - a\| && \text{(B.1)} \\
&= |\lambda(x-a) + (1-\lambda)(y-a)\| && \text{(B.2)} \\
&\leq \lambda\|x-a\| + (1-\lambda)\|y-a\| && \text{(B.3)} \\
&= \lambda\rho(0) + (1-\lambda)\rho(0) && \text{(B.4)} \\
&= \rho(0). && \text{(B.5)}
\end{aligned}$$

And from both inequalities together it follows that ρ is in fact a constant function,

$$\forall \lambda \in [0,1] : \rho(\lambda) = \rho(0). \tag{B.6}$$

Without loss of generality we can assume that $y = 0$. Otherwise we could translate a, x, y, M by $-y$ and rely on the translation invariance of the distances. Combining this with Equation B.6 leads us to the statement

$$\forall \lambda \in [0,1] : \|\lambda x - a\| = \rho(0). \tag{B.7}$$

It follows that $x = 0$[1] as well. And finally, from this that $x = y$ contradicting the assumption. □

---

[1] Can be shown by taking the derivative of the squared equation with respect to λ.

Appendix C

Errata

C.1. Erratum of Online Self-Calibration for Robotic Systems

Unfortunately, the derivation of the estimation algorithm in [4], an important application of the some theoretical contributions of this thesis (and with me as second author), has a major flaw: it is only defined for full-rank $\mathbf{J}_\psi$ by heavily leaning on the expression $(\mathbf{J}_\psi^T \mathbf{J}_\psi)^{-1}$ while it is essential for the algorithm to work especially in the case of rank deficiency. In the following we demonstrate that the same result can be acquired in an all-defined way.

Differently from the paper's approach, we start with the RRQR-decomposition of $\mathbf{J}_\psi$,

$$\mathbf{J}_\psi \quad = \quad \mathbf{Q}\mathbf{R}\mathbf{\Pi}^T, \tag{C.1}$$

where $\mathbf{\Pi} \in \mathbb{R}^{L\times L}$ is a permutation matrix, $\mathbf{R} \in \mathbb{R}^{M\times L}$ an upper triangular matrix, and $\mathbf{Q} \in \mathbb{R}^{M\times M}$ an orthogonal matrix. It follows from the special structure of $\mathbf{R}$ and the orthogonaily of $\mathbf{Q}$ that the unique orthogonal projector matrix onto the image space of $\mathbf{J}_\psi$ is equal to $\mathbf{Q}_1\mathbf{Q}_1^T$, where $\mathbf{Q}_1$ is the first $\operatorname{rank}(\mathbf{J}_\psi)$ columns of $\mathbf{Q}$, as introduced with Equation (20) of [4]. It is well known that the same projector can be expresses with the pseudoinverse of $\mathbf{J}_\psi$, denoted with $\mathbf{J}_\psi^\dagger$, yielding

$$\mathbf{J}_\psi \mathbf{J}_\psi^\dagger = \mathbf{Q}_1\mathbf{Q}_1^T. \tag{C.2}$$

With this pseudoinverse we can define a more general form of the full-rank matrix $\mathbf{C}$ from Equation (16) of [4] which remains well-defined in the case of a rank deficient $\mathbf{J}_\psi$:

$$\mathbf{C} \quad := \quad \begin{pmatrix} \mathbf{I}_\psi & \mathbf{0} \\ -\mathbf{J}_\theta^T \mathbf{J}_\psi^{\dagger\,T} & \mathbf{I}_\theta \end{pmatrix}, \tag{C.3}$$

Left-multiplying this – just as in the paper – to the left-hand side matrix from Equation (15) of [4], repeated here for convenience,

$$\begin{pmatrix} \mathbf{J}_\psi^T \mathbf{J}_\psi & \mathbf{J}_\psi^T \mathbf{J}_\theta \\ (\mathbf{J}_\psi^T \mathbf{J}_\theta)^T & \mathbf{J}_\theta^T \mathbf{J}_\theta \end{pmatrix} \begin{pmatrix} \Delta\psi \\ \Delta\theta \end{pmatrix} = \begin{pmatrix} \mathbf{J}_\psi^T \Delta\mathbf{e} \\ \mathbf{J}_\theta^T \Delta\mathbf{e} \end{pmatrix}, \tag{C.4}$$

yields a similar result to Equation (17):

$$\begin{pmatrix} \mathbf{J}_\psi^T \mathbf{J}_\psi & \mathbf{J}_\psi^T \mathbf{J}_\theta \\ \mathbf{0} & \mathbf{J}_\theta^T \mathbf{J}_\theta - \underbrace{\mathbf{J}_\theta^T \mathbf{J}_\psi^{\dagger\,T} \mathbf{J}_\psi^T}_{=:\mathbf{M}} \mathbf{J}_\theta \end{pmatrix}, \tag{C.5}$$

where the crucial **0** in the lower left follows from the fact

$$\begin{aligned} -\mathbf{J}_\theta^T \mathbf{J}_\psi^{\dagger\,T} \mathbf{J}_\psi^T \mathbf{J}_\psi + \mathbf{I}_\theta (\mathbf{J}_\psi^T \mathbf{J}_\theta)^T &= \mathbf{J}_\theta^T (\mathbf{J}_\psi - \mathbf{J}_\psi^{\dagger\,T} \mathbf{J}_\psi^T \mathbf{J}_\psi) \\ &= \mathbf{J}_\theta^T (\mathbf{J}_\psi - (\mathbf{J}_\psi \mathbf{J}_\psi{}^\dagger)^T \mathbf{J}_\psi) \\ &= \mathbf{J}_\theta^T (\mathbf{J}_\psi - \mathbf{J}_\psi \mathbf{J}_\psi{}^\dagger \mathbf{J}_\psi) \qquad \text{(C.6)} \\ &= \mathbf{0}, \qquad \text{(C.7)} \end{aligned}$$

which follows in turn from the general symmetry of $\mathbf{A}\mathbf{A}^\dagger$ (for (C.6)) and the identity $\mathbf{A}\mathbf{A}^\dagger\mathbf{A} = \mathbf{A}$ (for (C.7)), which are both defining properties of the pseudoinverse of some $\mathbf{A}$.

With this more general **C**, the lengthy derivation in Equation (20) of [4] to find an efficiently computable form of **M** in (C.5), is merely a corollary of the observations we already made, in particular (C.2):

$$\mathbf{M} = \mathbf{J}_\theta^T \mathbf{J}_\psi^{T\,\dagger} \mathbf{J}_\psi^T = \mathbf{J}_\theta^T (\mathbf{J}_\psi \mathbf{J}_\psi{}^\dagger)^T = \mathbf{J}_\theta^T \mathbf{Q}_1 \mathbf{Q}_1^T .$$

This result leads effectively to the same algorithm as presented in the paper – merely derived differently.

No further derivations in [4] implicitly assume full-rank $\mathbf{J}_\psi$. Hence, we showed that and how the algorithm proposed in the papers can indeed be derived for rank deficient $\mathbf{J}_\psi$.

Bibliography

See Section 2.3 for the first bibliography entries.

[19] P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization algorithms on matrix manifolds*. Princeton University Press, 2009.

[20] S. Agarwal, K. Mierle, and Others. Ceres solver, 2013. `https://code.google.com/p/ceres-solver/`.

[21] T. Ainscough, R. Zanetti, J. Christian, and P. D. Spanos. Q-method extended kalman filter. *JGCD*, 2014.

[22] S. Anderson and T. D. Barfoot. Towards relative continuous-time slam. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1033–1040, Karlsruhe, Germany, 6-10 May 2013. Institute of Electrical and Electronics Engineers New York. doi:10.1109/ICRA.2013.6630700.

[23] B. J. Bacon. Quaternion-based control architecture for determining controllability/maneuverability limits. In *AIAA Guidance, Navigation, and Control Conference*, 2012.

[24] I. Y. Bar-Itzhack and Y. Oshman. Attitude determination from vector observations: Quaternion estimation. *IEEE Transactions on Aerospace and Electronic Systems*, AES-21: 128–136, 1985. doi:10.1109/TAES.1985.310546.

[25] I. Y. Bar-Itzhack and J. Reiner. Recursive attitude determination from vector observations: Direction cosine matrix identification. *Journal of Guidance, Control, and Dynamics*, 7(1): 51–56, 1984. doi:10.2514/3.56362.

[26] T. D. Barfoot, J. R. Forbes, and P. T. Furgale. Pose estimation using linearized rotations and quaternion algebra. *Acta Astronautica*, 68(1-2):101–112, 2011. doi:10.1016/j.actaastro.2010.06.049.

[27] T. D. Barfoot, J. R. Forbes, and P. T. Furgale. Pose estimation using linearized rotations and quaternion algebra. *Acta Astronautica*, 68(1–2):101–112, 2011. doi:10.1016/j.actaastro.2010.06.049.

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

[29] R. Battin. *An Introduction to the Mathematics and Methods of Astrodynamics*. AIAA education series. American Institute of Aeronautics and Astronautics, 1987. ISBN 9780930403256. URL `https://books.google.ch/books?id=xZBTAAAAMAAJ`.

[30] O. Bauchau and L. Trainelli. The vectorial parameterization of rotation. *Nonlinear dynamics*, 32(1):71–92, 2003. doi:10.1023/A:1024265401576.

[31] S. Bonnable, P. Martin, and E. Salaün. Invariant extended kalman filter: theory and application to a velocity-aided attitude estimation problem. In *CDC/CCC 2009*, pages 1297–1304. IEEE, 2009.

[32] M. Bosse, R. Zlot, and P. Flick. Zebedee: Design of a spring-mounted 3-d range sensor with application to mobile mapping. *Robotics, IEEE Transactions on*, 28(5):1104–1119, 2012. doi:10.1109/TRO.2012.2200990.

[33] W. G. Breckenridge. Quaternions - proposed standard conventions. Technical Report JPL-IOM-343-79-1199, NASA, 1999.

[34] J. Carroll. The notation and use of quaternions for shuttle ascent steering. *Charles Stark Draper Laboratory Memo: SSV10C-75-47*, 1975.

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

[36] J. C. K. Chou. Quaternion kinematic and dynamic differential equations. *IEEE TRA*, 8(1): 53–64, Feb 1992. ISSN 1042-296X. doi:10.1109/70.127239.

[37] D. Choukroun, I. Y. Bar-Itzhack, and Y. Oshman. A novel quaternion kalman filter. *IEEE Transactions on Aerospace and Electronic Systems*, 42(1):174–190, 2006. doi:10.1109/TAES.2006.1603413.

[38] D. Choukroun, H. Weiss, I. Y. Bar-Itzhack, and Y. Oshman. Kalman filtering for matrix estimation. *IEEE Transactions on Aerospace and Electronic Systems*, 42(1):147–159, Jan. 2006. doi:10.1109/TAES.2006.1603411.

[39] D. Choukroun, H. Weiss, I. Y. Bar-Itzhack, and Y. Oshman. Direction cosine matrix estimation from vector observations using a matrix kalman filter. *IEEE Transactions on Aerospace and Electronic Systems*, 46(1):61–79, Jan. 2010. doi:10.1109/TAES.2010.5417148.

[40] J. L. Crassidis. Sigma-point kalman filtering for integrated gps and inertial navigation. *IEEE Transactions on Aerospace and Electronic Systems*, 42(2):750–756, 2006.

[41] J. L. Crassidis and F. L. Markley. Unscented filtering for spacecraft attitude estimation. *Journal of Guidance, Control, and Dynamics*, 26(4):536–542, July 2003. ISSN 0731-5090, 1533-3884. doi:10.2514/2.5102.

[42] J. L. Crassidis, F. L. Markley, and Y. Cheng. A survey of nonlinear attitude estimation methods. *Journal of Guidance, Control, and Dynamics*, 30(1):12–28, Jan-Feb 2007. doi:10.2514/1.22452.

[43] J. L. Crassidis, F. L. Markley, and Y. Cheng. Survey of nonlinear attitude estimation methods. *JGCD*, 30(1):12–28, 2007.

[44] C. de Boor. *A practical guide to splines*. Springer Verlag, New York, USA, 2001. doi:10.1137/1022106.

[45] A. de Ruiter and J. Forbes. On the solution of wahba's problem on $SO(n)$. *The Journal of the Astronautical Sciences*, 60(1):1–31, 2013. ISSN 0021-9142. doi:10.1007/s40295-014-0019-8.

[46] J. Diebel. Representing attitude: Euler angles, unit quaternions, and rotation vectors. *Matrix*, 58(15-16):1–35, 2006. URL `http://www.swarthmore.edu/NatSci/mzucker1/papers/diebel2006attitude.pdf`.

[47] H. Dong and T. Barfoot. Lighting-invariant visual odometry using lidar intensity imagery and pose interpolation. In K. Yoshida and S. Tadokoro, editors, *Field and Service Robotics*, volume 92 of *Springer Tracts in Advanced Robotics*, pages 327–342. Springer Berlin, 2014. ISBN 978-3-642-40685-0. doi:10.1007/978-3-642-40686-7_22. URL `http://dx.doi.org/10.1007/978-3-642-40686-7_22`.

[48] J. Eidson and K. Lee. Ieee 1588 standard for a precision clock synchronization protocol for networked measurement and control systems. In *Sensors for Industry Conference, 2002. 2nd ISA/IEEE*, pages 98–105. Ieee, 2002.

[49] A. English, P. Ross, D. Ball, B. Upcroft, and P. Corke. Triggersync: A time synchronisation tool. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6220–6226. IEEE, 2015.

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

[51] J. L. Farrell. Attitude determination by kalman filtering. *Automatica*, 6:419–430, 1970. doi:10.1016/0005-1098(70)90057-9.

[52] D. Firoozi and M. Namvar. Analysis of gyro noise in non-linear attitude estimation using a single vector measurement. *IET Control Theory and Applications*, 6(14):2226–2234, 2012. doi:10.1049/iet-cta.2011.0347.

[53] S. Flügge. *Principles of Classical Mechanics and Field Theory / Prinzipien der Klassischen Mechanik und Feldtheorie*. Encyclopedia of Physics. Springer Berlin Heidelberg, 2013. ISBN 9783642459436. URL `https://books.google.ch/books?id=u_GgBgAAQBAJ`.

[54] J. R. Forbes and A. H. J. de Ruiter. Linear-Matrix-Inequality-Based Solution to Wahba's Problem. *Journal of Guidance, Control, and Dynamics*, 38(1):147–151, Apr. 2014. ISSN 0731-5090. doi:10.2514/1.G000132.

[55] J. R. Forbes, A. H. J. de Ruiter, and D. E. Zlotnik. Continuous-time norm-constrained kalman filtering. *Automatica*, 50(10):2546 – 2554, 2014. ISSN 0005-1098. doi:10.1016/j.automatica.2014.08.007.

[56] E. Fresk and G. Nikolakopoulos. Full quaternion based attitude control for a quadrotor. In *2013 ECC, July*, pages 17–19, 2013.

[57] P. Furgale, T. D. Barfoot, and G. Sibley. Continuous-time batch estimation using temporal basis functions. In *2012 IEEE International Conference on Robotics and Automation*, pages 2088–2095, May 2012. doi:10.1109/ICRA.2012.6225005.

[58] P. Furgale, J. Rehder, and R. 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. doi:10.1109/IROS.2013.6696514.

[59] P. T. Furgale, T. D. Barfoot, and G. Sibley. 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, 14-18 May 2012. doi:10.1109/ICRA.2012.6225005.

[60] J. M. Galante, J. Van Eepoel, C. D'Souza, and B. Patrick. Fast kalman filtering for relative spacecraft position and attitude estimation for the raven iss hosted payload. *NTRS*, 2016.

[61] Google. ATAP Project Tango Google, Feb. 2014. URL `http://www.google.com/atap/projecttango/`.

[62] T. Gowers, J. Barrow-Green, and I. Leader. *The Princeton Companion to Mathematics*. Princeton University Press, 2010.

[63] H. F. Grip, T. Fossen, T. A. Johansen, and A. Saberi. Attitude estimation using biased gyro and vector measurements with time-varying reference vectors. *IEEE Transactions on Automatic Control*, 57(5):1332–1338, 2012. doi:10.1109/TAC.2011.2173415.

[64] H. Gui and G. Vukovich. Finite-time angular velocity observers for rigid-body attitude tracking with bounded inputs. *International Journal of Robust and Nonlinear Control*, pages n/a–n/a, 2016. ISSN 1099-1239. doi:10.1002/rnc.3554.

[65] S. Gwak, J. Kim, and F. C. Park. Numerical optimization on the euclidean group with applications to camera calibration. *Robotics and Automation, IEEE Transactions on*, 19 (1):65–74, 2003.

[66] B. Hall. *Lie groups, Lie algebras, and representations: an elementary introduction*, volume 222. Springer Berlin, 2003. doi:10.1007/978-0-387-21554-9.

[67] T. Hamel and R. Mahony. Attitude estimation on $SO(3)$ based on direct inertial measurements. *IEEE Conference on Robotics and Automation, Orlando, FL, May*, pages 2170–2175, 2006. doi:10.1109/ROBOT.2006.1642025.

[68] A. J. Hanson. Visualizing quaternions. In *ACM SIGGRAPH 2005 Courses*, page 1. ACM, 2005.

[69] A. Harrison and P. Newman. Ticsync: Knowing when things happened. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 356–363. IEEE, 2011.

[70] M. Hazewinkel. *Encyclopaedia of Mathematics: Volume 6: Subject Index—Author Index*. Springer Science & Business Media, 2013.

[71] W. Heard. *Rigid Body Mechanics: Mathematics, Physics and Applications*. Physics textbook. Wiley, 2008. ISBN 9783527618828. URL `https://books.google.ch/books?id=wsVcNbOEwCoC`.

[72] J. Hedborg, P.-E. Forssen, M. Felsberg, and E. Ringaby. Rolling shutter bundle adjustment. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1434 –1441, june 2012. doi:10.1109/CVPR.2012.6247831.

[73] C. Hertzberg, R. Wagner, U. Frese, and L. Schröder. Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds. *Information Fusion*, 14(1):57 – 77, 2013. ISSN 1566-2535. doi:10.1016/j.inffus.2011.08.003.

[74] H. Hirschmüller. Stereo processing by semiglobal matching and mutual information. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(2):328–341, 2008.

[75] D. Holm. *Geometric Mechanics: Part II: Rotating, Translating and Rolling*. Imperial College Press, 2008. ISBN 9781911299332. URL `https://books.google.ch/books?id=0s42DwAAQBAJ`.

[76] T. Howard and A. Kelly. Optimal rough terrain trajectory generation for wheeled mobile robots. *International Journal of Robotics Research*, 26(2):141–166, February 2007. doi:10.1177/0278364906075328.

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

[78] M. Hutter. *StarlETH & Co.: Design and control of legged robots with compliant actuation*. PhD thesis, ETH Zurich, 2013.

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

[80] K. J. Jensen. Generalized nonlinear complementary attitude filter. *Journal of Guidance, Control, and Dynamics*, 34(5):1588–1592, 2011. doi:10.2514/1.53467.

[81] J. L. Junkins and J. D. Turner. *Optimal Spacecraft Rotational Maneuvers*, volume 3. Elsevier, 1986. URL `https://books.google.ch/books?id=IqrMHiJWK1YC&q=Quaternion`.

[82] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert. iSAM2: Incremental smoothing and mapping using the Bayes tree. *Intl. J. of Robotics Research, IJRR*, 31(2):217–236, Feb 2012. doi:10.1177/0278364911430419.

[83] J. Kelly and G. 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. doi:10.1177/0278364910382802.

[84] J. Kelly and G. S. Sukhatme. A general framework for temporal calibration of multiple proprioceptive and exteroceptive sensors. In *Experimental Robotics*, pages 195–209. Springer, 2014.

[85] A. Khosravian and M. Namvar. Rigid body attitude control using a single vector measurement and gyro. *IEEE Transactions on Automatic Control*, 57(5):1273–1279, 2012. doi:10.1109/TAC.2011.2174663.

[86] M.-J. Kim, M.-S. Kim, and S. Y. 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. ISBN 0-89791-701-4. doi:10.1145/218380.218486.

[87] M.-S. Kim and K.-W. Nam. Interpolating solid orientations with circular blending quaternion curves. *Computer Aided Design*, 27(5):385–398, 1995.

[88] J. C. Kinsey and L. L. Whitcomb. Adaptive identification on the group of rigid-body rotations and its application to underwater vehicle navigation. *IEEE Transactions on Robotics*, 23(1):124–136, 2007. doi:10.1109/TRO.2006.886829.

[89] A. A. Kirillov, A. A. Kirillov, and A. A. Kirillov. *An introduction to Lie groups and Lie algebras*. Number 113. Cambridge University Press Cambridge, 2008. doi:10.1017/CBO9780511755156.

[90] A. Kolaman and O. Yadid-Pecht. Quaternion structural similarity: A new quality index for color images. *IEEE Transactions on Image Processing*, 21(4):1526–1536, April 2012. ISSN 1057-7149. doi:10.1109/TIP.2011.2181522.

[91] L. Koppel and S. L. Waslander. Manifold geometry with fast automatic derivatives and coordinate frame semantics checking in C++. *CoRR*, abs/1805.01810, 2018. URL `http://arxiv.org/abs/1805.01810`.

[92] J. Kuipers. *Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace, and Virtual Reality*. Mathematical Sciences Series. Princeton University Press, 1999. ISBN 9780691058726. URL `https://books.google.ch/books?id=uG3BQgAACAAJ`.

[93] Lambert, Furgale, Barfoot, and Enright. Field testing of visual odometry aided by a sun sensor and inclinometer. *Journal of Field Robotics*, 29(3):426–444, 2012. ISSN 1556–4967. doi:10.1002/rob.21412.

[94] E. J. Lefferts, F. L. Markley, and M. D. Shuster. Kalman filtering for spacecraft attitude estimation. *JGCD*, 5(5):417–429, 1982.

[95] E. J. Lefferts, F. L. Markley, and M. D. Shuster. Kalman filtering for spacecraft attitude estimation. *Journal of Guidance, Control, and Dynamics*, 5(5):417–429, Sept-Oct 1982. doi:10.2514/3.56190.

[96] E. Lengyel. *Mathematics for 3D Game Programming and Computer Graphics*. Cengage Learning, 2012.

[97] S. Leutenegger, P. Furgale, V. Rabaud, M. Chli, K. Konolige, and R. Siegwart. Keyframe-based visual-inertial slam using nonlinear optimization. Paper presented at Robotics: Science and Systems, Berlin, Germany, 24–28 June 2013. URL `http://roboticsproceedings.org/rss09/p37.pdf`.

[98] M. Li and A. I. Mourikis. Improving the accuracy of ekf-based visual-inertial odometry. In *ICRA 2012*, pages 828–835. IEEE, 2012.

[99] M. Li and A. I. Mourikis. High-precision, consistent EKF-based visual-inertial odometry. *International Journal of Robotics Research*, 32(6):690–711, May 2013. doi:10.1177/0278364913481251.

[100] M. Li, H. Yu, X. Zheng, and A. I. Mourikis. High-fidelity sensor modeling and self-calibration in vision-aided inertial navigation. In *ICRA 2014*, pages 409–416. IEEE, 2014.

[101] S. Lovegrove, A. Patron-Perez, and G. 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*, pages 93.1–93.12. British Machine Vision Association Press Durham, 2013. doi:10.5244/C.27.93.

[102] R. Mahony, T. Hamel, and J.-M. Pflimlin. Complementary filter design on the special orthogonal group $SO(3)$. 44th *IEEE Conference on Decision and Control and the European Control Conference, Seville, Spain, December 12-15*, pages 1477–1484, 2005. doi:10.1109/CDC.2005.1582367.

[103] R. Mahony, T. Hamel, and J.-M. Pflimlin. Nonlinear complementary filters on the special orthogonal group. *IEEE Transactions on Automatic Control*, 53(5):1203–1218, 2008. doi:10.1109/TAC.2008.923738.

[104] F. Markley and J. Crassidis. *Fundamentals of Spacecraft Attitude Determination and Control*. Space Technology Library. Springer New York, 2014. ISBN 9781493908028.

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

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

[107] F. L. Markley. 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, Riomaggiore, Italy, July 2004.

[108] A. Martinelli. Vision and imu data fusion: Closed-form solutions for attitude, speed, absolute scale, and bias determination. *IEEE Transactions on Robotics*, 28(1):44–60, 2012.

[109] J. R. Martins, P. Sturdza, and J. J. Alonso. The complex-step derivative approximation. *ACM Transactions on Mathematical Software (TOMS)*, 29(3):245–262, 2003.

[110] M. T. Mason. *Mechanics of Robotic Manipulation*. MIT Press, 2016. ISBN 9780262263740.

[111] J. Maye, P. Furgale, and R. Siegwart. Self-supervised calibration for robotic systems. In *2013 IEEE Intelligent Vehicles Symposium (IV)*, pages 473–480, June 2013. doi:10.1109/IVS.2013.6629513.

[112] J. S. McCabe, A. J. Brown, K. J. DeMars, and J. M. Carson III. Comparison of factorization-based filtering for landing navigation. *NTRS*, 2017.

[113] 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. doi:10.1002/rob.21444.

[114] F. Mirzaei and S. Roumeliotis. A Kalman filter-based algorithm for IMU-camera calibration: Observability analysis and performance evaluation. *Robotics, IEEE Transactions on*, 24(5):1143–1156, October 2008. ISSN 1552-3098. doi:10.1109/TRO.2008.2004486.

[115] J. B. Moore and P. K. Tam. Fixed-lag smoothing for nonlinear systems with discrete measurements. *Information Sciences*, 6:151–160, 1973. doi:10.1016/0020-0255(73)90032-7.

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

[117] D. Mortari. Second estimator of the optimal quaternion. *Journal of Guidance, Control and Dynamics*, 23(5):885–888, 2000. doi:10.2514/2.4618.

[118] A. I. Mourikis, N. Trawny, S. I. Roumeliotis, A. E. Johnson, A. Ansar, and L. Matthies. Vision-aided inertial navigation for spacecraft entry, descent, and landing. *IEEE Transactions on Robotics*, 25(2):264–280, April 2009. doi:10.1109/TRO.2009.2012342.

[119] R. Mukundan. Quaternions: From classical mechanics to computer graphics, and beyond. In *7th ATCM*, pages 97–105, 2002.

[120] R. M. Murray and S. S. Sastry. *A mathematical introduction to robotic manipulation*. CRC press, 1994.

[121] R. M. Murray, Z. Li, S. S. Sastry, and S. S. Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC press, 1994. URL `http://www.cds.caltech.edu/~murray/books/MLS/pdf/mls94-complete.pdf`.

[122] N. NAIF. Quaternions white paper, 2003. URL `ftp://naif.jpl.nasa.gov/pub/naif/misc/Quaternion_White_Paper/Quaternions_White_Paper.pdf`.

[123] L. Oth, P. Furgale, L. Kneip, and R. Siegwart. Rolling shutter camera calibration. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 1360–1367, 25–27 June 2013. doi:10.1109/CVPR.2013.179.

[124] F. C. Park and B. Ravani. Smooth invariant interpolation of rotations. *ACM Trans. Graph.*, 16:277–295, July 1997. ISSN 0730-0301. doi:10.1145/256157.256160.

[125] E. Pervin and J. A. Webb. Quaternions in computer vision and robotics. Technical report, Carnegie Mellon University, 1982. URL `http://repository.cmu.edu/cgi/viewcontent.cgi?article=3509&context=compsci`.

[126] P. Piniés, T. Lupton, S. Sukkarieh, and J. D. Tardós. Inertial aiding of inverse depth slam using a monocular camera. In *ICRA 2007*, pages 2797–2802. IEEE, 2007.

[127] D. Piponi. Automatic differentiation, c++ templates, and photogrammetry. *Journal of Graphics Tools*, 9(4):41–55, 2004.

[128] D. Poli and T. Toutin. Review of developments in geometric modelling for high resolution satellite pushbroom sensors. *The Photogrammetric Record*, 27(137):58–73, 2012. ISSN 1477-9730. doi:10.1111/j.1477-9730.2011.00665.x.

[129] S. Z. Queen. A kalman filter for mass property and thrust identification of the spin-stabilized magnetospheric multiscale formation. *NTRS*, 2015.

[130] J. Quesada, J. U. Llano, R. Sebastian, M. Castro, and E. Jacob. Evaluation of clock synchronization methods for measurement and control using embedded linux sbcs. In *Remote Engineering and Virtual Instrumentation (REV), 2012 9th International Conference on*, pages 1–7. IEEE, 2012.

[131] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, 2009.

[132] L. B. Rall. Perspectives on automatic differentiation: Past, present, and future? In *Automatic Differentiation: Applications, Theory, and Implementations*, pages 1–14. Springer, 2006.

[133] L. B. Rall and G. F. Corliss. An introduction to automatic differentiation. In M. Berz, C. H. Bischof, G. F. Corliss, and A. Griewank, editors, *Computational Differentiation: Techniques, Applications, and Tools*, pages 1–17. SIAM, Philadelphia, PA, 1996.

[134] C. E. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA, USA, 2006.

[135] J. Rehder, P. Beardsley, R. Siegwart, and P. 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.

[136] J. Rehder, R. Siegwart, and P. Furgale. A general approach to spatiotemporal calibration in multisensor systems. *IEEE Transactions on Robotics*, 32(2):383–398, 2016.

[137] E. Ringaby and P.-E. Forssén. Efficient video rectification and stabilisation for cell-phones. *International Journal of Computer Vision*, pages 1–18, 2011. ISSN 0920-5691. doi:10.1007/s11263-011-0465-8. 10.1007/s11263-011-0465-8.

[138] L. Rodman. *Topics in Quaternion Linear Algebra*. Princeton University Press, 2014.

[139] C. Roussillon, A. Gonzalez, J. Solà, et al. Rt-slam: A generic and real-time visual slam implementation. In *ICVS*, pages 31–40. Springer, 2011.

[140] I. Sa, S. Hrabar, and P. Corke. Outdoor flight testing of a pole inspection uav incorporating high-speed vision. In *Field and Service Robotics*, pages 107–121. Springer, 2015.

[141] A. L. Schwab. Quaternions, finite rotation and euler parameters. Technical report, Delft University of Technology, 2002. URL `http://bicycle.tudelft.nl/schwab/Publications/quaternion.pdf`.

[142] A. L. Schwab and J. Meijaard. How to draw euler angles and utilize euler parameters. In *IDETC/CIE*, pages 10–13, 2006.

[143] R. Shivarama and E. P. Fahrenthold. Hamilton's equations with euler parameters for rigid body dynamics modeling. *JDSMC*, 126(1):124–130, 2004.

[144] K. Shoemake. Quaternion calculus and fast animation, computer animation: 3-d motion specification and control. In *Proc. SIGGRAPH'87*, 1987.

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

[146] M. D. Shuster. Survey of attitude representations. *Journal of the Astronautical Sciences*, 41:439–517, Oct 1993. URL `http://malcolmdshuster.com/Pub_1993h_J_Repsurv_scan.pdf`.

[147] M. D. Shuster. The nature of the quaternion. *The Journal of the Astronautical Sciences*, 56(3):359–373, 2008. ISSN 0021-9142. doi:10.1007/BF03256558.

[148] M. D. Shuster and S. D. Oh. Three-axis attitude determination from vector observations. *Journal of Guidance, Control, and Dynamics*, 4(1):70–77, January-February 1981. doi:10.2514/3.19717.

[149] B. Siciliano and J. . E. Slotine. A general framework for managing multiple tasks in highly redundant robotic systems. In *Fifth International Conference on Advanced Robotics 'Robots in Unstructured Environments*, pages 1211–1216 vol.2, June 1991. doi:10.1109/ICAR.1991.240390.

[150] J. Sola. Quaternion kinematics for the error-state kf. *LAAS-CNRS, Toulouse, France, Tech. Rep*, 2012. URL `http://www.iri.upc.edu/people/jsola/JoanSola/objectes/notes/kinematics.pdf`.

[151] H. Sommer, I. Gilitschenski, M. Bloesch, S. M. Weiss, R. Siegwart, and J. Nieto. Why and how to avoid the flipped quaternion multiplication, 2018. URL `https://arxiv.org/abs/1801.07478`.

[152] B. Stevens and F. Lewis. *Aircraft Control and Simulation*. Wiley, 2003. ISBN 9780471371458. URL `https://books.google.ch/books?id=T0Ux6av4btIC`.

[153] H. Strasdat, J. Montiel, and A. J. Davison. Visual slam: Why filter? *Image and Vision Computing*, 30(2):65–77, 2012. ISSN 0262-8856. doi:10.1016/j.imavis.2012.02.009.

[154] Z. Taylor and J. Nieto. Motion-based calibration of multimodal sensor extrinsics and timing offset estimation. *IEEE Transactions on Robotics*, 32(5):1215–1229, 2016.

[155] C. H. Tong and T. D. Barfoot. 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, 6-10 May 2013. doi:10.1109/ICRA.2013.6631321.

[156] C. H. Tong, P. T. Furgale, and T. D. Barfoot. Gaussian process Gauss-Newton for non-parametric simultaneous localization and mapping. *International Journal of Robotics Research*, 32(5):507–525, 2013. doi:10.1177/0278364913478672.

[157] N. Trawny and S. I. Roumeliotis. Indirect Kalman filter for 3D attitude estimation. Technical Report 2005-002, University of Minnesota, Dept. of Comp. Sci. & Eng., Mar 2005.

[158] J. Vandersteen, M. Diehl, C. Aerts, and J. Swevers. Spacecraft attitude estimation and sensor calibration using moving horizon estimation. *Journal of Guidance, Control, and Dynamics*, 36(3):734–742, May 2013. ISSN 0731-5090, 1533-3884. doi:10.2514/1.58805.

[159] J. Vince. *Quaternions for Computer Graphics*. SpringerLink : Bücher. Springer London, 2011. ISBN 9780857297600. URL `https://books.google.ch/books?id=CDgrYqDsSBAC`.

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

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

[162] D. Yazell. Origins of the unusual space shuttle quaternion definition. In *Aerospace Sciences Meetings, pages–. American Institute of Aeronautics and Astronautics*, volume 66, 2009.

[163] R. Zanetti, M. Majji, R. Bishop, and D. Mortari. Norm-constrained kalman filtering. *Journal of Guidance, Control, and Dynamics*, 32(5):1458–1465, Sept-Oct 2009. doi:10.2514/1.G000344.

[164] L. Zhang, Z. Liu, and C. H. Xia. Clock synchronization algorithms for network measurements. In *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 1, pages 160–169. IEEE, 2002.

Curriculum Vitae

Hannes Sommer
born December 29, 1980
Tübingen, Germany

2019–	*Sevensense Robotics AG* Robotics engineer
2014–2019	*ETH Zurich, Switzerland* Doctoral studies at the Autonomous Systems Lab; Supervised by Prof. Roland Siegwart and Prof. Stephan Weiss
2012–2014	*ETH Zurich, Switzerland* Research Assistant at the Autonomous Systems Lab;
2010–2013	*Interactive Scape GmbH* Software engineering in visual computing
2009–2010	*TU Berlin, Germany* Diploma in Mathematics
2002–2010	*TU Berlin, Germany* Study of Mathematics and Physics
2000–2001	*Institute for Water and River Basin Management, KIT, Germany* Civil service
1991–2000	*Eduard Spranger Gymnasium Landau (Pfalz), Germany* Matura in Mathematics, Physics, and social studies