HAL open science

Deep learning, inertial measurements units, and odometry : some modern prototyping techniques for navigation based on multi-sensor fusion

Martin Brossard

▶ To cite this version:

Martin Brossard. Deep learning, inertial measurements units, and odometry : some modern prototyping techniques for navigation based on multi-sensor fusion. Machine Learning [cs.LG]. Université Paris sciences et lettres, 2020. English. NNT : 2020UPSLM067 . tel-03262132

HAL Id: tel-03262132

https://pastel.hal.science/tel-03262132v1

Submitted on 16 Jun 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

PSL UNIVERSITÉ PARIS

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PSL

Préparée à MINES ParisTech

Deep learning, Inertial Measurements Units, and Odometry

Some Modern Prototyping Techniques for Navigation Based on Multi-Sensor Fusion

Deep Learning, Centrales Inertielles et Odométrie

Techniques de Prototypages Modernes pour la Navigation basées sur la Fusion de Capteurs

Soutenue par

Martin Brossard

Le 22 Septembre 2020

École doctorale n°621

Ingénierie des Systèmes, Matériaux, Mécanique, Energétique

Spécialité

Informatique temps réel, robotique, automatique

Composition du jury :

Dana KULIC Professeur, Université de Monash	*Rapporteur*
Tim BARFOOT Professeur, Université d Toronto	*Rapporteur*
David FILLIAT Professeur, ENSTA	*Président*
Yacine CHITOUR Professeur, CentraleSupélec	*Examinateur*
Silvère BONNABEL Professeur, Université de Nouvelle-Calédonie	*Directeur de thèse*
Axel BARRAU Ingénieur de Recherche, Safran	*Examinateur*

MINES ParisTech | PSL

Acknowledgements/Remerciements

First of all, I thank the thesis reporters *Dana Kulic* and *Tim Barfoot* for accepting to report this thesis. It is an honor for me to consider their insightful suggestions to make this manuscript clearer. I switch to 🇫🇷 pour remercier *David Filliat* d'avoir présider la soutenance et *Yacine Chitour* en temps que membre du jury et ancien prof.

Je resterai *ad vitam æternam* redevable envers *Silvère* pour m'avoir fait confiance et guidé durant toute cette aventure. Je ne mesure pas encore la chance d'avoir eu un directeur de thèse avec un tel niveau de bienveillance! Merci ∞-ment à *Axel* pour m'avoir co-encadré et partagé avec moi ses idées les plus audacieuses et son optimisme inaltérable.

Je tiens à remercier du fond du ♥ toutes les personnes qui ont rendu mon travail quotidien aux Mines agréable, pour ne pas dire génial. En premier lieu, je remercie *Paul* pour ses conseils avisés et sa sympathie. Tant pour la qualité de leur conseils que la bonne humeur qu'ils répandent, Merci aux anciens doctorants *Marion*, *Xavier*, *Aubrey*, *Florent*, *Hughes Phillip*, *Daniele*, *Michelle*, *Guillaume*, aux ± récent doctorants *Marin*, *Grégoire*, *Sami*, *Sofiane*, aux encore nouveaux doctorants *Colin*, *Jean-Pierre*, *Pierre*, ainsi qu'à *Martin*, *Emmanuel*, *Olivier* et *Eren*.

Merci ensuite aux membres permanants du Centre de Robotique, à commencer par *Arnaud* pour m'avoir accueilli au laboratoire, *Christophe*, *Alexis*, *Christine*, *Jean-Emmanuel*, *Cyril* & *Kelly*, *François*, *Amaury*, *Fabien*, *Bogdan*, *Simon* et *Brigitte*. Je remercie ×2 *Christine* pour son assistance et *Jean-Emmanuel* pour son expertise.

En dehors des murs de l'école, merci à *Mattieu* pour l'avoir rencontré au 🇨🇦 et une seconde fois Merci à *Hughes*, *Sami* et *Sofiane* pour les séjours 🎿. Merci aux membres et doctorants de l'université de la Nouvelle Calédonie pour l'accueil chaleureux qu'ils m'ont réservé lors de mon passage en 🇳🇨. Merci avec un peu d'avance à l'équipe d'*Arcturus Industries* qui rend la fin de thèse et l'après-thèse particulièrement excitants.

Une partie de ma thèse s'est jouée au Palais de la Découverte. Merci à *Guillaume*, *Laure* et *Robin*, de l'unité de maths, qui m'ont fait découvrir pleins de choses. Merci aussi aux coaches de la semaine des jeunes chercheurs *Aurélié*, *Véronique*, *Ludovic*, *Élodie* et aux doctorants *Amina*, *Héloïse*, *Jérémy*, *Delphine*, *Auréaliane* et *Damien* qui m'ont fait -eux-aussi- découvrir de nouvelles choses.

Un grand merci à mes amis pour me rendre tout simplement heureux, avec une pensée spécifique à *Bruno*, *Medhi*, *Guillaume*, *Sarah* qui ont ou vont bientôt soutenir leur thèse. Merci aussi à *Nabil* pour m'avoir pré-formé à la thèse, et *Fethi* pour m'avoir fourni un 🏠 des plus agréables.

Un grand merci à mes parents, à mes frères *Baptiste* & *Clément* pour m'avoir hébergé la semaine précédent la soutenance, pour leurs encouragements et leur soutien. Comme mon futur s'écrit en 🇪🇸, je finis par "*Espero con interés la continuación*". ☺

23 Septembre 2020 *Martin Brossard*

Contents

Acknowledgements/Remerciements **1**

List of Publications **7**

1 Introduction **9**
1 Contributions of the Thesis 10
2 Organization of the Manuscript 10

I Unscented Kalman Filtering on Manifold & Lie Groups **15**

2 Introduction to Part I **17**
1 The Problem of Bayesian Filtering and the Kalman Filter 17
2 Nonlinear Filtering: the Extended Kalman Filter (EKF) 18
3 Nonlinear Filtering: the Unscented Kalman Filter (UKF) 19
4 Introduction to Filtering on Manifolds 19
5 Matrix Lie Groups 22
6 Nonlinear & Invariant Observers on Lie Groups 24
7 Invariant Extended Kalman Filter 25

3 Exploiting Symmetries to Design EKFs with Consistency Properties **29**
1 Introduction 30
2 Contributions 32
3 General Theory 33
4 Application to Multi-Sensor Fusion for Navigation 37
5 Simulation Results 40
6 Conclusion 43

4 A New Approach to Unscented Kalman Filtering on Manifolds **45**
1 Introduction 45
2 Unscented Kalman Filtering on Parallelizable Manifolds 47
3 Application to UKF on Lie Groups 51
4 UKF-M Implementation 53
5 Extension to General Manifolds 56
6 Concluding Remarks 58

5 Invariant Kalman Filtering for Visual Inertial SLAM **59**
1 Introduction 59
2 Visual Inertial SLAM Problem Modeling 61
3 Proposed Algorithms 62

4 Simulation Results . . . 65
5 Experimental Results . . . 67
6 Conclusion . . . 69

6 Unscented Kalman Filter on Lie Groups for Visual Inertial Odometry 73
1 Introduction . . . 73
2 Problem Modeling . . . 75
3 Unscented Based Inferred Jacobian for UKF-LG Update . . . 76
4 Proposed Filters . . . 77
5 Experimental Results . . . 81
6 Conclusion . . . 83

II Measurement Noise Estimation for Kalman Filter Tuning 85

7 Introduction to Part II 87
1 Illustrative Example . . . 88
2 Content of Part II . . . 88

8 AI-IMU Dead-Reckoning 91
1 Introduction . . . 92
2 Relation to Previous Literature . . . 93
3 IMU and Problem Modelling . . . 94
4 Kalman Filtering with Pseudo-Measurements . . . 96
5 Proposed AI-IMU Dead-Reckoning . . . 98
6 Experimental Results . . . 101
7 Conclusion . . . 107

9 A New Approach to 3D ICP Covariance Estimation 111
1 Introduction . . . 112
2 Proposed Approach . . . 114
3 Practical Covariance Computation . . . 117
4 Experimental Results . . . 120
5 Complementary Experimental Results . . . 122
6 Conclusion . . . 125

III Using Deep Learning to Extract Information from an IMU 127

10 Introduction to Part III 129
1 Relations between Neural Networks and the Kalman Filter . . . 129
2 Designed “Deep” Kalman Filter with Neural Networks . . . 130
3 Adding Information to Kalman Filter with Neural Networks . . . 130

11 RINS-W: Robust Inertial Navigation System on Wheels 133
1 Introduction . . . 134
2 Inertial Navigation System & Sensor Model . . . 134
3 Specific Motion Profiles for Wheeled Systems . . . 136
4 Proposed RINS-W Algorithm . . . 138
5 Results on Car Dataset . . . 141
6 Conclusion . . . 145

12 Denoising IMU Gyroscopes with Deep Learning for Attitude Estimation 147
1 Introduction 148
2 Kinematic & Low-Cost IMU Models 149
3 Learning Method for Denoising the IMU 150
4 Experiments 153
5 Discussion 160
6 Conclusion 161

13 Additional Results for Inertial Navigation of Cars 163
1 Introduction 163
2 Experimental Results 163
3 Further Experimental Results 170
4 Conclusion 175

14 Conclusion of the Thesis 177

Bibliography 194

15 Appendix: Enhancing Kalman Filter with Deep Learning in Practice 195
1 Introduction 195
2 Problem Modelling 196
3 Enhancing the Kalman Filter with Deep Learning 197
4 Results 203
5 Conclusion 204

List of Publications

This thesis lead to the publication of four peer-reviewed international journal papers and six articles in the proceedings of international peer-reviewed conferences.

2017

- M. Brossard, S. Bonnabel, and J-P. Condomines. “Unscented Kalman Filtering on Lie Groups." *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 2485-2491. doi: 10.1109/IROS.2017.8206066.

2018

- M. Brossard, S. Bonnabel, and A. Barrau. “Invariant Kalman Filtering for Visual Inertial SLAM." *21st International Conference on Information Fusion*. IEEE, 2018, pp. 2021-2028. doi: 10.23919/ICIF.2018.8455807.
- M. Brossard, S. Bonnabel, and A. Barrau. “Unscented Kalman Filter on Lie Groups for Visual Inertial Odometry." *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 649-655. doi: 10.1109/IROS.2018.8593627.
- M. Brossard, A. Barrau, and S. Bonnabel. “Exploiting Symmetries to Design EKFs with Consistency Properties for Navigation and SLAM." *IEEE Sensors Journal* 19.4: 1572-1579, 2018. doi: 10.1109/JSEN.2018.

2019

- M. Brossard, and S. Bonnabel. “Learning Wheel Odometry and IMU Errors for Localization." *International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 291-297, doi: 10.1109/ICRA.2019.8794237.
- M. Brossard, A. Barrau, and S. Bonnabel. “RINS-W: Robust Inertial Navigation System on Wheels." *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 2068-2075. doi: 10.1109/IROS40897.2019.8968593.

2020

- M. Brossard, A. Barrau, and S. Bonnabel. “A Code for Unscented Kalman Filtering on Manifolds (UKF-M)." *International Conference on Robotics and Automation (ICRA)*. IEEE, 2020.

- M. Brossard, S. Bonnabel, and A. Barrau. "A New Approach to 3D ICP Covariance Estimation." *IEEE Robotics and Automation Letters* 5.2: 744-751, 2020. doi: 10.1109/LRA.2020.2965391.
- M. Brossard, A. Barrau, and S. Bonnabel. "AI-IMU Dead-Reckoning." *IEEE Transaction on Intelligent-Vehicles*, 2020, doi: 10.1109/TIV.2020.2980758.
- M. Brossard, S. Bonnabel, and A. Barrau. "Denoising IMU Gyroscopes with Deep Learning for Open-Loop Attitude Estimation." *IEEE Robotics and Automation Letters*, 2020.
- M. Brossard, A. Barrau, P. Chauchat, and S. Bonnabel, "Associating Uncertainty to Extended Poses for on Lie Group IMU Preintegration with Rotating Earth." *arXiv preprint arXiv:2007.14097*, 2020.

CHAPTER 1

Introduction

This thesis deals with state estimation of vehicles or robots that navigate and which are equipped with various sensors. The first practical digital state estimators appeared in the 1960s, during the advent of the space age and digital computers. The Kalman filter was introduced by Rudolph Emil Kalman in 1960 as a theoretical optimal state estimator and was immediately applied to navigation for the Apollo project, to estimate the trajectories of the manned space capsule when going to the Moon and back. More generally the Kalman filter is a tool to estimate the state of a dynamical system had a huge impact on the engineering world, especially the aerospace community. This is dramatically illustrated by awarding Kalman for the development of his optimal digital technique for state estimation with the 2008 Draper Prize, which is one of three prizes that constitute the “Nobel prizes of engineering".

Sixty years later, and though the field of Kalman filter based navigation may be considered as mature, there is still ongoing research on the subject of digital state estimation to estimate the trajectories of vehicles in real time. There are various reasons why.

From the theoretical point of view, researchers have long tried to overcome the shortcomings of the Kalman filter when applied to nonlinear systems. Indeed, the Kalman filter is optimal for state estimation of *linear* systems. However, systems are hardly linear in practice, especially in the field of navigation, and the simple extension of the Kalman filter, the Extended Kalman Filter (EKF), based on linearizations about the estimated trajectory turns out to work well in many cases (e.g., to estimate the trajectory of the space capsule) but may completely fail when confronted with challenging problems. This has prompted other Kalman filter variants, like the unscented Kalman filter, as well as different approaches such as particle filters that were very successful in the signal processing community and the optimization-based methods known as smoothing that is currently very successful in the robotics community.

Another reason why there is still research on the subject of state estimation for navigation is the recent ubiquity of cameras (vision) as a low-cost and very informative sensor. They return a high dimensional information that is not easily related to the state, and pose quite a number of problems (both practical and theoretical) when one wants to fuse vision with a more classical sensor suite, such as inertial sensors, wheel-speed sensors, and Global Navigation Satellite Systems (GNSS). Navigating without absolute measurements (as those provided by GNSS), but only with measurements that are relative to an unknown environment is referred to as the problem of Simultaneous Localization And Mapping (SLAM) in the robotics community, and has proved a very

challenging application to the extended Kalman filter, even at a theoretical level.

Finally, (Kalman) filtering is based on physical models, for the trajectory followed by the vehicle where a physical model of the state evolution is used, and to mathematically relate the state to the sensors' measurements. Filtering also relies on some parameters that need to be tuned by the user. As can be expected, the models are idealized and do not completely match physical reality. In the same way, tuning the parameters is difficult as the parameters are supposed to represent the magnitude of the uncertainty, but this uncertainty is largely unknown and relies on idealized statistical models that do not match what is actually unknown in practice.

1 1 Contributions of the Thesis

The goal of this thesis is to contribute to the state-of-the-art regarding Kalman filtering for navigation, by focusing on various challenging aspects. The contributions focus on two different routes. The first route consists in building on the recent variant of the Kalman filter called Invariant Extended Kalman Filter (IEKF) to address challenging issues, namely the inconsistency of EKF for SLAM, navigation with vision sensors, and the more general question of Kalman filtering in the case where the state space is not an Euclidean space (filtering on manifolds). The second route consists in using recent tools from the field of Artificial Intelligence (AI), namely (deep) neural networks, to improve Kalman filters, notably to automatically relate sensors' measurements to the state, and to tune the filter "optimally", that is, use machine learning techniques to find a dynamical tuning strategy of the Kalman filter parameters that best matches the data.

This goal is ambitious, and it leads to contributions that all revolve around the Kalman filter but which intervene at various levels and which differ in nature at times. As a result, the reading of the present manuscript might prove a little difficult, in the sense that contributions are to some extent "scattered". To help the reader better situate where the contributions lie, we have made a diagram, represented on Figure 1.1, where the thesis contributions are depicted in the green rectangles. The goal of the state estimator is to compute as best as possible accurate state estimate, and to convey the extent of uncertainty about its own estimate, which is encoded in the state covariance matrix as concerns the Kalman filter. The success in practice of a state estimator relies on three main components: correct physical and uncertainty models, an efficient estimator for the given models, and finely tuned uncertainty magnitude, and estimator parameters. Those correspond to various blocks on the figure, with relations between them that appear. Although it might be too early for the reader being unfamiliar with filtering to look much into the diagram, the reader should bear in mind its existence and refer to it throughout the manuscript when need be.

A last note regarding the present work. Our work is essentially concerned with the production of novel algorithms in the field of navigation and state estimation for robotics or aerospace. In many cases, our algorithms are tested and benchmarked on publicly available datasets. In this spirit, we have strived for the production of open source code, and most of our algorithms have been made publicly available.

2 2 Organization of the Manuscript

This thesis is divided into three parts. To some extent, the parts can be read independently. Note, this comes at the price of several repetitions throughout the manuscript.

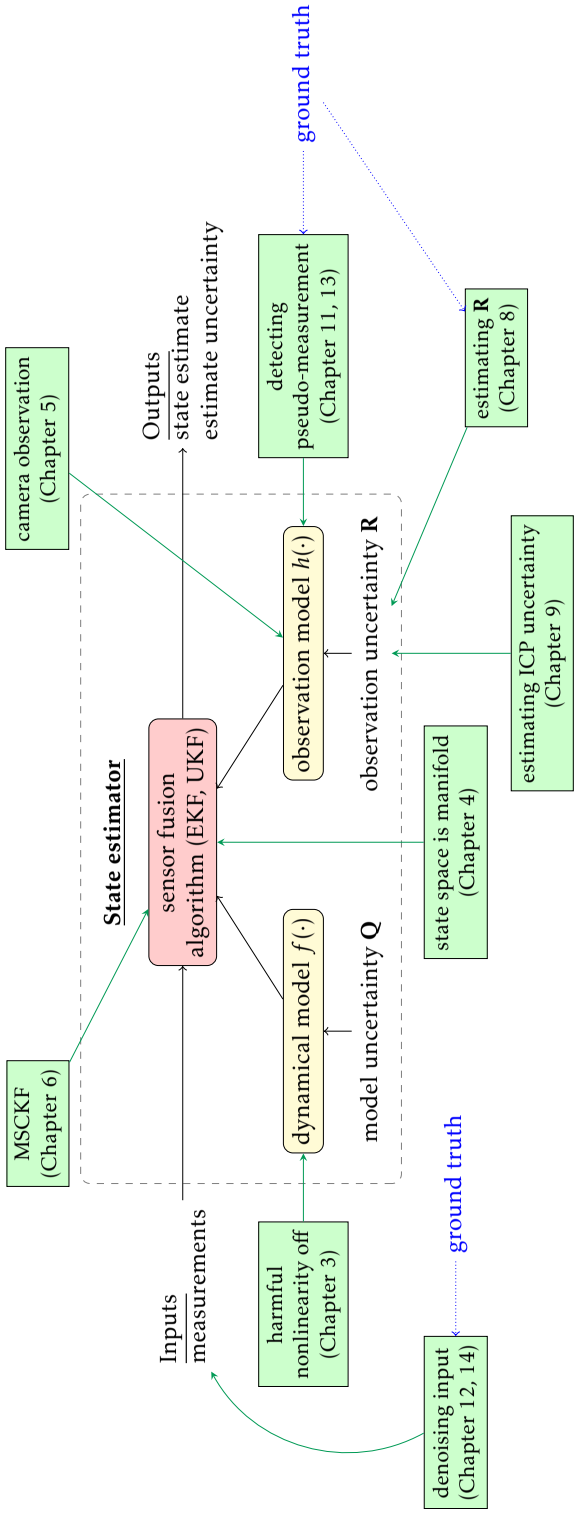


Figure 1.1: Schema of a general state estimation system, along with the contributions of the present thesis (green rectangles). The state estimator computes state estimates along with their uncertainty in function of measurement inputs. The state estimator is divided into a sensor fusion algorithm (red), and models (yellow), where models are parametrized by uncertainty (noise covariance matrices) $\mathbf{Q}$ and $\mathbf{R}$.

Part I: the first part is dedicated to theoretical contributions to the field of nonlinear Kalman filtering, and their application to the context where one navigates with camera and seeks to fuse inertial sensors with vision sensors. In Chapter 3, we focus on the (abstract) theoretical problem of SLAM, that is, navigating with relative measurements only in a totally unknown environment. For this problem, the EKF has been long known as a debateable solution, owing to its inconsistency in this context. We revisit the theoretical problem of EKF inconsistency for SLAM, and show how the use of the recently introduced Kalman filter variant IEKF may resolve those issues.

Then, in Chapter 4, we study how the well established variant of EKF called Unscented Kalman Filter (UKF), may be adapted in the case where the state does not belong to a vector space, but to a more complicated space, namely a manifold. The most basic example, and also a historical motivation, is attitude estimation, where the state is a rotation matrix. Our method applies to all Lie groups, and in particular allows to propose an unscented version of the IEKF. This is desirable as UKF readily allows fast prototyping, in the sense that it spares the sometimes difficult computation of Jacobians that must be performed in the EKF methodology. We have proposed a code available online that allows devising and benchmarking algorithms quickly.

Finally, Chapters 5 and 6 are devoted to the problem of SLAM in the presence of vision sensors. We build on the state-of-the art Kalman filter based solution, namely the Multi-State Constrained Kalman Filter (MSCKF), to attack the problem. Our contributions essentially consists in building an invariant version of this filter, that then comes with consistency properties using the results of Chapter 3, and to propose a computationally efficient unscented version, building on our work described in Chapter 4.

Part II: the second part focuses on the assessment of uncertainty of sensors. Although each sensor comes with specifications, some issues arise. For instance when a wheeled vehicle navigates some assumptions may be made regarding the motion, such as the fact the vertical velocity (in the vehicle's frame) is negligible, and the lateral velocity is small (wheeled vehicles tend to roll without slip although slip is inevitable). This information may be used by the Kalman filter in the form of a pseudo-measurement. However, this information is uncertain: in turns the vehicle slips more than in straight lines. As a result, a machine (deep) learning algorithm may "learn" the extent of uncertainty that lies in the formulated assumption, and relate it to the inertial measurements. This is the object of Chapter 8, where we show our method based only on inertial sensors competes with state-of-the art methods that use inertial sensors plus vision.

Another type of uncertainty may stem from the nature of the observation. Indeed mobile robotics systems are increasingly equipped with laser scanners that return depth images of the environment. An algorithm, called Iterative Closest Point (ICP), allows estimating the displacement of the robot between two scans. It is then important for the filter that estimates the robot's trajectory based on various sensors (including the scanners) to know the amount of uncertainty present in the ICP's estimated displacement. Even if one knows the precision of the depth measurements from the scanner, it is not easily related to the ICP's estimate precision. In Chapter 9, we propose a simple theoretical approach based on statistics and an algorithm for this problem which goes beyond existing solutions, and overcomes their main drawbacks.

Part III: the third part is wholly focused on the use of deep learning, which is a subset of modern Artificial Intelligence (AI), as a tool to extract pertinent information from inertial measurements, and to use this information to navigate. In all cases, the method

is supervised, and some ground truth information is necessary for the algorithms to learn.

In Chapter 11, we build upon the Zero velocity UPdaTe (ZUPT), which is a way to correct the estimates from the Inertial Navigation System (INS) when a wheeled vehicle stops. Indeed, the INS cannot measure directly the vehicle has stopped, and hence benefits from this side information (indeed, when a vehicle stops its trajectory is then in part known and the corresponding estimates may be corrected). In commercial INS this information when available always comes from additional wheel speed sensors, which are connected to the INS. In our work, we leverage deep learning to make the INS detect stops based only on the inertial sensors, hence relaxing the need for additional sensors besides inertial sensors. Evaluations on a publicly available car dataset demonstrates indeed that the proposed scheme may achieve a final precision of 20 m for a 21 km long trajectory of a vehicle driving for over an hour, equipped with an IMU of moderate precision (the gyro drift rate is 10 deg/h).

In Chapter 12 we focus on the estimation of the orientation of the vehicle only, namely its attitude. We leverage deep learning to refine the modelling of the measurements emanating from the inertial sensors, and to calibrate them. The algorithm outperforms the state-of-the-art.

Chapter 13 presents a few additional results in the same vein. Finally, in Appendix, we provide the reader with a few tips for Kalman filtering enhanced by deep learning. This chapter is a feedback to the community of our experience with this subject, developed throughout this thesis.

Part I

Unscented Kalman Filtering on Manifold & Lie Groups

CHAPTER 2

Introduction to Part I

This chapter introduces preliminary mathematical notions about Kalman filtering, manifolds, Lie groups, along with the notation used throughout this thesis.

In this chapter, the mathematical problem is based on models (regarding state's dynamics, measurements, and uncertainty) that are considered as given and wholly valid, and we discuss state estimation in this context[1].

1 1 The Problem of Bayesian Filtering and the Kalman Filter

Consider a classical discrete linear system in $\mathbb{R}^p$:

$$\mathbf{x}_{n+1} = \mathbf{F}_n\mathbf{x}_n + \mathbf{u}_n + \mathbf{G}_n\mathbf{w}_n, \tag{2.1}$$

$$\mathbf{y}_{n+1} = \mathbf{H}_{n+1}\mathbf{x}_{n+1} + \mathbf{n}_{n+1}, \tag{2.2}$$

where $\mathbf{x}_n \in \mathbb{R}^p$ is the state of the system at time n, $\mathbf{u}_n$ a given input vector, $\mathbf{y}_n$ a measurement observation, and $\mathbf{w}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_n)$, $\mathbf{n}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{N}_n)$ are independent Gaussian noises polluting the dynamics and the observation at step n. $\mathbf{F}_n$, $\mathbf{G}_n$ and $\mathbf{H}_n$ are the matrices defining the dynamics of the system and the function of the system observed through $\mathbf{y}_n$.

Let $\mathbf{x}_0 \sim \mathcal{N}(\hat{\mathbf{x}}_0, \mathbf{P}_0)$ be a Gaussian prior on the state at time $n = 0$. The problem of Bayesian filtering we address is to find the best estimates $\hat{\mathbf{x}}_n$ conditionally to initial state and past measurements $\mathbf{y}_1, \dots, \mathbf{y}_n$, i.e. finding $\hat{\mathbf{x}}_n$ that minimizes $\mathcal{E}\left[\|\mathbf{x}_n - \hat{\mathbf{x}}_n\|_2 | \mathbf{y}_1, \dots, \mathbf{y}_n\right]$.

When the system is linear, the Kalman filter is the best Bayesian filter, as the distribution of $\mathbf{x}_n$, given the propagation matrices $\mathbf{F}_n$ and the observations $\mathbf{y}_n$ stays Gaussian, thanks to the following two standard properties:

- if $\mathbf{x} \sim \mathcal{N}(\hat{\mathbf{x}}, \mathbf{P})$, $\mathbf{w} \sim \mathcal{N}(\hat{\mathbf{w}}, \mathbf{Q})$, and $\mathbf{F}$, $\mathbf{G}$, $\mathbf{u}$ are given matrices and vector, then $\mathbf{F}\mathbf{x} + \mathbf{u} + \mathbf{G}\mathbf{w}$ is a Gaussian, of mean $\mathbf{F}\hat{\mathbf{x}} + \mathbf{u} + \mathbf{G}\hat{\mathbf{w}}$ and covariance $\mathbf{F}\mathbf{P}\mathbf{F}^T + \mathbf{G}\mathbf{Q}\mathbf{G}^T$;

- if $\mathbf{x} \sim \mathcal{N}(\hat{\mathbf{x}}, \mathbf{P})$, $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \mathbf{N})$, and $\mathbf{H}$ is a given matrix, then $(\mathbf{x}, \mathbf{H}\mathbf{x} + \mathbf{n})$ forms a Gaussian vector. In turn, $\mathcal{E}[\mathbf{x}|\mathbf{H}\mathbf{x} + \mathbf{n}]$ is also a Gaussian, whose mean and covariance are given by the conditioning formulas, which lead to the Kalman filter equations, defined thereafter.

[1] it is only in the second and third parts of this thesis that we will question the mathematical models used for state estimation, and see how we can use machine learning to refine them in the face of collected data.

Thus, given a Gaussian prior, the Kalman filter defines an estimate $\hat{\mathbf{x}}_n$ at each step n through the following initialization and two step sequence:

$$\text{Initialization } \hat{\mathbf{x}}_{0|0} \sim \mathcal{N}(\hat{\mathbf{x}}_0, \mathbf{P}_0) \tag{2.3}$$

$$\text{Propagation} \begin{cases} \hat{\mathbf{x}}_{n+1|n} = \mathbf{F}_n \hat{\mathbf{x}}_{n|n} + \mathbf{u}_n \\ \mathbf{P}_{n+1|n} = \mathbf{F}_n \mathbf{P}_n \mathbf{F}_n^T + \mathbf{G}_n \mathbf{Q}_n \mathbf{G}_n^T \end{cases} \tag{2.4}$$

$$\text{Update} \begin{cases} \mathbf{S} = \mathbf{H}_{n+1} \mathbf{P}_{n+1|n} \mathbf{H}_{n+1}^T + \mathbf{N}_{n+1} \\ \mathbf{K} = \mathbf{P}_{n+1|n} \mathbf{H}_{n+1}^T \mathbf{S}^{-1} \\ \hat{\mathbf{x}}_{n+1|n+1} = \hat{\mathbf{x}}_{n+1|n} + \mathbf{K}(\mathbf{y}_{n+1} - \mathbf{H}_{n+1} \hat{\mathbf{x}}_{n+1|n}) \\ \mathbf{P}_{n+1|n+1} = (\mathbf{I} - \mathbf{K}\mathbf{H}_{n+1}) \mathbf{P}_{n+1|n} \end{cases} \tag{2.5}$$

The linear Kalman filter exhibits several optimality properties, it is in particular an unbiased estimator with minimum variance. These properties mostly come from two of its features: the particular Kalman gain $\mathbf{K}$ used, and the fact that its error is autonomous: it does not depend on the state estimates $\hat{\mathbf{x}}_{n|n}$ nor the observations $\mathbf{y}_n$. Indeed, let the error of the system be

$$\mathbf{e}_n = \mathbf{x}_n - \hat{\mathbf{x}}_n. \tag{2.6}$$

Then it satisfies

$$\mathbf{e}_{n+1|n} = \mathbf{F}_n \mathbf{e}_{n|n} + \mathbf{G}_n \mathbf{w}_n \tag{2.7}$$

$$\mathbf{e}_{n+1|n+1} = \mathbf{e}_{n+1|n} - \mathbf{K}\mathbf{H}_{n+1} \mathbf{e}_{n+1|n}. \tag{2.8}$$

Therefore, the error evolves solely according to the noises and their statistical properties, which are supposed to be known in this context. In particular, this also holds for its associated covariance $\mathbf{P}_n$. This autonomy property is crucial, as will be pointed out in invariant filtering in Section 7.

2 2 Nonlinear Filtering: the Extended Kalman Filter (EKF)

The original Kalman filter was designed for linear systems with linear observations. To handle non-linear dynamics or observations, an extension of the algorithm was proposed, called the Extended Kalman Filter (EKF). Consider the following non-linear system

$$\mathbf{x}_{n+1} = f(\mathbf{x}_n, \mathbf{u}_n, \mathbf{w}_n), \tag{2.9}$$

$$\mathbf{y}_n = h(\mathbf{x}_n) + \mathbf{n}_n. \tag{2.10}$$

Then the EKF follows similar steps as the linear version, up to the following differences:

- the estimate is propagated as $\hat{\mathbf{x}}_{n+1|n} = f(\hat{\mathbf{x}}_{n|n}, \mathbf{u}_n, \mathbf{0})$;
- the update writes $\hat{\mathbf{x}}_{n+1|n+1} = \hat{\mathbf{x}}_{n+1|n} + \mathbf{K}(\mathbf{y}_{n+1} - h(\hat{\mathbf{x}}_{n+1|n}))$;
- $\mathbf{F}_n$, $\mathbf{G}_n$, and $\mathbf{H}_n$ in (2.1) and (2.2) are replaced by $\hat{\mathbf{F}}_n$, $\hat{\mathbf{G}}_n$ and $\hat{\mathbf{H}}_n$, the Jacobians of $f(\cdot)$ and $h(\cdot)$ computed at the estimates $\hat{\mathbf{x}}_{n|n}$ and $\hat{\mathbf{x}}_{n+1|n}$ respectively.

This last point is crucial, as it means that the estimated covariance depends on the current estimate. That, in addition to the fact that the EKF outputs only an approximation of the probability $\mathcal{P}(\mathbf{x}_n | \hat{\mathbf{x}}, \mathbf{y}_1, \ldots, \mathbf{y}_n)$ which is not a Gaussian anymore, is well-known to make the EKF lose the optimality properties of the linear case.

The EKF is an *error state filter*, where the covariance $\mathbf{P}_n$ represents the uncertainty of the error (2.6), $\mathbf{e}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{P}_n)$. By the way, one compute the Jacobian after first-order approximation of the error, which is obtained for propagation, when noise is turned off, as

$$\begin{aligned}
\mathbf{e}_{n+1|n} &= \mathbf{x}_{n+1} - \hat{\mathbf{x}}_{n+1|n} && (2.11)\\
&= f(\mathbf{x}_n, \mathbf{u}_n, \mathbf{0}) - f(\hat{\mathbf{x}}_{n|n}, \mathbf{u}_n, \mathbf{0})\\
&= f(\hat{\mathbf{x}}_{n|n} + \mathbf{e}_{n|n}, \mathbf{u}_n, \mathbf{0}) - f(\hat{\mathbf{x}}_{n|n}, \mathbf{u}_n, \mathbf{0})\\
&= \cancel{f(\hat{\mathbf{x}}_{n|n}, \mathbf{u}_n, \mathbf{0})} + \frac{\partial f}{\partial \mathbf{x}}|_{\hat{\mathbf{x}}_{n|n}} \mathbf{e}_{n|n} - \cancel{f(\hat{\mathbf{x}}_{n|n}, \mathbf{u}_n, \mathbf{0})} + o(\mathbf{e}_{n|n})\\
&= \frac{\partial f}{\partial \mathbf{x}}|_{\hat{\mathbf{x}}_{n|n}} \mathbf{e}_{n|n} + o(\mathbf{e}_{n|n}), && (2.12)
\end{aligned}$$

where $f(\hat{\mathbf{x}}_{n|n} + \mathbf{e}_{n|n}, \mathbf{u}_n, \mathbf{0}) = f(\hat{\mathbf{x}}_{n|n}, \mathbf{u}_n, \mathbf{0}) + \frac{\partial f}{\partial \mathbf{x}}|_{\hat{\mathbf{x}}_{n|n}} \mathbf{e}_{n|n} + o(\mathbf{e}_{n|n})$. Thus it appears that $\hat{\mathbf{F}}_n = \frac{\partial f}{\partial \mathbf{x}}|_{\hat{\mathbf{x}}_{n|n}}$ albeit $\hat{\mathbf{F}}_n$ is not defined as the first-order Taylor expansion of $f(\cdot)$ and change if, e.g., $\mathbf{e}_n = \sqrt{\|\mathbf{x}_n - \hat{\mathbf{x}}_n\|}$. The same holds for $\hat{\mathbf{G}}_n$ and $\hat{\mathbf{H}}_n$.

3 3 Nonlinear Filtering: the Unscented Kalman Filter (UKF)

The Unscented Kalman Filter (UKF) [1,2] is an alternative to the EKF for handling non-linear dynamics or observations (2.9)-(2.10) that uses a deterministic sampling technique known as the unscented transform to pick a minimal set of sample points (called sigma points) around the mean. The sigma points are then propagated through the nonlinear functions, from which a new state estimate and error state covariance are inferred. UKF is arguably more accurate than EKF, and to some extent simpler to implement for the practitioner not familiar with Kalman filter theory (no analytical Jacobian are required for an UKF).

It exists several variant of UKFs [3] and we consider here the one that provides a direct analogy with the EKF [4,5], where the unscented transform numerically computes statistical Jacobians. Thus the UKF follows similar steps as the EKF version, up to the following differences:

- $\mathbf{F}_n$, $\mathbf{G}_n$, and $\mathbf{H}_n$ in (2.1) and (2.2) are replaced by $\hat{\mathbf{F}}_n$, $\hat{\mathbf{G}}_n$ and $\hat{\mathbf{H}}_n$, which are the *statistical* Jacobians of $f(\cdot)$ and $h(\cdot)$. As for the EKF, they are computed at the estimates $\hat{\mathbf{x}}_{n|n}$ and $\hat{\mathbf{x}}_{n+1|n}$ respectively and depend on the state estimates.

Jacobians are inferred following Algorithm 1 and Algorithm 2, where sigma point parameters are computed in `set_weights`$(\cdot)$, see Algorithm 3, where the parameter α is chosen by the practitioner, typically $\alpha = 10^{-3}$, and controls the spread of the sigma points.

This technique has two main advantages over EKF and one caveat: it removes the requirement to explicitly calculate Jacobians, performs generally better than the EKF, but its computational complexity is cubic in the size of the state vector (the computational complexity of EKF is quadratic).

4 4 Introduction to Filtering on Manifolds

We now consider the situation where the state lives on a manifold $\mathcal{M}$, see [6] for a introduction to manifold with optimization perspective, and [7] which is a reference from the robotics perspective. We now provide a rather "loose" definition of a manifold.

Algorithm 1: UKF computation of $\hat{\mathbf{F}}_n$ and $\hat{\mathbf{G}}_n$

Input: $\hat{\mathbf{x}}_{n+1|n}, \hat{\mathbf{x}}_{n|n}, \mathbf{P}_{n|n}, \mathbf{u}_n, \mathbf{G}_n, \alpha$;
// Sigma-point parameters
$\lambda, \{w_j\}_{j=0,\dots,2d} = \texttt{set_weights}(\dim(\mathbf{P}_{n|n}), \alpha)$;
$\xi_j = \mathrm{col}(\sqrt{(\lambda+d)\mathbf{P}_{n|n}})_j,\ j = 1,\dots,d$
$\xi_j = -\mathrm{col}(\sqrt{(\lambda+d)\mathbf{P}_{n|n}})_{j-d},\ j = d+1,\dots,2d$ // sigma point errors
// add sigma points on state and propagate them
$\hat{\mathbf{x}}_n^j = \hat{\mathbf{x}}_{n|n} + \xi_j, j = 1,\dots,2d$;
$\hat{\mathbf{x}}_{n+1}^j = f(\hat{\mathbf{x}}_n^j, \mathbf{u}_n, \mathbf{0}), j = 1,\dots,2d$;
// Compute cross-covariance
$\Sigma = \sum_{j=1}^{2d} w_j(\hat{\mathbf{x}}_{n+1|n} - \hat{\mathbf{x}}_{n+1}^j)(\hat{\mathbf{x}}_{n|n} - \hat{\mathbf{x}}_n^j)^T$;
$\hat{\mathbf{F}}_n = \Sigma \mathbf{P}_{n|n}^{-1}$ // Infer Jacobian
// Proceed similarly for noise
$\lambda, \{w_j\}_{j=0,\dots,2d} = \texttt{set_weights}(\dim(\mathbf{Q}), \alpha)$;
$\mathbf{w}^j = \mathrm{col}(\sqrt{(\lambda+d)\mathbf{Q}})_j,\ j = 1,\dots,d$
$\mathbf{w}^j = -\mathrm{col}(\sqrt{(\lambda+d)\mathbf{Q}})_{j-d},\ j = d+1,\dots,2d$;
$\hat{\mathbf{x}}_{n+1}^j = f(\hat{\mathbf{x}}_{n|n}, \mathbf{u}_n, \mathbf{w}^j), j = 1,\dots,2d$;
$\Sigma = \sum_{j=1}^{2d} w_j(\hat{\mathbf{x}}_{n+1|n} - \hat{\mathbf{x}}_{n+1}^j)(\mathbf{w}^j)^T$;
$\hat{\mathbf{G}}_n = \Sigma \mathbf{Q}^{-1}$ // Infer Jacobian
Output: $\hat{\mathbf{F}}_n$, $\hat{\mathbf{G}}_n$

Definition 1 (manifold)**.** *A manifold $\mathcal{M}$ is a set whose subsets are identified through charts (injective mappings) with subsets of $\mathbb{R}^n$.*

Consider a practical example where the state lives on the 2-sphere manifold, modeling e.g., a spherical pendulum [8], see Figure 2.1 where

$$\mathbf{x}_n \in \mathbb{S}_2 := \{\mathbf{x} \in \mathbb{R}^3 \mid \|\mathbf{x}\| = 1\}. \tag{2.13}$$

Assume we have an estimate $\hat{\mathbf{x}}_{n+1|n} \in \mathbb{S}_2$ of $\mathbf{x}_{n+1}$ whose covariance is $\mathbf{P}_{n|n-1}$. Assume now we have a linear observation of the state, e.g. by viewing the pendulum with a monocular camera in a plan, i.e. $\mathbf{y}_{n+1} = \mathbf{H}\mathbf{x}_{n+1} + \mathbf{n}$ with $\mathbf{n}$ a Gaussian noise. Applying Kalman filter update (2.5), we obtain

$$\hat{\mathbf{x}}_{n+1|n+1} = \hat{\mathbf{x}}_{n+1|n} + \mathbf{K}(\mathbf{y}_{n+1} - \mathbf{H}\hat{\mathbf{x}}_{n+1|n}) \notin \mathbb{S}_2 \tag{2.14}$$

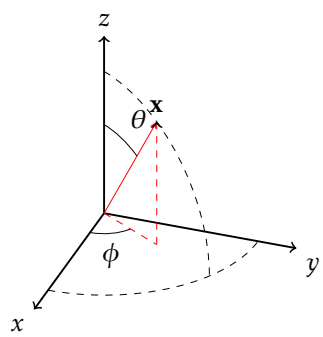


Figure 2.1: The example of a spherical pendulum, where $\mathbf{x} = [x, y, z]^T \in \mathbb{S}_2$.

Algorithm 2: UKF computation of $\hat{\mathbf{H}}_n$

Input: $\hat{\mathbf{y}}_{n+1}, \hat{\mathbf{x}}_{n+1|n}, \mathbf{P}_{n+1|n}, \alpha$;
// Sigma-point parameters
$\lambda, \{w_j\}_{j=0,\dots,2d} = \texttt{set_weights}(\dim(\mathbf{P}_{n+1|n}), \alpha)$;
$\xi_j = \mathrm{col}(\sqrt{(\lambda+d)\mathbf{P}_{n+1|n}})_j,\ j = 1,\dots,d$
$\xi_j = -\mathrm{col}(\sqrt{(\lambda+d)\mathbf{P}_{n+1|n}})_{j-d},\ j = d+1,\dots,2d$ // sigma point errors
// add sigma points on state and propagate them
$\hat{\mathbf{x}}^j_{n+1} = \hat{\mathbf{x}}_{n+1|n} + \xi_j, j = 1,\dots,2d$;
$\hat{\mathbf{y}}^j_{n+1} = h(\hat{\mathbf{x}}^j_{n+1}), j = 1,\dots,2d$;
// Compute cross-covariance
$\Sigma = \sum_{j=1}^{2d} w_j(\hat{\mathbf{y}}_{n+1} - \hat{\mathbf{y}}^j_{n+1})(\hat{\mathbf{x}}_{n+1|n} - \hat{\mathbf{x}}^j_{n+1})^T$;
$\hat{\mathbf{H}}_n = \Sigma \mathbf{P}^{-1}_{n+1|n}$ // Infer Jacobian
Output: $\hat{\mathbf{H}}_n$

Algorithm 3: `set_weights`$(\cdot)$ function used for computing UKF parameters.

Input: d, α;
$\lambda = (\alpha^2 - 1)d$;
$w_0 = \lambda/(\lambda + d) + 3 - \alpha^2$;
$w_j = 1/(2(d+\lambda)), j = 1,\dots,2d$;
Output: λ, $\{w_j\}_{j=0,\dots,2d}$;

such that the estimated state $\hat{\mathbf{x}}_{n+1|n+1}$ as no reason to stay on the 2-sphere. The Kalman filter indeed assumes $\mathbf{x}_{n+1}$ lives in a vector space $\mathbb{R}^p$. However, it exists many problems in robotics involving rotation, pose and sphere which are not vector space but manifold. Two problems directly appears here:

- the error-state covariance $\mathbf{P}_n$ has dimension three although the state has only two degree of freedom. In addition, $\mathbf{P}_n$ is supposed to represent a statistical dispersion over the manifold, and this very concept is even not clear;
- one would hope that $\mathbf{e}_n$ and $\hat{\mathbf{x}}_n$ remains at all times in the manifold, however adding elements of a manifold does not make sense as

$$\mathbf{x}^1, \mathbf{x}^2 \in \mathcal{M} \not\Rightarrow \mathbf{x}^1 + \mathbf{x}^2 \in \mathcal{M}. \tag{2.15}$$

  The addition operator + is just not defined on a manifold, and $\hat{\mathbf{x}}_n$ needs to be constrained on the manifold.

An instinctive solution consists in changing the coordinate of the state space and writing the systems in term of spherical coordinates (ϕ, θ), see Figure 2.1, as

$$\phi = \arctan(y/x) \tag{2.16}$$
$$\theta = \arccos(z). \tag{2.17}$$

This is especially cumbersome as one need to define a new filter where the observation becomes nonlinear, while now state estimates have singularities at the north and south poles (for $x = 0$).

This raises important questions when designing a filter on a manifold $\mathcal{M}$:

- How may we design an error $\mathbf{e}_n$ that handles the manifold constraints? Is it only related to the coordinate system we used? How may we avoid singularity issues?

- How may we perform state update and compute $\hat{\mathbf{x}}_{n+1|n+1}$? How to compute Jacobians?
- Solving those questions naturally leads to different ways to define a state error. So, which error should be used for a given problem?

Manifolds are highly general objects, and one may search to answer these questions in the more restrictive case when the manifold is a matrix Lie group (and hence comes with additional structure).

5 5 Matrix Lie Groups

In this section we recall the definitions and basic properties of matrix Lie groups and Lie algebra. A matrix Lie group $G \in \mathbb{R}^{n\times n}$ is a set of square invertible matrices that is a group, i.e., the following properties hold:

$$\mathbf{I}_n \in G; \quad \forall \chi \in G, \chi^{-1} \in G; \quad \forall \chi^1, \chi^2 \in G, \chi^1\chi^2 \in G. \tag{2.18}$$

Matrix Lie groups are particular examples of Lie groups, which are sets being

- groups, i.e., have an inner product, an inverse function, and a neutral element (e.g., matrix product, matrix inverse and identity for matrix Lie groups);
- differential manifolds, i.e., one can derive at each point on them;
- and such that the inner product and the inverse function are smooth for the manifold structure.

Although Lie group theory can get quite involved, all the considered sets in this thesis are matrix Lie groups, therefore we will restrict to them.

Locally about the identity matrix $\mathbf{I}_n$, the group G can be identified with a vector space $\mathbb{R}^d$ using the matrix exponential map $\exp_m(\cdot)$, where $d = \dim(G)$. Indeed, to any $\xi \in \mathbb{R}^d$ one can associate a matrix $\xi^\wedge$ of the tangent space of G at $\mathbf{I}_n$, called the Lie algebra $\mathfrak{g}$, and we call $(\cdot)^\wedge : \mathbb{R}^d \to \mathfrak{g}$ the wedge operator. We then define the exponential map $\exp : \mathbb{R}^d \to G$ for Lie groups as

$$\exp(\xi) = \exp_m(\xi^\wedge) \tag{2.19}$$

Locally, around zero, it is a bijection, and one can define the Lie logarithm map $\log : G \to \mathbb{R}^d$ as its inverse $\log(\exp(\xi)) = \xi$.

Two specific matrix Lie groups are heavily used in robotics: the special orthogonal group, denoted $SO(3)$, which can represent rotations, and the special Euclidean group, $SE(3)$, which can represent poses. Additionally, in [9,10], the author noticed there is a natural Lie group structure underlying the 3D SLAM problem, called $SE_p(3)$, which resolves some consistency issues of the conventional EKF based SLAM, see Chapter 5.

5.1 5.1 The Special Orthogonal Group $SO(3)$

The special orthogonal group, representing rotations, is the set of valid rotation matrices

$$SO(3) = \left\{\mathbf{R} \in \mathbb{R}^{3\times 3} | \mathbf{R}\mathbf{R}^T = \mathbf{I}, \det(\mathbf{R}) = 1\right\}. \tag{2.20}$$

The $\mathbf{R}\mathbf{R}^T = \mathbf{I}$ orthogonality condition impose six constraints on the nine-parameter rotation matrix, thereby reducing the number of degrees of freedom to three. $SO(3)$ is clearly not a vector space: it is not closed under addition and the zero matrix does live in $SO(3)$.

The inverse, wedge operator and exponential maps are given as

$$\mathbf{R}^{-1} = \mathbf{R}^T, \tag{2.21}$$

$$\xi^\wedge = \begin{bmatrix} 0 & -\xi_3 & \xi_2 \\ \xi_3 & 0 & -\xi_1 \\ -\xi & \xi_1 & 0 \end{bmatrix}, \tag{2.22}$$

$$\exp(\xi) = \mathbf{I}_3 + \frac{\sin(\|\xi\|)}{\|\xi\|}\xi^\wedge + 2\frac{\sin(\|\xi\|/2)^2}{\|\xi\|^2}(\xi^\wedge)^2. \tag{2.23}$$

We note here the wedge operator corresponds to the skew-symmetric operator $(\cdot)_\times$. The inverse $\log(\cdot)$ is the vector associated to the skew-symmetric matrix defined as

$$\log(\mathbf{R}) = \frac{\theta}{2\sin(\theta)}(\mathbf{R} - \mathbf{R}^T), \tag{2.24}$$

where $\theta = \arccos\left(\frac{\mathrm{trace}(\mathbf{R})-1}{2}\right)$.

5.2 5.2 The Special Euclidean Group $SE(3)$

The special Euclidean group, representing poses (i.e., translation and rotation), is the set of valid transformation matrices

$$SE(3) := \left\{ \mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{p} \\ \mathbf{0} & 1 \end{bmatrix} \in \mathbb{R}^{4\times 4} | \mathbf{R} \in SO(3), \mathbf{p} \in \mathbb{R}^3 \right\}, \tag{2.25}$$

and are thus represented by a matrix $\mathbf{T}$ (a.k.a. homogeneous coordinates), where $\mathbf{R}$ denotes a rotation matrix and $\mathbf{p}$ a translation.

The inverse, wedge operator and exponential maps are given as

$$\mathbf{T}^{-1} = \begin{bmatrix} \mathbf{R}^T & -\mathbf{R}^T\mathbf{p} \\ \mathbf{0} & 1 \end{bmatrix}, \tag{2.26}$$

$$\xi^\wedge = \begin{bmatrix} \phi_\times & \rho \\ \mathbf{0} & 0 \end{bmatrix}, \tag{2.27}$$

$$\exp(\xi) = \mathbf{I}_4 + \xi^\wedge + \frac{1-\cos(\theta)}{\theta^2}(\xi^\wedge)^2 + \frac{\theta - \sin(\theta)}{\theta^3}(\xi^\wedge)^3, \tag{2.28}$$

where $\xi = [\phi^T, \rho^T]^T$ and $\theta = \|\phi\|$. The inverse $\log(\cdot)$ involves the logarithm of $SO(3)$, $\log_{SO(3)}(\cdot)$, and is defined as

$$\log(\mathbf{T}) = \begin{bmatrix} \log_{SO(3)}(\mathbf{R}) \\ (\mathbf{I}_3 - \frac{1}{2}\phi_\times + \frac{1}{\theta^2}(1 - \frac{A}{2B})(\phi_\times)^2)\rho \end{bmatrix}, \tag{2.29}$$

where $A = \sin(\theta)/\theta$ and $B = (1-\cos(\theta))/\theta^2$. For more information about $SE(3)$ and its use in state estimation for robotics see the recent monographs [11,12].

5.3 5.3 The Group $SE_p(3)$

The group $SE_p(3)$ is an extension of the group $SE(3)$ introduced in [9,10] essentially for SLAM, and is defined as follows

$$SE_p(3) := \left\{ \mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{p}_1 & \cdots & \mathbf{p}_p \\ \mathbf{0} & & \mathbf{I}_p & \end{bmatrix} \in \mathbb{R}^{3+p\times 3+p} | \mathbf{R} \in SO(3), \mathbf{p}_1, \ldots, \mathbf{p}_p \in \mathbb{R}^3 \right\}, \tag{2.30}$$

and are thus represented by a matrix $\mathbf{T}$, where $\mathbf{R}$ denotes a rotation matrix and $\mathbf{p}_1, \ldots, \mathbf{p}_p$ are translations. The dimension of the group, and thus of the Lie algebra, is $3+3p$.

The inverse, wedge operator and exponential maps are given as

$$\mathbf{T}^{-1} = \begin{bmatrix} \mathbf{R}^T & -\mathbf{R}^T\mathbf{p}_1 & \cdots & -\mathbf{R}^T\mathbf{p}_p \\ \mathbf{0} & & \mathbf{I}_p & \end{bmatrix}, \tag{2.31}$$

$$\xi^\wedge = \begin{bmatrix} \phi_\times & \rho_1 & \cdots & \rho_p \\ \mathbf{0} & & \mathbf{0} & \end{bmatrix}, \tag{2.32}$$

$$\exp(\xi) = \mathbf{I}_{3+p} + \xi^\wedge + \frac{1-\cos(\theta)}{\theta^2}(\xi^\wedge)^2 + \frac{\theta - \sin(\theta)}{\theta^3}(\xi^\wedge)^3, \tag{2.33}$$

where $\xi = [\phi^T, \rho_1^T, \cdots, \rho_p^T]^T$ and $\theta = \|\phi\|$. The inverse $\log(\cdot)$ is defined as

$$\log(\mathbf{T}) = \begin{bmatrix} \log_{SO(3)}(\mathbf{R}) \\ \mathbf{J}\rho_1 \\ \vdots \\ \mathbf{J}\rho_p \end{bmatrix}, \tag{2.34}$$

where $A = \sin(\theta)/\theta$, $B = (1-\cos(\theta))/\theta^2$, and $\mathbf{J} = (\mathbf{I}_3 - \frac{1}{2}\phi_\times + \frac{1}{\theta^2}(1 - \frac{A}{2B})(\phi_\times)^2)$.

We leverage this group in Chapter 5 and Chapter 6, where we embed the robot orientation $\mathbf{R} \in SO(3)$, velocity $\mathbf{v} \in \mathbb{R}^3$ and position $\mathbf{p} \in \mathbb{R}^3$ along with the position of landmarks $\mathbf{p}_i \in \mathbb{R}^3$ in a matrix $\chi \in SE_p(3)$ defined as

$$\chi = \begin{bmatrix} \mathbf{R} & \mathbf{v} & \mathbf{p} & \mathbf{p}_1 & \cdots & \mathbf{p}_{p-2} \\ & \mathbf{0} & & & \mathbf{I}_p & \end{bmatrix}. \tag{2.35}$$

6 6 Nonlinear & Invariant Observers on Lie Groups

There has been a huge body of research devoted to nonlinear and invariant observers and filters on Lie groups over the past decade, especially for attitude estimation, where invariant observers are deterministic observers meant to be appropriate to dynamical systems possessing symmetries [13].

In the early 2000s, there were essentially two streams of research that bolstered the development of observers on Lie groups. The first was initiated by [14] and seeks to design nonlinear observers that share the symmetries of the original system. The theory was formalized and developed in [15] and applied to estimation on Lie groups in [16]. At the same time, the complementary filter on $SO(3)$ for attitude estimation was introduced in [17]. This filter makes extensive use of left-invariant errors on $SO(3)$ and the autonomy properties of the error equation. Owing to its simplicity and global convergence guarantees, it has become a renowned attitude estimator and has proved useful for micro aerial vehicle control. [18–24] study observers that are akin to the complementary filter on $SO(3)$ for attitude estimation, and similar ideas have been applied to pose estimation in [25] and homography in [26]. Invariant observer theory were

then developed in e.g. [20,27–29] for attitude estimation, [30,31] for navigation, [32] for gradient-like observers, and applied in [33] for the design of invariant controllers.

The idea to use similar techniques for noisy (instead of deterministic) systems on Lie groups lead to the more recent theory of invariant Kalman filtering.

7 7 Invariant Extended Kalman Filter

The first introduction of invariant Kalman filtering is presented to our best knowledge in [34] that was then applied for attitude estimation in [16]. These results showed that, when considering distributions inspired from the Gaussian but specifically designed to take symmetries into account, theoretical results similar to those of the linear case could be derived, and have been applied in [35].

In turn, this showed that there was a kind of filters which could efficiently encode the past of the system, and carry out reliable estimation of local navigation in the long run [10,36]. This framework, based on the theory of Lie groups, is known as invariant filtering, and especially the Invariant Extended Kalman Filter (IEKF). Its theoretical and practical advantages were brought to light by the Ph.D. thesis of Axel Barrau [37], in particular with high-grade IMUs. Its extension to $H\infty$ norm [38] and smoothing has been recently proposed in [39,40]. Generally speaking, the IEKF is close to the approach of [18], however these did not leverage its links with group affine dynamics and group actions, and therefore autonomous errors. The IEKF can be seen as an extension of the Multiplicative EKF [41] for more general state spaces.

The IEKF [42,43] is a nonlinear Kalman filter variant devised to estimate systems where the state live a on matrix Lie groups, e.g. $SO(3)$ for attitude estimation, where the state is a rotation matrix. We note here the state $\chi_n \in G$ to make clear the distinction with a state living in Euclidean space.

Its goal is to track the mean and covariance of a distribution,

$$\chi = \varphi(\hat{\chi}, \xi), \ \xi \sim \mathcal{N}(\mathbf{0}, \mathbf{P}), \tag{2.36}$$

where $\varphi : \mathcal{M} \times \mathbb{R}^d \to \mathcal{M}$ is a smooth function, which is defined Gaussian in the Lie algebra $\mathfrak{g}$ of the state space (see Chapter 4). The IEKF is designed to be applied to specific systems: as the Kalman filter is adapted to linear systems, the IEKF is well-suited for the so called linear observed systems on groups [44]. Thus, the IEKF make advantages of:

- the state-space structure G. Notably, the estimated state $\hat{\chi}_n$ is guarantee to remain in the group without any ad hoc reproduction;
- the system equations $f(\cdot)$ and $h(\cdot)$, such that the dynamics of the error is autonomous, i.e. the error does not depend on state estimate. This is due the computation of the Jacobian, e.g. $\mathbf{F}_n$, which does not depend on $\hat{\chi}_n$ and avoid thus some linearization error.

7.1 7.1 Considered State-Space Systems

Let us define two types of discrete linear systems in a matrix Lie group G as:

Left invariant system

$$\chi_{n+1} = f(\chi_n, \mathbf{u}_n)\exp(\mathbf{w}_n) \tag{2.37}$$

$$\mathbf{y}_{n+1} = \chi_{n+1}\mathbf{d} + \mathbf{n}_n \tag{2.38}$$

Right invariant system

$$\chi_{n+1} = f(\chi_n, \mathbf{u}_n)\exp(\mathbf{w}_n) \tag{2.39}$$

$$\mathbf{y}_{n+1} = \chi_{n+1}^{-1}\mathbf{d} + \mathbf{n}_n \tag{2.40}$$

In these systems, $\mathbf{w}_n$ and $\mathbf{n}_n$ are noise similarly defined as in Section 2, the propagation function $f(\cdot)$ respects the group affine dynamics property [44]

$$\forall \mathbf{u}, \mathbf{a}, \mathbf{b} \quad f(\mathbf{a}, \mathbf{u}) f(\mathbf{I}, \mathbf{u})^{-1} f(\mathbf{b}, \mathbf{u}), \tag{2.41}$$

and with $\mathbf{d}$ a vector. The two observation models are respectively left and right equivariant, each associated to a variant of the IEKF: the left IEKF and the right IEKF.

7.2 7.2 Invariant Extended Kalman Filter Equations

The Left IEKF: the left IEKF estimates the state of the system through the same propagation and update sequence scheme of the of Kalman filter, where the state error relation is defined as

$$\chi_n = \hat{\chi}_n \exp(\xi_n), \tag{2.42}$$

where $\xi_n \sim \mathcal{N}(\mathbf{0}, \mathbf{P}_n)$ is the error and the $\exp(\cdot)$ in (2.44) is the exponential map of G that ensure $\hat{\chi}_n$ to stay in G. The *vector* error $\xi_n \in \mathbb{R}^{\dim G}$ extend definition of the error $\mathbf{e}_n$ defined in Section 2 for vector space, as the exponential map for $\mathbb{R}^p$ is the vector addition. Note the name *left* or left-invariant IEKF takes its root in (2.42): multiplying (2.42) by $\mathbf{U} \in G$ let the error ξ_n unchanged.

The filter equations are given as

$$\text{Propagation} \begin{cases} \hat{\chi}_{n+1|n} = f(\hat{\chi}_{n|n}, \mathbf{u}_n) \\ \mathbf{P}_{n+1|n} = \mathbf{F}_n \mathbf{P}_n \mathbf{F}_n^T + \mathbf{G}_n \mathbf{Q}_n \mathbf{G}_n^T \end{cases} \tag{2.43}$$

$$\text{Update} \begin{cases} \mathbf{S} = \mathbf{H}_{n+1} \mathbf{P}_{n+1|n} \mathbf{H}_{n+1}^T + \mathbf{N}_{n+1} \\ \mathbf{K} = \mathbf{P}_{n+1|n} \mathbf{H}_{n+1}^T \mathbf{S}^{-1} \\ \hat{\chi}_{n+1|n+1} = \hat{\chi}_{n+1|n} \exp\Big(\mathbf{K}(\hat{\chi}_{n+1}^{-1} (\mathbf{y}_{n+1} - \mathbf{d})\Big) \\ \mathbf{P}_{n+1|n+1} = (\mathbf{I} - \mathbf{K}\mathbf{H}_{n+1}) \mathbf{P}_{n+1|n} \end{cases}. \tag{2.44}$$

Let us first focus on the propagation step. To compute the Jacobian $\mathbf{F}_n$ and $\mathbf{G}_n$, the key idea underlying the EKF is apply, see Section 2, and we linearize the *error* system through a first-order Taylor expansion of the nonlinear functions $f(\cdot)$ at the estimated state. To compute $\mathbf{F}_n$ (and similarly $\mathbf{G}_n$), we write

$$\begin{aligned}
\exp(\xi_{n+1|n}) &= f(\hat{\chi}_{n|n}, \mathbf{u}_n)^{-1} f(\chi_{n|n}, \mathbf{u}_n) && (2.45)\\
&= f(\hat{\chi}_{n|n}, \mathbf{u}_n)^{-1} f(\hat{\chi}_{n|n} \exp(\xi_{n|n}), \mathbf{u}_n) && (2.46)\\
&= f(\hat{\chi}_{n|n}, \mathbf{u}_n)^{-1} f(\hat{\chi}_{n|n}[\mathbf{I} + \xi_{n|n}^\wedge], \mathbf{u}_n) + o(\|\xi_{n|n}\|) && (2.47)\\
&= \cancel{f(\hat{\chi}_{n|n}, \mathbf{u}_n)^{-1}} \cancel{f(\hat{\chi}_{n|n}, \mathbf{u}_n)} [\mathbf{I} + (\frac{\partial f}{\partial \chi}|_{\hat{\chi}_{n|n}, \mathbf{u}_n} \xi_{n|n})^\wedge] + o(\|\xi_{n|n}\|) && (2.48)\\
&= \mathbf{I} + (\frac{\partial f}{\partial \chi}|_{\hat{\chi}_{n|n}, \mathbf{u}_n} \xi_{n|n})^\wedge + o(\|\xi_{n|n}\|) && (2.49)\\
\mathbf{I} + \xi_{n+1|n}^\wedge &= \mathbf{I} + (\frac{\partial f}{\partial \chi}|_{\hat{\chi}_{n|n}, \mathbf{u}_n} \xi_{n|n})^\wedge + o(\|\xi_{n+1|n}\|, \|\xi_{n|n}\|) && (2.50)
\end{aligned}$$

where the error definition is used in (2.45)-(2.46) and the first order Baker-Campbell-Hausdorff (BCH) approximation $\exp(\xi) \simeq \mathbf{I} + \xi^\wedge$ is applied in (2.47) and (2.50). We thus identify $\mathbf{F}_n = \frac{\partial f}{\partial \chi}|_{\hat{\chi}_{n|n}, \mathbf{u}_n}$ such that

$$\xi_{n+1|n} = \mathbf{F}_n \xi_{n|n}, \tag{2.51}$$

where $\frac{\partial f}{\partial \mathcal{X}} = \frac{\partial f(\mathcal{X}\exp(\xi),\mathbf{u})}{\partial \xi}$ is defined by small perturbation acting on *right* of the state. When $f(\cdot)$ respects (2.41), $\mathbf{F}_n$ is independent of the state estimate.

The derivation of the update is simpler as we measure a vector $\mathbf{y}$. First, the update is based on the measurement rewritten as

$$\mathbf{z}_{n+1} = \hat{\mathcal{X}}_{n+1}^{-1}\mathbf{y}_{n+1} \tag{2.52}$$
$$= \exp(\xi_n)\mathbf{d} + \hat{\mathcal{X}}_{n+1}^{-1}\mathbf{n}_n. \tag{2.53}$$

and thus contains a modified innovation step. The Jacobian $\mathbf{H}_n$ is obtained by the following identification

$$\mathbf{z} - \hat{\mathbf{z}} = \mathbf{d} - \exp(\xi_{n+1|n})\mathbf{d} \tag{2.54}$$
$$= \mathbf{d} - [\mathbf{I} + \xi_{n+1|n}^{\wedge}]\mathbf{d} + o(\|\xi_{n+1|n}\|) \tag{2.55}$$
$$= \xi_{n+1|n}^{\wedge}\mathbf{d} + o(\|\xi_{n+1|n}\|) \tag{2.56}$$
$$= \mathbf{H}_n\xi_{n+1|n} + o(\|\xi_{n+1|n}), \tag{2.57}$$

and we see $\mathbf{H}_n$ is independent w.r.t. the state estimate.

The Right IEKF: the right IEKF differs from the left IEKF by defining the state error as

$$\mathcal{X}_n = \exp(\xi_n)\hat{\mathcal{X}}_n, \tag{2.58}$$

and rewriting the measurement as

$$\mathbf{z}_{n+1} = \hat{\mathcal{X}}_{n+1}\mathbf{y}_{n+1} \tag{2.59}$$
$$= \exp(\xi_n)\mathbf{d} + \hat{\mathcal{X}}_{n+1}\mathbf{n}_n, \tag{2.60}$$

where we have a different noise so now $\mathbf{N}_{n+1}$ depends on the state estimate. This leads to the right IEKF algorithm defined as

$$\text{Propagation}\begin{cases}\hat{\mathcal{X}}_{n+1|n} = f(\hat{\mathcal{X}}_{n|n},\mathbf{u}_n)\\ \mathbf{P}_{n+1|n} = \mathbf{F}_n\mathbf{P}_n\mathbf{F}_n^T + \mathbf{G}_n\mathbf{Q}_n\mathbf{G}_n^T\end{cases} \tag{2.61}$$

$$\text{Update}\begin{cases}\mathbf{S} = \mathbf{H}_{n+1}\mathbf{P}_{n+1|n}\mathbf{H}_{n+1}^T + \mathbf{N}_{n+1}\\ \mathbf{K} = \mathbf{P}_{n+1|n}\mathbf{H}_{n+1}^T\mathbf{S}^{-1}\\ \hat{\mathcal{X}}_{n+1|n+1} = \exp\left(\mathbf{K}(\hat{\mathcal{X}}_{n+1}\left(\mathbf{y}_{n+1} - \mathbf{d}\right)\right)\hat{\mathcal{X}}_{n+1|n}\\ \mathbf{P}_{n+1|n+1} = (\mathbf{I} - \mathbf{K}\mathbf{H}_{n+1})\mathbf{P}_{n+1|n}\end{cases} \tag{2.62}$$

The Jacobians are obtained similarly as for the left IEKF, and $\mathbf{H}_n$ is also independent w.r.t. state estimates.

CHAPTER 3

Exploiting Symmetries to Design EKFs with Consistency Properties for Navigation and SLAM

Part of the content of this chapter has been published in IEEE Sensors Journal.

Résumé 🇫🇷

Le filtre de Kalman étendu peut s'avérer inconsistent en présence d'inobservabilité sous un groupe de symétrie. Dans ce chapitre, nous construisons d'abord un EKF alternatif basé sur une erreur d'état non linéaire alternative. Cet EKF est intimement lié à la théorie de l'EKF invariant. Ensuite, sous une simple hypothèse de compatibilité entre l'erreur et le groupe de symétrie, nous prouvons que le modèle linéarisé de l'EKF alternatif capture automatiquement les directions non observables, et de nombreuses propriétés souhaitables du cas linéaire suivent alors directement. Cela fournit un nouveau résultat fondamental dans la théorie du filtrage. Nous appliquons la théorie à la fusion multi-capteurs pour la navigation, lorsque tous les capteurs sont liés au véhicule et n'ont pas accès à des mesures absolues, comme cela se produit généralement dans des environnements privés de GPS. Dans le contexte du SLAM, des simulations Monte-Carlo et des comparaisons avec l'OC-EKF, l'EKF et les algorithmes de lissage basés sur l'optimisation (iSAM) illustrent les résultats.

Chapter abstract

The extended Kalman filter may prove inconsistent in the presence of unobservability under a group of transformations. In this chapter we first build an alternative EKF based on an alternative nonlinear state error. This EKF is intimately related to the theory of the Invariant EKF. Then, under a simple compatibility assumption between the error and the transformation group, we prove the linearized model of the alternative EKF automatically captures the unobservable directions, and many desirable properties of the linear case then directly follow. This provides a novel fundamental result in filtering theory. We apply the theory to multi-sensor fusion for navigation, when all the sensors are attached to the vehicle and do not have access to absolute information, as typically occurs in GPS-denied environments. In the context of SLAM, Monte-Carlo runs and comparisons to OC-EKF, robocentric EKF, and optimization-based smoothing algorithms (iSAM) illustrate the results.

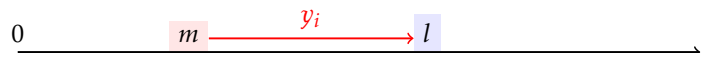


Figure 3.1: 1D toy SLAM problem. A robot at position m observes a landmark at position l with a range measurement y_i.

1 1 Introduction

We first start with a very basic example that explains the specificity of the theoretical estimation problem posed by SLAM. Then we proceed with a brief literature review about the subproblem we consider.

1.1 1.1 Introduction to SLAM Unobservability

As in SLAM we only have *relative measurements* between components of the state, the Kalman filter tends to correlate those components after each measurement. This correlation profoundly modifies the information that the filter possesses about the system, although it does not necessarily improve the marginal knowledge about each component (e.g., the localization of the robot which is what we are most interested in throughout this thesis). This is why it is not obvious for the reader unfamiliar with this problem to grasp what is at play in the theoretical SLAM problem and this is why we present the following simple example.

Consider a linear one-dimensional mobile robot at position m that observes *one* landmark, at position l, as represented in Figure 3.1. The robot is equipped with a device (a telemeter) that allows to measure the distance to the landmark. The range measurement of the landmark viewed from the robot writes

$$y_n = l_n - m_n + w_n, \tag{3.1}$$

where w_n is a Gaussian noise (the uncertainty associated with the telemeter's measurement). We search to estimate m and l with a Kalman filter with initial state estimation error $\mathbf{e}_{0|0} = [m_0 - \hat{m}_{0|0}, l_0 - \hat{l}_{0|0}]$ whose initial covariance is $\mathbf{P}_{0|0} = \mathrm{diag}([\sigma_m, \sigma_l]^2)$, i.e. the robot has a prior (uncertain) knowledge on the landmark and on its own position. Assume the robot now observes the landmark with a perfect telemeter (w_n has standard deviation equal to zero), which is an extreme yet illustrative case. The covariance matrix of the state error, after this measurement has been incorporated writes according to the Kalman filter's equations

$$\mathbf{P}_{1|1} = \frac{\sigma_m^2 \sigma_l^2}{\sigma_m^2 + \sigma_l^2} \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \tag{3.2}$$

which has rank 1. In this limit of perfect measurement, the landmark and robot estimates become fully correlated. As $\max(\sigma_m^2, \sigma_l^2) \geq \frac{\sigma_m^2 \sigma_l^2}{\sigma_m^2 + \sigma_l^2} \geq \min(\sigma_m^2, \sigma_l^2)$, we see the uncertainty on both the robot's and landmark's position has decreased, albeit not necessarily much (typically a factor 2). However, full correlation implies that the instant when the robot knows perfectly its position, it may infer the exact position of the landmark.

What we see is that the state remains uncertain after measurement has been performed (and will always remain so if we don't have access to absolute measurements), but in a particular direction of the state space (that is, the Kernel of the covariance matrix $[1, -1]^T$), much information has been gained: in this direction the filter has aquired full certainty. This makes the SLAM problem quite specific.

In more control theoretic terms, this is a matter of observability: the SLAM problem is unobservable, which is rather uncommon in state estimation for engineering problems. Indeed, the robot obtains only relative information as shifting the system as $m_n \leftarrow m_n + \delta d, l_n \leftarrow l_n + \delta d$ lets the measurement y_n unchanged. This hold even if the robot is moving following

$$m_{n+1} = m_n + u_n, \tag{3.3}$$

with u_n a known input (e.g. stemming from odometers) and that multiple observations are done, as shifting the robot lets the propagation function unchanged (if $m_n \leftarrow m_n + \delta d$ and $m_{n+1} \leftarrow m_{n+1} + \delta d$, (3.3) still holds). We say $\mathbf{d} = [1,1]^T$ is an observable direction, or degree of freedom, of the system (3.1)-eq:robot1d. Indeed, we saw that $[1,-1]^T$ is a vector along which there is no statistical dispersion (it lies in the Kernel of the covariance matrix), and its orthocomplement is in fact a direction in which on the contrary no information may be acquired. The intuitive explanation is clear [45]: "if the robot and landmark positions are shifted equally, it will not be possible to distinguish the shifted position from the original one through the measurements.".

In terms of mathematics, the fact the Kalman filter may not acquire information in this direction may be explained by resorting to the so-called information matrix, that is, the inverse of the covariance matrix. We have indeed that at all times and no matter the measurements $\mathbf{d}^T \mathbf{P}^1_{n|n} \mathbf{d}$ is non-increasing [10] as long as measurements are relative to the robot's frame. This holds for this toy example as

$$\mathbf{d}^T \mathbf{P}_{0|0} \mathbf{d} = \mathbf{d}^T \mathbf{P}_{n|n} \mathbf{d} = \sigma_r^{-2} + \sigma_l^{-2}, \tag{3.4}$$

However this consistency property w.r.t. unobservable directions is not guaranteed when we move to nonlinear problems: when using an EKF $\mathbf{P}_{n|n}$ depend on state estimates through the computations of Jacobians about the estimated trajectory. If this quantity is increasing, we say the filter acquires spurious information along the direction $\mathbf{d}$ and this is what is referred to as inconsistency in the present context of statistical filtering.

We now provide the three main examples of unobservable directions for SLAM systems.

Example 1 (2D SLAM)**.** *The state consists on the robot position, the robot heading, and landmark positions [45]. There are three unobservable direction corresponding to the global translation and rotation of the system.*

Example 2 (3D monocular visual SLAM)**.** *The state consists on the robot position, the robot heading, and landmark positions in 3D where observation are provided with a monocular camera. There are seven unobservable directions: corresponding to the global translation, global rotation and scale of the system [46].*

Example 3 (3D stereo visual SLAM)**.** *The state is the same as in Example 2 where observation are provided with a stereo camera. There are six unobservable directions: corresponding to the global translation and global rotation of the system.*

Example 4 (3D visual inertial SLAM)**.** *The state is the same as in Example 2 enhanced with robot velocity, where an IMU provides scale information and allow to recover roll and pitch. There are thus four unobservable directions corresponding to the global translation and the rotation around gravity (yaw) [47].*

1.2 1.2 EKF SLAM Inconsistency: Previous Literature

EKF inconsistency is defined as the fact that the filter returns a covariance matrix that is too optimistic [48], leading to inaccurate estimates. EKF inconsistency in the context of SLAM has been the object of many papers, see e.g. [46,47,49–56]. Theoretical analysis [49,53,56] reveals inconsistency is caused by the inability of EKF to reflect the unobservable degrees of freedom of SLAM raised in the above section. Indeed, the filter tends to erroneously acquire information along the directions spanned by these unobservable degrees of freedom. The Observability Constrained (OC)-EKF [52,56] constitutes one of the most advanced solutions to remedy this problem and has been fruitfully adapted, e.g. for VIO, cooperative localization, and unscented Kalman filter [5,45,46]. The idea is to pick a linearization point that is such that the unobservable subspace "seen" by the filter is of appropriate dimension.

2 2 Contributions

In this chapter we propose a novel general theory. We first propose to build EKFs based on an alternative error $\mathbf{e} = \eta(\chi, \hat{\chi})$, generalizing the IEKF methodology [42], where we opt for EKF to obtain theoretical guarantee which can not be demonstrated with UKF. This means the covariance matrix $\mathbf{P}$ reflects the dispersion of $\mathbf{e}$, and not of $\chi - \hat{\chi}$. When unobservability stems from symmetries, the technique may resolve the consistency issues of the EKF. Indeed symmetries are encoded by the action $\phi_\alpha(\chi)$ of a transformation group G [57], where $\alpha \in G$ denotes the corresponding infinitesimal unobservable transformation. Under the simple condition that **the image of matrix $\frac{\partial}{\partial \alpha}\eta(\phi_\alpha(\chi), \chi)$ is independent of χ**, the EKF based on $\mathbf{e}$ is proved to possess the desirables properties of the linear case regarding unobservability, and is thus consistent.

2.1 2.1 Specific Application to SLAM

The specific application to SLAM was released on Arxiv in 2015 [10] as part of Axel Barrau's PhD and encountered immediate successes reported in [43,58–62], although the work was never published elsewhere.

More precisely, [9] notice back the SLAM problem bears a nontrivial Lie group structure. [10] formalize the group introduced in [9] and call it $SE_{l+1}(3)$, and proved that for odometry based SLAM, using the right invariant error of $SE_{l+1}(3)$ and devising an EKF based on this error, i.e., a Right-Invariant EKF (RIEKF), the linearized system possesses the desirable properties of the linear case, since it automatically correctly captures unobservable directions for SLAM. Thus, virtually all properties of the linear Kalman filter regarding unobservability may be directly transposed: the information about unobservable directions is non-increasing (see Proposition 2), the dimension of the unobservable subspace has appropriate dimension (this relates to the result of OC-EKF [5,45,56]), the filter's output is invariant to linear unobservable transformations, even if they are stochastic and thus change the EKF's covariance matrix along unobservable directions [58]. The right-invariant error for the proposed Lie group structure was also recently shown to lead to deterministic observers having exponential convergence properties in [63].

Along the same lines, using the right-invariant error of the group $SE_2(3)$, [42] propose alternative consistent IEKF for visual inertial SLAM and VIO applications [59–62,62]. In particular, [62] demonstrate that an alternative Invariant MSCKF based on the right-invariant error of $SE_2(3)$ naturally enforces the state vector to remain in

the unobservable subspace, a consistency property which is preserved when considering point and line features [62], or when a network of magnetometers is available [61].

2.2 2.2 Chapter's Organization

Section 3 presents the general theory. Section 4 applies the theory to the general problem of navigation in the absence of absolute measurements, as typically occurs in GPS-denied environments. Section 5 is dedicated to SLAM and compares the proposed EKF to conventional EKF, OC-EKF [56], robocentric mapping filter [64] and iSAM [65,66].

Preliminary ideas and results can be found in the preprint [10] posted on Arxiv in 2015. Although the present chapter is a major rewrite, notably including a novel general theory encompassing the particular application to SLAM of [10], [10] is a technical report serving as preliminary material for the present chapter. Matlab codes used for the chapter are available at `https://github.com/CAOR-MINES-ParisTech/esde`.

3 3 General Theory

Let us consider the following dynamical system in discrete time with state $\chi_n \in \mathcal{M}$ and observations $\mathbf{y}_n \in \mathbb{R}^p$:

$$\chi_n = f(\chi_{n-1}, \mathbf{u}_n, \mathbf{w}_n), \tag{3.5}$$

$$\mathbf{y}_n = h(\chi_n, \mathbf{v}_n), \tag{3.6}$$

where $f(\cdot)$ is the function encoding the evolution of the system, $\mathbf{w}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_n)$ is the Gaussian process noise, $\mathbf{u}_n$ is the input, $h(\cdot)$ is the observation function and $\mathbf{v}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_n)$ a Gaussian measurement noise. This model is slightly more general than (4.13)-(4.14).

We now define mathematical symmetries, see [15,57].

Definition 2. *An action of a (Lie) group G on $\mathcal{M}$ is defined as a family of bijective maps $\phi_\alpha : \mathcal{M} \to \mathcal{M}$, $\alpha \in G$ satisfying*

$$\forall \chi \in \mathcal{M} \quad \phi_{Id}(\chi) = \chi, \tag{3.7}$$

$$\forall \alpha, \beta \in G, \chi \in \mathcal{M} \quad \phi_\alpha\left(\phi_\beta(\chi)\right) = \phi_{\alpha\beta}(\chi), \tag{3.8}$$

where Id corresponds to the identity of the group G.

Definition 3. *Let $\phi_.(\cdot)$ be defined as in* (3.7)-(3.8). *We say that system* (3.5)-(3.6) *is totally invariant under the action of $\phi_.(\cdot)$ if*

1. *the dynamics are equivariant under $\phi_.(\cdot)$, i.e.,*

$$\forall \alpha, \chi, \mathbf{u}, \mathbf{w} \quad \phi_\alpha(f(\chi, \mathbf{u}, \mathbf{w})) = f(\phi_\alpha(\chi), \mathbf{u}, \mathbf{w}), \tag{3.9}$$

2. *the observation map $h(\cdot)$ is invariant w.r.t. $\phi_.(\cdot)$*

$$\forall \alpha, \chi \quad h(\phi_\alpha(\chi), \mathbf{v}) = h(\chi, \mathbf{v}). \tag{3.10}$$

"Symmetry" is defined as invariance to transformations $\phi_\alpha(\cdot)$. Throughout the chapter we will rather use the term *invariant*, along the lines of the preceding definition.

As in this chapter we pursue the design of consistent EKFs, we will focus on the system "seen" by an EKF: it consists of the linearization of system (3.5)-(3.6) about the estimated trajectory $(\hat{\chi}_n)_{n\geq 0}$ in the state space. Along the lines of [45,53] we use the linearized system about a trajectory.

Let $(\chi_n)_{n\geq 0}$ denote a solution of (3.5) with noise turned off. The local observability matrix [67] at χ_{n_0} for the time interval between time-steps n_0 and n_0+N is defined as

$$\mathcal{O}(\chi_{n_0}) = \begin{bmatrix} \mathbf{H}_{n_0} \\ \mathbf{H}_{n_0+1}\mathbf{F}_{n_0+1} \\ \vdots \\ \mathbf{H}_{n_0+N}\mathbf{F}_{n_0+N}\cdots\mathbf{F}_{n_0+1} \end{bmatrix}, \tag{3.11}$$

with the Jacobians $\mathbf{F}_n = \frac{\partial f}{\partial \chi}|_{\chi_{n-1},\mathbf{u}_n,\mathbf{w}_n}$, $\mathbf{H}_n = \frac{\partial h}{\partial \chi}|_{\chi_n,\mathbf{v}_n}$.

First we show the directions spanned by the action of G are necessarily unobservable directions of the linearized system, that is, they lie in the kernel of the observability matrix.

Theorem 1. *If system* (3.5)-(3.6) *is invariant in the sense of Definition 3, then the directions* $\frac{\partial}{\partial\alpha}\phi_\alpha|_{Id}(\chi_{n_0})$ *infinitesimally spanned by* $\phi_\cdot(\cdot)$ *at any* χ_{n_0} *necessarily lie in* $\mathrm{Ker}\,\mathcal{O}(\chi_{n_0})$, *with* $\mathcal{O}(\chi_{n_0})$ *defined by* (3.11), *and are thus unobservable.*

Proof. Differentiating[1] (3.9) and (3.10) w.r.t. α at Id we obtain

$$\frac{\partial}{\partial\alpha}\phi_\alpha\Big|_{\alpha=Id}(f(\chi,\mathbf{u},\mathbf{w})) = \frac{\partial f}{\partial\chi}|_{(\chi,\mathbf{u},\mathbf{w})}\frac{\partial}{\partial\alpha}\phi_\alpha\Big|_{\alpha=Id}(\chi), \tag{3.12}$$

$$\frac{\partial h}{\partial\chi}|_{(\chi,\mathbf{v})}\frac{\partial}{\partial\alpha}\phi_\alpha\Big|_{\alpha=Id}(\chi) = \mathbf{0} \quad \forall\chi\in\mathcal{M}. \tag{3.13}$$

Let $(\chi_n)_{n\geq 0}$ denote a solution of (3.5) with noise turned off. (3.13) applied at χ_{n_0} yields $\mathbf{H}_{n_0}\frac{\partial}{\partial\alpha}\phi_\alpha|_{Id}(\chi_{n_0}) = \mathbf{0}$. Considering then (3.12) at χ_{n_0} leads to $\frac{\partial}{\partial\alpha}\phi_\alpha|_{Id}\left(f(\chi_{n_0},\mathbf{u}_{n_0+1},\mathbf{w}_{n_0+1})\right) = \mathbf{F}_{n_0+1}\frac{\partial}{\partial\alpha}\phi_\alpha|_{Id}\left(\chi_{n_0}\right)$. Applying (3.13) at $\chi_{n_0+1} = f(\chi_{n_0},\mathbf{u}_{n_0+1},\mathbf{w}_{n_0+1})$ yields $\mathbf{H}_{n_0+1}\frac{\partial}{\partial\alpha}\phi_\alpha|_{Id}\left(\chi_{n_0+1}\right) = \mathbf{0}$, and thus $\mathbf{H}_{n_0+1}\mathbf{F}_{n_0+1}\frac{\partial}{\partial\alpha}\phi_\alpha|_{Id}\left(\chi_{n_0}\right) = \mathbf{0}$. A simple recursion proves $\frac{\partial}{\partial\alpha}\phi_\alpha|_{Id}\left(\chi_{n_0}\right) \subset \mathrm{Ker}\,\mathcal{O}(\chi_{n_0})$. □

Indeed, no matter the number of observations and moves, we are inherently unable to detect an initial (infinitesimal) transformation $\phi_\alpha(\cdot)$, the problem being *invariant* to it.

3.1 3.1 Observability Issues of the Standard EKF

Let $(\hat{\chi}_n)_{n\geq 0}$ be a sequence of state estimates given by an EKF. The linearized system "seen" by the EKF involves the *estimated* observability matrix

$$\hat{\mathcal{O}}(\hat{\chi}_{n_0}) = \begin{bmatrix} \hat{\mathbf{H}}_{n_0} \\ \hat{\mathbf{H}}_{n_0+1}\hat{\mathbf{F}}_{n_0+1} \\ \vdots \\ \hat{\mathbf{H}}_{n_0+N}\hat{\mathbf{F}}_{n_0+N}\cdots\hat{\mathbf{F}}_{n_0+1} \end{bmatrix}, \tag{3.14}$$

[1] On Lie groups differentiation can indeed be rigorously defined as $\frac{\partial}{\partial\alpha}\phi_\alpha\Big|_{\alpha=Id}(\chi)\delta\alpha := \frac{d}{ds}\phi_{\exp(s\delta\alpha)}(\chi)|_{s=0}$ with $\delta\alpha$ in the Lie algebra, which mean partial derivative of $\phi_\alpha(\chi)$ with respect to α at $\alpha = Id$.

where Jacobians are computed *at the estimates*. The directions spanned by $\phi_\cdot(\cdot)$ at $\hat{\chi}_{n_0}$ necessarily lie in $\mathrm{Ker}\mathcal{O}(\hat{\chi}_{n_0})$, as proved by Proposition 1, *but* there is a null probability that they lie in $\mathrm{Ker}\hat{\mathcal{O}}(\hat{\chi}_{n_0})$, because of the noise and Kalman updates, see [45]. A major consequence is that the EKF gains spurious information along the unobservable directions.

In fact, this problem stems from the choice of estimation error $\chi - \hat{\chi}$ that does not match unobservability of system (3.5)-(3.6): changing the estimation error may resolve the problem.

3.2 3.2 EKF Based on a Nonlinear Error

In this section, we define an EKF based on a nonlinear function $\eta(\chi, \hat{\chi}) \in \mathcal{M}$ that provides an alternative to the usual linear estimation error $\chi - \hat{\chi}$. We prove consistency under compatibility assumptions of the group action and $\eta(\cdot)$.

The methodology builds upon the alternative errors

$$\mathbf{e}_{n-1|n-1} = \eta\left(\chi_{n-1}, \hat{\chi}_{n-1|n-1}\right), \tag{3.15}$$

$$\mathbf{e}_{n|n-1} = \eta\left(\chi_{n}, \hat{\chi}_{n|n-1}\right). \tag{3.16}$$

The filter is displayed in Algorithm 4. As the covariance matrix $\mathbf{P}^e$ is supposed to reflect the dispersion of $\mathbf{e}$, we need to define Jacobians w.r.t our alternative state error. At line 2 $\hat{\mathbf{F}}^e_n$, $\hat{\mathbf{G}}^e_n$ are Jacobians of the error propagation function, and at line 3, $\hat{\mathbf{H}}^e_n$, $\hat{\mathbf{J}}^e_n$ are Jacobians of the error measurement defined through the following first order approximations

$$\mathbf{e}_{n|n-1} \simeq \hat{\mathbf{F}}^e_n \mathbf{e}_{n-1|n-1} + \hat{\mathbf{G}}^e_n \mathbf{w}_n, \tag{3.17}$$

$$\mathbf{y}_n - h\left(\hat{\chi}_{n|n-1}, \mathbf{0}\right) \simeq \hat{\mathbf{H}}^e_n \mathbf{e}_{n|n-1} + \hat{\mathbf{J}}^e_n \mathbf{v}_n. \tag{3.18}$$

At line 4, $\mathbf{e}^+_n$ denotes the (best) error estimate according to the EKF. However, defining the (best) state corresponding to estimate $\hat{\chi}_{n|n}$ is not straightforward as in the linear case where $\hat{\chi}_{n|n} = \hat{\chi}_{n|n-1} + \mathbf{e}^+_n$. At line 5 we use a retraction $\varphi : \mathcal{M} \times \mathbb{R}^q \to \mathcal{M}$, that is, *any* function $\varphi(\cdot)$ which is consistent with the error to the first order, i.e., $\mathbf{e}^+_n \approx \eta(\hat{\chi}_{n|n}, \hat{\chi}_{n|n-1})$. This will make the link between this chapter and Chapter 4, where we first choose the retraction and then implicitly define the error.

Note that, we recover the conventional EKF if we let $\mathbf{e} = \eta(\chi, \hat{\chi}) = \chi - \hat{\chi}$ be the usual linear error.

3.3 3.3 Compatibility Assumptions and Main Consistency Result

The matrix[2] $\frac{\partial}{\partial \alpha}|_{Id}\eta(\phi_\alpha(\chi), \chi)$ reflects how infinitesimal transformations of the state produced by the action of G affect the error variable $\mathbf{e} = \eta(\cdot)$. In Assumption 1 below this matrix is used to define a kind of "compatibility" between an invariance group G and a nonlinear error function η, which leads to the main result of this chapter (Theorem 1).

Assumption 1. *The image of the matrix* $\frac{\partial}{\partial \alpha}|_{Id}\eta(\phi_\alpha(\chi), \chi)$ *is a fixed subspace* $\mathcal{C}$ *that does not depend on* χ.

[2] i.e. $\eta(\phi_\alpha(\chi), \chi)$ is a matrix-valued map $g(\chi, \alpha)$, differentiated w.r.t. α.

Algorithm 4: EKF based on a non-linear state error

Input: Initial estimate $\hat{\chi}_0$ and uncertainty matrix $\mathbf{P}_0^e$
while *filter is running* **do**
  Propagation
    $\hat{\chi}_{n|n-1} = f\left(\hat{\chi}_{n-1|n-1}, \mathbf{u}_n, \mathbf{0}\right)$;
    $\mathbf{P}^e_{n|n-1} = \hat{\mathbf{F}}^e_n \mathbf{P}^e_{n-1|n-1} (\hat{\mathbf{F}}^e_n)^T + \hat{\mathbf{G}}^e_n \mathbf{Q}_n (\hat{\mathbf{G}}^e_n)^T$;
  end
  Update
    $\mathbf{K}_n = \hat{\mathbf{H}}^e_n \mathbf{P}^e_{n|n-1} / \left(\hat{\mathbf{H}}^e_n \mathbf{P}^e_{n|n-1} (\hat{\mathbf{H}}^e_n)^T + \hat{\mathbf{J}}^e_n \mathbf{R}_n (\hat{\mathbf{J}}^e_n)^T\right)$;
    $\mathbf{e}^+_n = \mathbf{K}_n \left(\mathbf{y}_n - h\left(\hat{\chi}_{n|n-1}, \mathbf{0}\right)\right)$;
    $\hat{\chi}_{n|n} = \varphi\left(\hat{\chi}_{n|n-1}, \mathbf{e}^+_n\right)$; // state update
    $\mathbf{P}^e_{n|n} = \left(\mathbf{I} - \mathbf{K}_n \hat{\mathbf{H}}^e_n\right) \mathbf{P}^e_{n|n-1}$;
  end
end

Proposition 1 proved that if the system (3.5)-(3.6) is totally invariant, then the directions infinitesimally spanned by $\phi_\cdot(\cdot)$ at any point χ_{n_0} lie in $\mathrm{Ker}\,\mathcal{O}(\chi_{n_0})$, with $\mathcal{O}(\chi_{n_0})$ defined by (3.11), and thus they are unobservable. We have also recalled this is not true for the linearized system seen by the EKF, i.e. for $\hat{\mathcal{O}}(\hat{\chi}_{n_0})$ [45]. We have the following powerful result:

Theorem 1. *If the system* (3.5)-(3.6) *is invariant, and under Assumption 1, the unobservable directions $\mathcal{C}$ spanned by $\phi_\cdot(\cdot)$ at any point $\hat{\chi}_{n_0}$,* measured using error $\eta(\cdot)$ *necessary lie in* $\mathrm{Ker}\,\hat{\mathcal{O}}^e(\hat{\chi}_{n_0})$, *with* $(\hat{\mathbf{F}}^e_n, \hat{\mathbf{H}}^e_n)_{n \geq n_0}$ *defined by* (3.17)-(3.18).

Proof. Let $\hat{\mathbf{M}}_n := \frac{\partial}{\partial \alpha}|_{Id} \eta(\phi_\alpha(\hat{\chi}_n), \hat{\chi}_n)$. Recalling (3.18), we have $h(\phi_\alpha(\hat{\chi}_n)) - h(\hat{\chi}_n) \simeq \hat{\mathbf{H}}^e_n \eta(\phi_\alpha(\hat{\chi}_n), \hat{\chi}_n) \simeq \hat{\mathbf{H}}^e_n \hat{\mathbf{M}}_n \delta\alpha$ with $\delta\alpha$ a linearized approximation to $\alpha \in G$. But also $h(\phi_\alpha(\hat{\chi}_n)) - h(\hat{\chi}_n) \simeq \frac{\partial h}{\partial \hat{\chi}}|_{(\hat{\chi}_n, \mathbf{0})} \frac{\partial}{\partial \alpha} \phi|_{Id}(\hat{\chi}_n) \delta\alpha$. From (3.13) the latter is $\mathbf{0}$. Thus $\hat{\mathbf{H}}^e_n u = \mathbf{0}$ for any $u \in \mathcal{C}$, i.e. $\hat{\mathbf{H}}^e_n \mathcal{C} = \mathbf{0}$.

Using (3.17), $\hat{\mathbf{F}}^e$ is defined as $\eta(f(\chi), f(\hat{\chi})) \simeq \hat{\mathbf{F}}^e \eta(\chi, \hat{\chi})$, thus $\eta(f(\phi_\alpha(\hat{\chi}_n)), f(\hat{\chi}_n)) \simeq \hat{\mathbf{F}}^e_{n+1} \eta(\phi_\alpha(\hat{\chi}_n), \hat{\chi}_n) \simeq \hat{\mathbf{F}}^e_{n+1} \hat{\mathbf{M}}_n \delta\alpha$. Besides, using (3.9) yields $\eta(f(\phi_\alpha(\hat{\chi}_n)), f(\hat{\chi}_n)) = \eta(\phi_\alpha(f(\hat{\chi}_n)), f(\hat{\chi}_n)) \simeq \frac{\partial}{\partial \alpha}|_{Id} \eta(\phi_\alpha(f(\hat{\chi}_n)), f(\hat{\chi}_n)) \delta\alpha \in \mathcal{C}$ applying Assumption 1 at $f(\hat{\chi}_n)$. Thus for any $\delta\alpha$ we have $\hat{\mathbf{F}}^e_{n+1} \hat{\mathbf{M}}_n \delta\alpha \subset \mathcal{C}$ and thus $\hat{\mathbf{F}}^e_{n+1} \mathcal{C} \subset \mathcal{C}$. We have thus proved

$$\hat{\mathbf{H}}^e_n \mathcal{C} = \mathbf{0}, \quad \hat{\mathbf{F}}^e_{n+1} \mathcal{C} \subset \mathcal{C}, \quad \text{for any } n \text{ and } \hat{\chi}_n. \tag{3.19}$$

This proves the result through an immediate recursion. □

We obtain the consistency property we pursue: the linearized model has the desirable property of the linear Kalman filter regarding the unobservabilities, when expressed in terms of error $\eta(\cdot)$. As a byproduct, the unobservable subspace seen by the filter is automatically of appropriate dimension.

3.4 3.4 Consequences in Terms of Information

In the linear Gaussian case, the inverse of the covariance matrix output by the Kalman filter is the Fisher information available to the filter (as stated in [48] p. 304). Thus,

the inverse of the covariance matrix $\mathbf{P}^{-1}_{n|n}$ output by any EKF should reflect an absence of information gain along unobservable directions. Otherwise, the output covariance matrix would be too optimistic, i.e., the filter inconsistent [48].

Note that, the covariance matrix $\mathbf{P}^{e}_{n|n}$ *reflects the dispersion of the error* $\mathbf{e}$ of (3.15)-(3.16), as emphasized by the superscript e. We have the following consistency result:

Theorem 2. *Let $u_{n_0} \in Im\ \frac{\partial}{\partial\alpha}|_{Id}\eta(\phi_\alpha(\hat{\chi}_{n_0}), \hat{\chi}_{n_0})$ be an unobservable direction spanned by $\phi_.(\cdot)$ at the estimate $\hat{\chi}_{n_0}$, measured using alternative error $\eta(\cdot)$. Let $(u_n)_{n\geq n_0}$ with $u_n = \hat{\mathbf{F}}^e_n u_{n-1}$ be its propagation through the linearized model. Under Assumption 1 the Fisher information according to the filter about $(u_n)_{n\geq n_0}$ is non-increasing , i.e.,*

$$u_n^T(\mathbf{P}^e_{n|n})^{-1}u_n \leq u_{n-1}^T(\mathbf{P}^e_{n-1|n-1})^{-1}u_{n-1}. \tag{3.20}$$

Proof. At propagation step we have $u_n^T(\mathbf{P}^e_{n|n-1})^{-1}u_n = u_{n-1}^T(\hat{\mathbf{F}}^e_n)^T(\hat{\mathbf{F}}^e_n\mathbf{P}^e_{n-1|n-1}(\hat{\mathbf{F}}^e_n)^T + \hat{\mathbf{G}}^e_n\mathbf{Q}_n(\hat{\mathbf{G}}^e_n)^T)^{-1}\hat{\mathbf{F}}^e_n u_{n-1} \leqslant u_{n-1}^T(\hat{\mathbf{F}}^e_n)^T\left(\hat{\mathbf{F}}^e_n\mathbf{P}^e_{n-1|n-1}(\hat{\mathbf{F}}^e_n)^T\right)^{-1}\hat{\mathbf{F}}^e_n u_{n-1}$ since $\mathbf{Q}$ is positive semidefinite. As $(\hat{\mathbf{F}}^e_n)^{-1}\hat{\mathbf{F}}^e_n = \mathbf{I}$ we have just proved $u_n^T(\mathbf{P}^e_{n|n-1})^{-1}u_n \leqslant u_{n-1}^T(\mathbf{P}^e_{n-1|n-1})^{-1}u_{n-1}^T$.

At update step (in information form) $u_n^T(\mathbf{P}^e_{n|n})^{-1}u_n = u_n^T\left((\mathbf{P}^e_{n|n-1})^{-1} + (\hat{\mathbf{H}}^e_n)^T\hat{\mathbf{R}}_n^{-1}\hat{\mathbf{H}}^e_n\right)u_n$. But using (3.19) we see $u_i \in \mathcal{C}\ \forall i \geq n_0$ and thus $\mathbf{H}^e_n u_n = \mathbf{0}$ so $u_n^T(\mathbf{P}^e_{n|n})^{-1}u_n = u_n^T(\mathbf{P}^e_{n|n-1})^{-1}u_n \leqslant u_{n-1}^T(\mathbf{P}^e_{n-1|n-1})^{-1}u_{n-1}$. □

The theorem essentially ensures the *linearized model* of the filter has a structure which guarantees that the covariance matrix at all times reflects an absence of "spurious" (Bayesian Fisher) information gain over unobservable directions, ensuring strong consistency properties of our alternative EKF.

4 4 Application to Multi-Sensor Fusion for Navigation

In this section, we consider a navigating vehicle or a robot equipped with sensors which only measure quantities relative to the vehicle's frame. Thus the vehicle cannot acquire information about its absolute position and orientation, which results in inevitable unobservability. The state space is $\mathcal{M} = SO(3) \times \mathbb{R}^{3l+3m+k}$ and the state χ is defined as

$$\chi = (\mathbf{R},\ \mathbf{p}_1,\ \cdots,\ \mathbf{p}_l,\ \mathbf{v}_1,\ \cdots,\ \mathbf{v}_m,\ \mathbf{b}) \in \mathcal{M}, \tag{3.21}$$

where $\mathbf{R} \in SO(3)$ represents the orientation of the vehicle, i.e., its columns are the axes of the vehicle's frame, and where

1. $\{\mathbf{p}_i \in \mathbb{R}^3\}_{i=1,\ldots,l}$ are vectors of the global frame, such as the vehicle's position,
2. $\{\mathbf{v}_i \in \mathbb{R}^3\}_{i=1,\ldots,m}$ are velocities in the global frame, and higher order derivatives of the $\mathbf{p}_i$'s.
3. $\{\mathbf{b}\} \in \mathbb{R}^k$, are quantities being invariant to global transformations, such as sensors' biases or camera's calibration parameters.

Without restriction, we consider in the following $l = m = 1$ for convenience of notation.

Definition 4. *The Special Euclidean group $SE(3)$ describes rigid motions in 3D and is defined as $SE(3) = \{\alpha = (\mathbf{R}_\alpha, \mathbf{p}_\alpha), \mathbf{R}_\alpha \in SO(3), \mathbf{p}_\alpha \in \mathbb{R}^3\}$. Given $\alpha, \beta \in SE(3)$, the group operation is $\alpha\beta = (\mathbf{R}_\alpha\mathbf{R}_\beta, \mathbf{R}_\alpha\mathbf{p}_\beta + \mathbf{p}_\alpha)$ and the inverse $\alpha^{-1} = (\mathbf{R}_\alpha^T, -\mathbf{R}_\alpha^T\mathbf{p}_\alpha)$. We denote Id the identity.*

Changes of global frame are encoded as the action $\phi_\alpha(\chi)$ of an element $\alpha \in SE(3)$ on $\mathcal{M}$. Thus, quantities expressed in the global frame (such as the vehicle position) are rotated and translated by the action $\phi_\alpha(\cdot)$, whereas quantities expressed in the vehicle's frame, such as Inertial Measurement Unit (IMU) biases, are left unchanged. The action then writes (with $i = 1,\dots,l$ and $j = 1,\dots,m$)

$$\phi_\alpha(\chi) = \Big(\mathbf{R}_\alpha\mathbf{R},\ \mathbf{R}_\alpha\mathbf{p}_i + \mathbf{p}_\alpha,\ \mathbf{R}_\alpha\mathbf{v}_j,\ \mathbf{b}\Big). \tag{3.22}$$

Assumption 2. *The vehicle's dynamic does not depend on the choice of global frame, and the vehicle's sensors only have access to relative observations, i.e. no absolute information is available.*

As a result, the equations write (3.5)-(3.6) and are invariant to the action of $SE(3)$ in the sense of Definition 3.

To differentiate w.r.t. elements of $SE(3)$ we resort to its Lie algebra and do in detail what is sketched in Footnote 1.

Definition 5. *The Lie algebra $\mathfrak{se}(3)$ of $SE(3)$ encodes small rigid motions about the identity. It is defined as $\{(\delta\boldsymbol{\omega}_\times, \delta\mathbf{p});\ \delta\boldsymbol{\omega}, \delta\mathbf{p} \in \mathbb{R}^3\}$, where $\boldsymbol{\omega}_\times$ is the skew symmetric matrix associated with cross product with $\boldsymbol{\omega} \in \mathbb{R}^3$. For any $\delta\alpha \in \mathfrak{se}(3)$, we have $\alpha := \exp_{SE(3)}(\delta\alpha) \in SE(3)$ where $\exp_{SE(3)}(\cdot)$ denotes the exponential map of $SE(3)$ (for a definition see* (3.34)-(3.36) *below with $l = 1, m = 0, k = 0$).*

Writing $(\mathbf{R}_\alpha, \mathbf{p}_\alpha) = \exp_{SE(3)}[(\delta\boldsymbol{\omega}_\times, \delta\mathbf{p})]$ in (3.25) we see the directions infinitesimally spanned by $\phi_\cdot(\cdot)$ at χ in the direction $\delta\alpha = (\delta\boldsymbol{\omega}_\times, \delta\mathbf{p})$ write for the state (3.21):

$$\frac{\partial}{\partial\alpha}\phi_\alpha|_{Id}(\chi)\delta\alpha = \Big(\delta\boldsymbol{\omega}_\times\mathbf{R},\ \delta\boldsymbol{\omega}_\times\mathbf{p}_i + \delta\mathbf{p},\ \delta\boldsymbol{\omega}_\times\mathbf{v}_j,\ \mathbf{b}\Big) \tag{3.23}$$

where $i = 1,\dots,l$ and $j = 1,\dots,m$.

Example 5. *[SLAM] Consider a simple SLAM system with one robot and one landmark [45]. Let $\mathbf{p}_R$ be the position of the robot, $\mathbf{R}$ the orientation of the robot, and $\mathbf{p}_L$ the landmark's position. The state is*

$$\chi = (\mathbf{R},\ \mathbf{p}_R,\ \mathbf{p}_L) \in \mathcal{M} = SO(3) \times \mathbb{R}^6. \tag{3.24}$$

The dynamics write $f(\chi, \mathbf{u}, \mathbf{w}) = (\mathbf{R}\bar{\mathbf{R}}, \mathbf{p}_R + \mathbf{R}\bar{\mathbf{R}}\bar{\mathbf{p}}, \mathbf{p}_L)$ where $\bar{\mathbf{R}}, \bar{\mathbf{p}}$ denote orientation and position increments typically measured through odometry. The observation of the landmark in the robot's frame is of the form $\mathbf{y} = \tilde{h}(\mathbf{R}^T(\mathbf{p}_L - \mathbf{p}_R))$. Translations and rotations of the global frame correspond to actions of elements $\alpha = (\mathbf{R}_\alpha, \mathbf{p}_\alpha) \in SE(3)$ as

$$\phi_\alpha(\chi) = (\mathbf{R}_\alpha\mathbf{R},\ \mathbf{R}_\alpha\mathbf{p}_R + \mathbf{p}_\alpha,\ \mathbf{R}_\alpha\mathbf{p}_L + \mathbf{p}_\alpha). \tag{3.25}$$

The system is obviously invariant.

Referring to (3.23)*, as $l = 2, m = 0$ and $\mathbf{p}_R = \mathbf{p}_1, \mathbf{p}_L = \mathbf{p}_2$, the directions spanned by $\phi_\cdot(\cdot)$ at χ are as follows*

$$(\delta\boldsymbol{\omega}_\times\mathbf{R},\ \delta\boldsymbol{\omega}_\times\mathbf{p}_R + \delta\mathbf{p},\ \delta\boldsymbol{\omega}_\times\mathbf{p}_L + \delta\mathbf{p}). \tag{3.26}$$

with $(\delta\boldsymbol{\omega}_\times, \delta\mathbf{p}) \in \mathfrak{se}(3)$. As a direct consequence of Prop. 1, those directions are unobservable. The system continues to be invariant even if a sophisticated model is assumed: the motion equations do not depend a choice of global frame.

Example 6. *[VIO, or Visual Inertial Navigation System (VINS)] Consider a vehicle equipped with an IMU and a camera, as in [46]. Let* $\mathbf{v}$ *denote the vehicle velocity,* $\mathbf{b}$ *the IMU bias and/or scale factors. The state is*

$$\chi = (\mathbf{R},\ \mathbf{p}_R,\ \mathbf{v},\ \mathbf{b}) \tag{3.27}$$

and observations correspond to landmarks' bearings in the vehicle frame, whereas inputs $\mathbf{u}_n \in \mathbb{R}^6$ *are provided by an IMU. A change of global frame* $\alpha = (\mathbf{R}_\alpha, \mathbf{p}_\alpha)$ *writes*

$$\phi_\alpha(\chi) = (\mathbf{R}_\alpha\mathbf{R},\ \mathbf{R}_\alpha\mathbf{p}_R + \mathbf{p}_\alpha,\ \mathbf{R}_\alpha\mathbf{v},\ \mathbf{b}), \tag{3.28}$$

where we restrict $\mathbf{R}_\alpha$ *to be around the gravity axis, i.e. with* $\mathbf{R}_\alpha\mathbf{g} = \mathbf{g}$*, since the vertical is measured [46]. The action* $\phi_\cdot(\cdot)$ *is then also invariant.*

4.1 4.1 EKF Based on a Nonlinear Error

For the general state χ of (3.21) consider the nonlinear error

$$\eta(\chi, \hat{\chi}) = \left(\mathbf{R}\hat{\mathbf{R}}^T, \hat{\mathbf{p}}_i - \hat{\mathbf{R}}\mathbf{R}^T\mathbf{p}_i, \hat{\mathbf{v}}_j - \hat{\mathbf{R}}\mathbf{R}^T\mathbf{v}_j, \mathbf{b} - \hat{\mathbf{b}}\right). \tag{3.29}$$

where $i = 1,\dots,l$ and $j = 1,\dots,m$. We set $l = m = 1$ for simplicity. To linearize, we have the following first order vector approximation of (3.29) that lives in $\mathbb{R}^{3(1+l+m)+k}$

$$\check{\eta}(\chi, \hat{\chi}) = \left(\mathbf{e}_R,\ \hat{\mathbf{p}} - (\mathbf{e}_R)_\times\mathbf{p},\ \hat{\mathbf{v}} - (\mathbf{e}_R)_\times\mathbf{v},\ \mathbf{b} - \hat{\mathbf{b}}\right), \tag{3.30}$$

$$\mathbf{R}\hat{\mathbf{R}}^T = \exp_{SO(3)}(\mathbf{e}_R) \simeq \mathbf{I} + (\mathbf{e}_R)_\times + o(\|\mathbf{e}_R\|^2), \tag{3.31}$$

where $\mathbf{e}_R \in \mathbb{R}^3$.

Theorem 3. *The error* (3.29) *is compatible with the action* (3.22) *of* $SE(3)$ *in the sense of Assumption 1.*

Proof. For $\alpha \in SE(3)$, using (3.29) we obtain that

$$\begin{aligned}\eta(\phi_\alpha(\chi), \chi) = \Big(&\mathbf{R}_\alpha\mathbf{R}\mathbf{R}^T,\ \mathbf{p} - \mathbf{R}(\mathbf{R}_\alpha\mathbf{R})^T[\mathbf{R}_\alpha\mathbf{p} + \mathbf{p}_\alpha],\\ &\mathbf{v} - \mathbf{R}(\mathbf{R}_\alpha\mathbf{R})^T\mathbf{R}_\alpha\mathbf{v},\ \mathbf{b} - \mathbf{b}\Big) = \left(\mathbf{R}_\alpha, -\mathbf{R}_\alpha^T\mathbf{p}_\alpha,\ \mathbf{0}_3,\ \mathbf{0}_k\right),\end{aligned} \tag{3.32}$$

such that $\eta(\phi_\alpha(\chi), \chi)$ turns out to be *independent* of χ, and the result is readily obtained by differentiation at $\alpha = Id$. □

4.2 4.2 Choice of the Retraction

When the state space is a Lie group, and one uses errors that are invariant with respect to right multiplication, the theory of IEKF [16,42] suggests to use $\varphi(\chi, \mathbf{e}) = \exp(\mathbf{e})\chi$ where $\exp(\cdot)$ denotes the Lie group exponential. In [9], it was noticed a natural Lie group structure underlies the (odometry based) SLAM problem. [10] formalize more elegantly this group, and called it $SE_{l+1}(3)$. The current chapter is a generalization to more complete state (3.21), which may be endowed with the group structure $SE_{l+m}(3)\times\mathbb{R}^k$ (direct product of groups $SE_{l+m}(3)$ and $\mathbb{R}^k$). For state (3.21) we thus suggest $\chi^+ = \varphi(\chi, \mathbf{e}) := \exp_{SE_{l+m}(3)\times\mathbb{R}^k}(\mathbf{e})\chi$, i.e.

$$\varphi(\chi, \mathbf{e}) = \left(\delta\mathbf{R}^+\mathbf{R}, \delta\mathbf{R}^+\mathbf{p}_i + \delta\mathbf{p}_i^+, \delta\mathbf{R}^+\mathbf{v}_j + \delta\mathbf{v}_j^+, \mathbf{b} + \delta\mathbf{b}^+\right) \tag{3.33}$$

Error	EKF [56]	**proposed**, see (3.29)	Robocent. [64]
orientation	$\hat{\theta}-\theta$	$\hat{\theta}-\theta$	$\hat{\theta}-\theta$
position	$\hat{\mathbf{p}}_R-\mathbf{p}_R$	$\hat{\mathbf{p}}_R-\mathbf{R}(\hat{\theta})\mathbf{R}(\theta)^T\mathbf{p}_R$	$\mathbf{R}(\hat{\theta})^T\hat{\mathbf{p}}_R-\mathbf{R}(\theta)^T\mathbf{p}_R$
landmark	$\hat{\mathbf{p}}_L-\mathbf{p}_L$	$\hat{\mathbf{p}}_L-\mathbf{R}(\hat{\theta})\mathbf{R}(\theta)^T\mathbf{p}_L$	$\mathbf{R}(\hat{\theta})^T(\hat{\mathbf{p}}_L-\hat{\mathbf{p}}_R)-\mathbf{R}(\theta)^T(\mathbf{p}_L-\mathbf{p}_R)$

Figure 3.2: Alternative state error definitions on the 2D SLAM problem for different solutions (extension of the approach to 3D is immediate), with state $\chi=(\theta,\mathbf{p}_R,\mathbf{p}_L)$ with $\mathbf{p}_R$ the position of the robot, θ its orientation, and $\mathbf{p}_L$ a landmark position. $\mathbf{R}(\theta)$ denotes the planar rotation of angle θ.

with $i=1,\dots,l,\ j=1,\dots,m$, and where

$$\left[\begin{array}{c|ccccc} \delta\mathbf{R}^+ & \delta\mathbf{p}_1^+ & \cdots & \delta\mathbf{p}_l^+ & \delta\mathbf{v}_1^+ & \cdots & \delta\mathbf{v}_m^+ \\ \hline \mathbf{0} & & & \mathbf{I}_{3l+3m} & & & \end{array}\right] \tag{3.34}$$

$$:=\mathbf{I}+\mathbf{S}+\frac{1-\cos(\|\mathbf{e}_R\|)}{\|\mathbf{e}_R\|^2}\mathbf{S}^2+\frac{\|\mathbf{e}_R\|-\sin(\|\mathbf{e}_R\|)}{\|\mathbf{e}_R\|^3}\mathbf{S}^3, \tag{3.35}$$

$$\mathbf{S}:=\left[\begin{array}{c|ccccc} (\mathbf{e}_R)_\times & \mathbf{e}_{\mathbf{p}_1} & \cdots & \mathbf{e}_{\mathbf{p}_l} & \mathbf{e}_{\mathbf{v}_1} & \cdots & \mathbf{e}_{\mathbf{v}_m} \\ \hline \mathbf{0} & & & \mathbf{0}_{3l+3m} & & & \end{array}\right], \tag{3.36}$$

$\delta\mathbf{b}^+=\mathbf{e}_\mathbf{b}$, $\mathbf{0}=\mathbf{0}_{3+3m+3l\times 3}$ and $\mathbf{I}=\mathbf{I}_{3+3m+3l\times 3}$.

4.3 4.3 Extension to Problems Involving Multiple Robots

Consider a problem consisting of M systems. We have M global orientations, one for each system, that transform as the global frame's orientation. For such problems, we define a collection of χ_i of (3.21), and the alternative state error of the problem $\eta(\cdot)$ merely writes

$$\eta(\chi_1,\dots,\chi_M)=(\eta_1(\chi_1,\hat{\chi}_1),\dots,\eta_M(\chi_M,\hat{\chi}_M)), \tag{3.37}$$

where $\eta_i(\chi_i,\hat{\chi}_i)$ is the error (3.29) for the i-th system.

5 5 Simulation Results

This section considers the 2D wheeled-robot SLAM problem and illustrates the performances of the proposed approach. We conduct similar numerical experiment as in the sound work [45] dedicated to EKF inconsistency and benefits of the OC-EKF, i.e., a robot makes 7 circular loops and 20 landmarks are disposed around the trajectory, see Figure 3.3. We refer the interested reader to the available Matlab code for the parameter setting and reproducing the present results.

We compare our approach to standard EKF which conveys an estimate of the linear error; OC-EKF [45] which linearizes the model in a nontrivial way to enforce the unobservable subspace of $\hat{\mathcal{O}}$ to have an appropriate dimension; robocentric EKF [64,68,69] which express *the state in the robot's frame* and then devise an EKF; and iSAM [65,66], a popular optimization technique both for SLAM and odometry estimation which finds the most likely state trajectory given all past measurements. The differences between the estimation errors used by the various EKF variants are recapped in Figure 3.2.

The results confirm the consistency guarantees of Theorem 1 and Proposition 2 are beneficial to the EKF in practice.

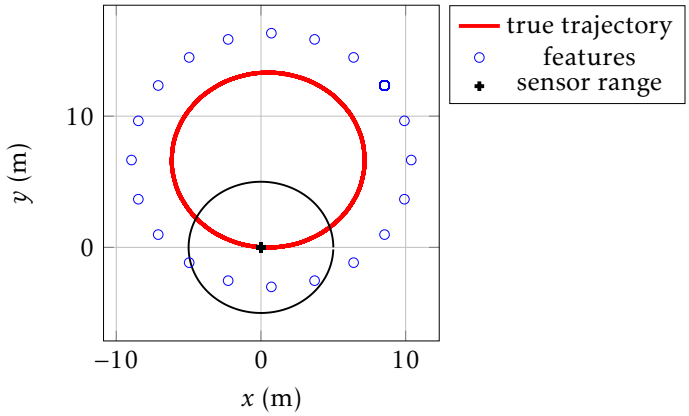


Figure 3.3: Simulated trajectory : the displayed loop is driven by a robot able to measure the relative position of the landmarks lying in a range of 5 m around it. Velocity is constant as well as angular velocity.

5.1 5.1 Monte-Carlo Based Numerical Results

Figure 3.4 displays the Normalized Estimation Error Squared (NEES), Root Mean Square Error (RMSE) and distance to Maximum-Likelihood estimate, over 1000 Monte-Carlo runs.

Consistency Evaluation: the NEES [48] provides information about the filter consistency, such that NEES > 1 reveals an inconsistency issue: the actual uncertainty is higher than the computed uncertainty. As expected, the NEES of the robot pose estimates in Figure 3.4 indicates that the standard EKF is inconsistent, whereas the other approaches are more consistent. The proposed EKF and iSAM obtain the best NEES, whereas the NEES of OC-EKF and robocentric EKF slightly increase after the first turn, i.e. at the first loop closure.

Accuracy Performances: we evaluate accuracy through RMSE of the robot position error. This confirms that: “solving consistency issues improves the accuracy of the estimate as a byproduct, as wrong covariances yield wrong gains" [48]. Numerical results are displayed in Figure 3.1.

Distance to Maximum-Likelihood Estimate: we use as a third performance criterion distance to the estimates returned by iSAM [65], which are optimal in the sense that it returns the Maximum A Posteriori (MAP) estimate. To this respect, we see that the proposed EKF is the closest to iSAM.

Execution Time: we provide the execution time of the filters for the 100 Monte-Carlo runs in Table 3.1, which are implemented in Matlab and tested on Precision Tower 7910 armed with CPU E5-2630 v4 2.20 Hz. The iSAM’s execution time is not included since it cannot be compared: it is implemented using C++ and an optimized code, whereas we used Matlab based simulations. It is thus evidently lower. Regarding computational complexity, our proposed filter has similar complexity as the standard EKF and OC-EKF, since its EKF-based structure makes it quadratic in the state dimension, i.e., number of landmarks. The use of a retraction at the update step instead of mere

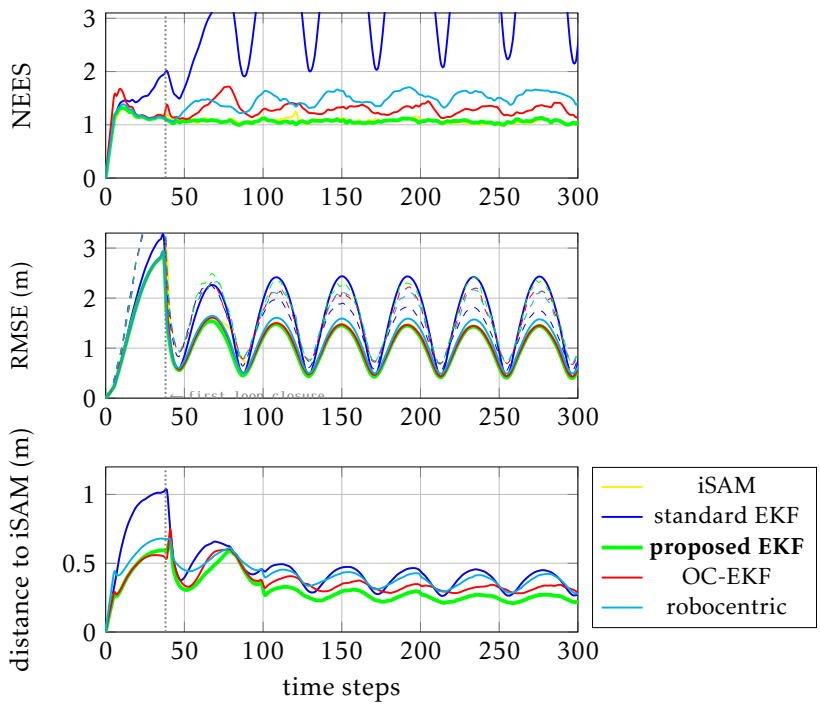


Figure 3.4: Average performances of the different methods over 1000 runs. NEES (for the robot 3-DoF pose) is the consistency indicator, and full consistency corresponds to NEES equal to 1. We see proposed EKF and iSAM are the more consistent, followed by OC-EKF and robocentric EKF, whereas standard EKF is not consistent. The accuracy is evaluated in terms of the robot position RMSE. Standard EKF shows degraded performances as compared to others, which all achieve comparable performances. Finally, filters are evaluated in terms of average proximity of robot's estimated position with iSAM's, which computes the most likely state χ_n given all past measurements $\mathbf{y}_1, \ldots, \mathbf{y}_n$. It is used as a reference of the best achievable estimate. We see that the proposed EKF are the closest to iSAM. Dashed lines correspond to 3σ confidence upper bounds, and we see EKF is over-optimistic. Figures best seen in color.

addition may slightly increase the computational burden, but the impact in the execution time proves negligible. The robocentric filter is penalized because it moves the landmarks during propagation, which in turn impacts the propagation of the covariance matrix. In our solution landmarks remain fixed during propagation. Note that the proposed solution can be implemented using recent techniques [70,71] to decrease computational load. To implement the robocentric and OC-EKF we report that we used the code of [45], see Acknowledgments.

These simulations confirm that regarding SLAM, the proposed filter is an alternative to the OC-EKF. Contrarily to OC-EKF the model is linearized at the (best) estimate, and is thus much closer to standard EKF methodology, and applies to a large class of problems without explicit computation of the unobservable directions.

Filter	EKF	[56]	[64]	**proposed**
NEES	4.05	1.28	1.49	**1.07**
RMSE robot (m)	1.76	1.20	1.27	**1.18**
distance to iSAM (m)	0.45	0.36	0.41	**0.30**
Execution time (s)	**275**	290	414	278

Table 3.1: Average performances and computational time execution of the filters over the 1000 Monte-Carlo runs.

6 6 Conclusion

This work evidences the EKF for robot navigation is not inherently inconsistent but the choice of the estimation error for linearization is pivotal: properly defining the error the EKF shall linearize yields consistency. For SLAM, Monte-Carlo simulations and real experiments have evidenced our alternative EKF outperforms the EKF and achieves similar performance as state-of-the art iSAM. It thus offers an alternative to OC-EKF based on a sound mathematical theory anchored in geometry. Moreover the general theory goes beyond basic SLAM.

CHAPTER 4

A New Approach to Unscented Kalman Filtering on Manifolds

Part of the content of this chapter has been presented in the International Conference on Robotics and Automation (ICRA 2020).

Résumé 🇫🇷

Le présent chapitre présente une nouvelle méthodologie pour le filtrage de Kalman sans-parfum (UKF) sur les variétés et les groupes de Lie. Au-delà des performances d'estimation, l'intérêt principal de l'approche est sa polyvalence, car notre méthode s'applique à de nombreux problèmes d'estimation d'état, et dispose d'une simplicité dans sa mise en œuvre pour ceux ne connaissant que partiellement les variétés et les groupes de Lie. Nous avons développé la méthode sur deux frameworks open source Python et Matlab indépendants, et qui permet d'implémenter et de tester rapidement l'approche. Les codes en ligne contiennent des didacticiels, des documentations et divers exemples que l'utilisateur peut facilement reproduire puis adapter pour un prototypage et une analyse comparative rapides. Le code est disponible à l'adresse `https://github.com/CAOR-MINES-ParisTech/ukfm`.

Chapter abstract

This chapter introduces a novel methodology for unscented Kalman filtering. Beyond filtering performances, the main interests of the approach are its versatility, as the method applies to numerous state estimation problems, and its simplicity of implementation for practitioners not being necessarily familiar with manifolds and Lie groups. We have developed the method on two independent open-source Python and Matlab frameworks we call *UKF-M*, for quickly implementing and testing the approach. The online repositories contain tutorials, documentations, and various relevant robotics examples that the user can readily reproduce and then adapt, for fast prototyping and benchmarking. The code is available at `https://github.com/CAOR-MINES-ParisTech/ukfm`.

1 1 Introduction

The present chapter tackles the more general problem of what happens to the Kalman filter when the state variable lives in a space which is not a vector space. Indeed, in

this case the Kalman state error $\mathbf{e}_n = \mathbf{x}_n - \hat{\mathbf{x}}_n$ introduced at Equation (2.6) does not even make sense. This is a problem as the Kalman filter theory is based on it, and even in the case where the equations are non-linear this error plays a key role to apply the EKF methodology. In this thesis, we chose to focus on the unscented based variant of the Kalman filter, namely the UKF, to study how it may be adapted to the context where the state variable is not a vector. A similar work for the EKF could be carried out, however it had been done already in part in the literature. Moreover, to treat this subject thoroughly one needs to properly study the way the equations are linearized on the manifold, and to derive a rigorous EKF on manifolds the notion of connection (in the sense of differential geometry) arises and complicates the exposure. In our work dedicated to UKF, we tried to keep things as simple as possible, and to excessively limit the prerequisites. This way we hope our approach may prove useful to the practioner, especially in the field of robotics.

1.1 1.1 Literature Review of Kalman Filtering on Manifolds

Filtering on manifolds is historically motivated by aerospace applications where one seeks to estimate (besides other quantities) the orientation of a body in space. Much work has been devoted to making the EKF work with orientations, namely quaternions or rotation matrices. The idea is to make the EKF estimate an error instead of the state directly, leading to error state EKFs [72–75] and their UKF counterparts [22,41,76]. The set of orientations of a body in space is the Lie group $SO(3)$ and efforts devoted to estimation on $SO(3)$ have paved the way to EKF on Lie groups, see [18,34,42,43,77,78] and unscented Kalman filtering on Lie groups, see [19,22,60,79–82].

Lie groups play a prominent role in robotics [83] and have drawn increasing attention for computer vision and robotics applications [42,84,85]. In the context of state estimation and localization, viewing poses as elements of the Lie group $SE(3)$ has proved relevant [7,86–91]. The use of the novel Lie group $SE_2(3)$ introduced in [42] has led to drastic improvement of Kalman filters for robot state estimation [42,43,59,91–95]. Similarly, using group $SE_k(n)$ introduced for SLAM in [9,10] makes EKF consistent or convergent [10,58,62,63,96,97]. Specifically for visual inertial odometry purpose, in [80], the authors devise an UKF that takes advantage of the Lie group structure of the robot's (quadrotor) pose $SE(3)$, and uses a probability distribution directly defined on the group (the distributions in [12]) to generate the sigma points, which is akin to the general unscented Kalman filtering on manifolds of [98]. Finally, there has been attempts to devise UKFs respecting natural symmetries of the systems' dynamics, namely the invariant UKF, see [35,99].

We introduces a novel and general framework for UKF on manifolds that is simpler than existing methods, and whose versatility allows direct application to all manifolds encountered in practice. Indeed, [80,82] proposes UKF implementations based on the Levi-Civita connection but mastering differential geometry is difficult. [19,22,79,80] are reserved for $SO(3)$ and $SE(3)$, while [81] is reserved for Lie groups and requires more knowledge of Lie theory than the present chapter.

1.2 1.2 Organization of the Chapter

In Section 2, we introduce a user-friendly approach to UKF on parallelizable manifolds. Section 3 applies the approach in the particular case where the manifold is a Lie group and recovers [60], but without requiring much knowledge of Lie groups. Section 4 describes the open sourced framework. We then show in Section 5 the method

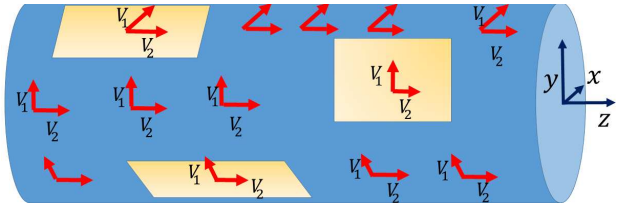


Figure 4.1: The cylinder is a parallelizable manifold. We can define vector fields V_1, V_2 that form a basis of the tangent space at any point.

may actually be extended to numerous manifolds encountered in robotics. The conclusion section discusses theoretical issues and provides clarifications related to Kalman filtering on manifolds.

2 2 Unscented Kalman Filtering on Parallelizable Manifolds

In this section we describe our simple methodology for UKF on parallelizable manifolds. We assume the reader to have approximate prior knowledge and intuition about manifolds and tangent spaces.

2.1 2.1 Parallelizable Manifolds

In order to "write" the equations of the extended or the unscented Kalman filter on a manifold, it may be advantageous to have global coordinates for tangent spaces that extends Definition 1.

Definition 6 (smooth manifold)**.** *A smooth manifold $\mathcal{M}$ of dimension d is said parallelizable if there exists a set of smooth vector fields $\{V_1, V_2, \cdots, V_d\}$ on the manifold such that for any point $\chi \in \mathcal{M}$ the tangent vectors $\{V_1(\chi), V_2(\chi), \cdots, V_d(\chi)\}$ form a basis of the tangent space at χ.*

Example 7 ($\mathbb{R}^p$)**.** *For the vector space $\mathbf{x} \in \mathbb{R}^p$ one can choose as vector fields:*

$$V_1(\mathbf{x}) = \mathbf{e}_1, \ldots, \; V_p(\mathbf{x}) = \mathbf{e}_p, \tag{4.1}$$

where the vector $\mathbf{e}_i$ has zero except 1 at the i-th position. A specificity of $\mathbb{R}^p$ is that the tangent vectors are independent of the point $\mathbf{x}$ and its tangent space is naturally identify with $\mathbb{R}^p$.

Example 8 (cylinder)**.** *The cylinder $\{\mathbf{x} = [x, y, z]^T \in \mathbb{R}^3 \mid x^2 + y^2 = 1\}$ is a basic example with $d = 2$. $V_1(\mathbf{x}) = [y, -x, 0]^T$ and $V_2 = [0, 0, 1]^T$ are two tangent vectors that form a local basis at $\mathbf{x}$, see Figure 4.1. The cylinder is a simple case but the notion of parallelizable manifolds is much broader. In particular, all Lie groups are parallelizable manifolds.*

Example 9 ($SO(3)$)**.** *For the rotation matrices $\mathbf{R} \in SO(3)$ one can choose as vector fields:*

$$V_1(\mathbf{R}) = \mathbf{R}\mathbf{e}_1^\wedge, \; V_2(\mathbf{R}) = \mathbf{R}\mathbf{e}_2^\wedge, \; V_3(\mathbf{R}) = \mathbf{R}\mathbf{e}_3^\wedge, \tag{4.2}$$

where $\mathbf{e}_1 = [1, 0, 0]^T$, $\mathbf{e}_2 = [0, 1, 0]^T$, and $\mathbf{e}_3 = [0, 0, 1]^T$.

Example 10 ($SE(3)$)**.** *Similarly as for rotation, one can choose as vector fields for pose* $\mathbf{T} \in SE(3)$*:*

$$V_1(\mathbf{T}) = \mathbf{T}\mathbf{e}_1^\wedge,\ V_2(\mathbf{T}) = \mathbf{T}\mathbf{e}_2^\wedge,\ V_3(\mathbf{T}) = \mathbf{T}\mathbf{e}_3^\wedge,$$
$$V_4(\mathbf{T}) = \mathbf{T}\mathbf{e}_4^\wedge,\ V_5(\mathbf{T}) = \mathbf{T}\mathbf{e}_5^\wedge,\ V_6(\mathbf{T}) = \mathbf{T}\mathbf{e}_6^\wedge. \tag{4.3}$$

It should be noted, though, that not all manifolds fall in this category. However, we will see in Section 5 how this issue can be addressed over-parameterizing the state.

2.2 2.2 Uncertainty Representation on Parallelizable Manifolds

Our goal is to estimate the state $\chi \in \mathcal{M}$ given all the sensor measurements. As sensors are flawed, it is impossible to exactly reconstruct χ. Instead, a filter maintains a "belief" about the state, that is, its statistical distribution given past sensors' readings. The Kalman filter in $\mathbb{R}^d$ typically maintains a Gaussian belief such that $\chi \sim \mathcal{N}(\hat{\chi}, \mathbf{P})$, which may be re-written in the form:

$$\chi = \hat{\chi} + \xi,\ \xi \sim \mathcal{N}(\mathbf{0}, \mathbf{P}). \tag{4.4}$$

We see that the belief is encoded using only a mean estimate $\hat{\chi}$, and a covariance matrix $\mathbf{P}$ that encodes the extent of dispersion of the belief around the estimate. The error is defined as $\xi = \hat{\chi} - \chi$ which is coherent with (2.6).

Consider a parallelizable manifold $\mathcal{M}$, and let $\{V_1, V_2, \cdots, V_d\}$ denote the associated vector fields. To devise a similar belief on $\mathcal{M}$, one needs of course local coordinates to write the mean $\hat{\chi} \in \mathcal{M}$. This poses no problem, though. The harder part is to find a way to encode dispersion around the estimate $\hat{\chi}$. It is now commonly admitted that the tangent space at $\hat{\chi}$ should encode such dispersion, and that covariance $\mathbf{P}$ should hence reflect dispersion in the tangent space. As additive noise (4.4) makes no sense for $\chi \in \mathcal{M}$, we define a probability distribution $\chi \sim \mathcal{N}_\varphi(\hat{\chi}, \mathbf{P})$, for the random variable $\chi \in \mathcal{M}$ as

$$\chi = \varphi(\hat{\chi}, \xi),\ \xi \sim \mathcal{N}(\mathbf{0}, \mathbf{P}), \tag{4.5}$$

where $\varphi : \mathcal{M} \times \mathbb{R}^d \to \mathcal{M}$ is a smooth function chosen by the user and satisfying $\varphi(\hat{\chi}, \mathbf{0}) = \hat{\chi}$. In (4.5), $\xi \in \mathbb{R}^d$ is a random Gaussian vector that encodes directions of the tangent space at $\hat{\chi}$, $\mathcal{N}(.,.)$ is the classical Gaussian distribution in Euclidean space, and $\mathbf{P} \in \mathbb{R}^{d \times d}$ the associated covariance matrix; and we also impose the Jacobian of $\varphi(\cdot)$ at $(\hat{\chi}, \mathbf{0})$ w.r.t. ξ to be Identity, see [6]. Using the parallelizable manifold property, we implicity use coordinates in the tangent space, as $\xi = (\xi^{(1)}, \xi^{(2)}, \cdots, \xi^{(d)})^T \in \mathbb{R}^d$ encodes the tangent vector $\xi^{(1)} V_1(\hat{\chi}) + \cdots + \xi^{(d)} V_d(\hat{\chi})$. Hence $\varphi(\cdot)$ is called a "retraction", see [6]. In (4.5), the noise-free quantity $\hat{\chi}$ is viewed as the mean, and the dispersion arises through $\varphi(\cdot)$. We stress that the distribution defined at (4.5) is not Gaussian. It is "only" Gaussian in coordinates related to map $\varphi(\cdot)$.

Example 11 ($\mathbb{R}^p$)**.** *Consider Example 7. We naturally recover the addition operator on vector space, i.e.* $\varphi(\mathbf{x}, \xi) = \mathbf{x} + \xi$.

Example 12 ($SO(3)$)**.** *Consider Example 9. Recall tangent vectors at* $\mathbf{R}$ *indicate small motions around* $\mathbf{R} \in SO(3)$. *Tangent vector* $\mathbf{R}\boldsymbol{\omega}^\wedge$ *indeed writes* $\omega_1 V_1(\mathbf{R}) + \omega_2 V_2(\mathbf{R}) + \omega_3 V_3(\mathbf{R})$, *see* (4.2). *We can then choose for* $\varphi(\cdot)$ *the following* $\varphi(\mathbf{R}, \boldsymbol{\omega}) = \mathbf{R}\exp_m(\boldsymbol{\omega}^\wedge) = \mathbf{R}\exp(\boldsymbol{\omega})$, *with* $\exp(\cdot)$ *the exponential map on* $SO(3)$, *see* (2.23).

Example 13 ($SE(3)$)**.** *Consider Example 10. Similary as for* $SO(3)$, *we can then choose for* $\varphi(\cdot)$ *the following* $\varphi(\mathbf{T}, \xi) = \mathbf{T}\exp(\xi)$, *with* $\exp(\cdot)$ *the exponential map on* $SE(3)$, *see* (2.28).

Finding an appropriate map $\varphi(\cdot)$ is not always straightforward. However there exists in theory some "canonical" $\varphi(\cdot)$.

Proposition 1. *One may define define $\varphi(\hat{\chi},\xi)$ as the point of $\mathcal{M}$ obtained by starting from $\hat{\chi}$ and integrating the vector field $\sum_{i=1}^{d}\xi^{(i)}V_i$ during one unit of time. In that case we call $\varphi(\cdot)$ an "exponential map".*

However, we sometimes have no closed form for the exponential map and one resorts to simpler retractions $\varphi(\cdot)$.

2.3 2.3 Bayesian Estimation Using the Unscented Transform

Consider a random variable $\chi \in \mathcal{M}$ with prior probability distribution $p(\chi)$. Suppose we obtain some additional information about χ through a measurement $\mathbf{y}$. The goal is to compute the posterior distribution $p(\chi|\mathbf{y})$. Let

$$\mathbf{y} = h(\chi) + \mathbf{n}, \tag{4.6}$$

be a measurement, where $h(\cdot) : \mathcal{M} \to \mathbb{R}^p$ represents the observation function and $\mathbf{n} \sim \mathcal{N}(\mathbf{0},)$ is a white Gaussian noise in $\mathbb{R}^p$ with known characteristics. The problem of Bayesian estimation we consider is as follows:

1. assume the prior distribution to follow (4.5) with known parameters $\hat{\chi}$ and $\mathbf{P}$;
2. assume one measurement $\mathbf{y}$ of (4.6) is available;
3. approximate the posterior distribution as

$$p(\chi|\mathbf{y}) \approx \varphi(\hat{\chi}^+,\xi^+), \tag{4.7}$$

   where $\xi^+ \sim \mathcal{N}(\mathbf{0},\mathbf{P}^+)$, and find parameters $\hat{\chi}^+$ and $\mathbf{P}^+$.

Letting $\chi = \varphi(\hat{\chi},\xi)$ in (4.6), we see $\mathbf{y}$ provides an information about $\xi \sim \mathcal{N}(\mathbf{0},\mathbf{P})$ and we may use the unscented transform of [1,2] to approximate the posterior $p(\xi|\mathbf{y})$ for ξ as follows, see Algorithm 5: we compute a finite number of samples ξ_j, $j = 1,\ldots,2d$, and pass each of these so-called sigma points through the measurement function

$$\mathbf{y}_j = h\Big(\varphi(\hat{\chi},\big(\xi_j\big)\Big),\ j = 1,\ldots,2d. \tag{4.8}$$

By noting $\mathbf{y}_0 = h(\varphi(\hat{\chi},\mathbf{0}))$ we then compute successively the measurement mean $\hat{\mathbf{y}} = w_m\mathbf{y}_0 + \sum_{j=1}^{2d} w_j\mathbf{y}_j$, the measurement covariance $\mathbf{P}_{\mathbf{yy}} = \sum_{j=0}^{2d} w_j(\mathbf{y}_j - \hat{\mathbf{y}})(\mathbf{y}_j - \hat{\mathbf{y}})^T + \mathbf{N}$ and the cross-covariance $\mathbf{P}_{\xi\mathbf{y}} = \sum_{j=1}^{2d} w_j\xi_j\big(\mathbf{y}_j - \hat{\mathbf{y}}\big)^T$, where w_m and w_j are weights defined in Algorithm 3 of Chapter 2. We then derive the conditional distribution of $\xi \in \mathbb{R}^d$ as

$$p(\xi|\mathbf{y}) \sim \mathcal{N}\Big(\hat{\xi},\mathbf{P}^+\Big), \text{ where} \tag{4.9}$$

$$\mathbf{K} = \mathbf{P}_{\xi\mathbf{y}}\mathbf{P}_{\mathbf{yy}},\ \hat{\xi} = \mathbf{K}(\mathbf{y} - \hat{\mathbf{y}}),\ \mathbf{P}^+ = \mathbf{P} - \mathbf{K}\mathbf{P}_{\mathbf{yy}}\mathbf{K}^T. \tag{4.10}$$

This may be viewed as a Kalman update on the error ξ, in the vein of error state Kalman filtering, see e.g. [75]. The problem is then to convert this into a distribution on the manifold in the form (4.5). We first represent $p(\xi|\mathbf{y})$ as $\hat{\xi} + \xi^+$ with $\xi^+ \sim \mathcal{N}(\mathbf{0},\mathbf{P}^+)$ and $\hat{\xi}$ considered as a noise free mean. We suggest to define the posterior $p(\chi|\mathbf{y})$ as

$$\chi \approx \varphi(\hat{\chi}^+,\xi^+), \quad \xi^+ \sim \mathcal{N}(\mathbf{0},\mathbf{P}^+), \tag{4.11}$$

Algorithm 5: Bayesian updating on parallelizable manifolds with prior (4.5) and observation (4.6)

Input: $\hat{\chi}, \mathbf{P}, \mathbf{y}, \mathbf{N}$;
// set sigma points
$\xi_j = \mathrm{col}(\sqrt{(\lambda+d)\mathbf{P}})_j,\ j = 1,\dots,d,$
$\xi_j = -\mathrm{col}(\sqrt{(\lambda+d)\mathbf{P}})_{j-d},\ j = d+1,\dots,2d;$
// compute measurement sigma points
$\mathbf{y}_0 = h(\varphi(\hat{\chi}, \mathbf{0}));$
$\mathbf{y}_j = h(\varphi(\hat{\chi}, \xi_j)),\ j = 1,\dots,2d;$
// infer covariance matrices
$\hat{\mathbf{y}} = w_m \mathbf{y}_0 + \sum_{j=1}^{2d} w_j \mathbf{y}_j;$
$\mathbf{P}_{\mathbf{yy}} = \sum_{j=0}^{2d} w_j (\mathbf{y}_j - \hat{\mathbf{y}})(\mathbf{y}_j - \hat{\mathbf{y}})^T + \mathbf{N};$
$\mathbf{P}_{\xi\mathbf{y}} = \sum_{j=1}^{2d} w_j \xi_j \left(\mathbf{y}_j - \hat{\mathbf{y}}\right)^T;$
// update state and covariance
$\mathbf{K} = \mathbf{P}_{\xi\mathbf{y}} \mathbf{P}_{\mathbf{yy}}^{-1};$ // gain matrix
$\hat{\chi}^+ = \varphi(\hat{\chi}, \mathbf{K}(\mathbf{y} - \hat{\mathbf{y}}));$
$\mathbf{P}^+ = \mathbf{P} - \mathbf{K}\mathbf{P}_{\mathbf{yy}}\mathbf{K}^T;$
Output: $\hat{\chi}^+, \mathbf{P}^+$;

where we have let

$$\hat{\chi}^+ = \varphi\left(\hat{\chi}, \hat{\xi}\right). \tag{4.12}$$

Note the approximation done in (4.11)-(4.12) actually consists in writing $\varphi\left(\hat{\chi}, \hat{\xi} + \xi^+\right) \approx \varphi\left(\varphi\left(\hat{\chi}, \hat{\xi}\right), \xi^+\right)$.

When $\mathcal{M} = \mathbb{R}^d$ the latter equality holds up to the first order in the dispersions $\hat{\xi}$, ξ^+, both assumed small. In the case where $\mathcal{M}$ is not a vector space, it may be geometrically interpreted as saying that moving from $\hat{\chi}$ along the direction $\hat{\xi} + \xi^+$ approximately consists in moving from $\hat{\chi}$ along $\hat{\xi}$ and then from the obtained point on $\mathcal{M}$ along ξ^+.

2.4 2.4 Unscented Kalman Filtering on Parallelizable Manifolds

Consider the dynamics

$$\chi_n = f(\chi_{n-1}, \omega_n, \mathbf{w}_n), \tag{4.13}$$

where the state χ_n lives in a parallelizable manifold $\mathcal{M}$, ω_n is a known input variable and $\mathbf{w}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_n)$ is a white Gaussian noise in $\mathbb{R}^q$. We consider observations of the form

$$\mathbf{y}_n = h(\chi_n) + \mathbf{n}_n, \tag{4.14}$$

where $\mathbf{n}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{N}_n)$ is a white Gaussian noise with known covariance that we assume additive for clarity of the algorithm derivation only. Compared to (2.9)-(2.10), the state now lives in a manifold. For system (4.13)-(4.14), we model the state posterior conditioned on past measurements using the uncertainty representation (4.5). To propagate the state, we start from the prior distribution $p(\chi_{n-1}) \sim \varphi(\hat{\chi}_{n-1}, \xi_{n-1})$ with $\xi_{n-1} \sim \mathcal{N}(\mathbf{0}, \mathbf{P}_{n-1})$ and $\hat{\chi}_{n-1}$, $\mathbf{P}_{n-1}$ known, and we seek to compute the state propagated distribution in the form

$$p(\chi_n | \chi_{n-1}) \sim \varphi(\hat{\chi}_n, \xi_n) \quad \text{with} \quad \xi_n \sim \mathcal{N}(\mathbf{0}, \mathbf{P}_n). \tag{4.15}$$

We define sigma points using (4.5) and the statistics of noise $\mathbf{w}_n$, and pass them through (4.13). Then, to find $\hat{\chi}_n$ one is faced with the optimization problem of computing a weighted mean on $\mathcal{M}$. This route has already been advocated in [22,41,76,81]. However, to keep the implementation simple and analog to the EKF, we suggest to merely propagate the mean using the unnoisy state model, leading to

$$\hat{\chi}_n = f(\hat{\chi}_{n-1}, \omega_n, \mathbf{0}). \tag{4.16}$$

To compute the covariance $\mathbf{P}_n$ from $\mathbf{P}_{n-1}$ of ξ_{n-1} we use the fact $\mathbf{w}_n$ and ξ_{n-1} are uncorrelated and proceed in two steps.

1. we generate sigma points in $\mathbb{R}^d$ corresponding to $\mathbf{P}_{n-1}$ and pass them through the unnoisy model (4.16) for nonlinear propagation of $\mathbf{P}_{n-1}$ through f. We obtain points χ_n^j on the manifold $\mathcal{M}$, and the distribution of propagated state is described as $\varphi(\hat{\chi}_n, \xi_n)$, with $\hat{\chi}_n$ known from (4.16). We need to be able to locally invert $\xi \mapsto \varphi(\hat{\chi}, \xi)$, i.e., to find a map denoted by $\varphi_{\hat{\chi}}^{-1}(\cdot) : \mathcal{M} \to \mathbb{R}^d$ such that

$$\varphi_{\hat{\chi}}^{-1}(\varphi(\hat{\chi}, \xi)) = \xi + O(\|\xi\|^2), \tag{4.17}$$

   that is, a map that allows one to assess the discrepancy between $\hat{\chi}$ and $\varphi(\hat{\chi}, \xi)$ is ξ indeed. Then we use $\varphi_{\hat{\chi}_n}^{-1}$ to map sigma points χ_n^j back into $\mathbb{R}^d$ and compute their empirical covariance Σ_n.

2. we then generate sigma points for process noise $\mathbf{w}_n$ similarly and obtain another covariance matrix encoding dispersion in $\mathbb{R}^d$ owed to noise, that adds up to Σ_n and thus clearly distinguish the contribution of the state error dispersion ξ_n from noise $\mathbf{w}_n$. When a new measurement arrives, belief is updated via Algorithm 5.

Algorithm 6 summarizes both steps, where the weights defined through `set_weights`(d, α) depend on a scale parameter α (generally set between 10^{-3} and 1), and sigma point dimension, see [1,60] and documentation in source code.

Using (4.16) to propagate the mean while using sigma points to compute covariance is also done in [90], in the particular case of pose compounding on $SE(3)$, with $\varphi(\cdot)$ the $SE(3)$ exponential map.

3 3 Application to UKF on Lie Groups

To apply the preceding methodology to any d-dimensional group $G = \mathcal{M}$, one first defines a basis of the Lie algebra. Then, to any vector $\xi \in \mathbb{R}^d$, recall one may associate an element denoted by $\xi^\wedge$ of the Lie algebra $\mathfrak{g}$. Let the vee operator $\vee$ denote its inverse, as in e.g., [90]. The Lie exponential map maps elements of the Lie algebra to the group. In (4.5) we may choose $\varphi(\hat{\chi}, \xi) := \hat{\chi} \exp_m(\xi^\wedge) = \hat{\chi} \exp(\xi)$, which corresponds to left concentrated Gaussians on Lie groups [78]. Note that, in the Lie group case, choosing left invariant vector fields for the V_i's and following Proposition 1 we exactly recover the latter expression.

We may invert $\varphi(\cdot)$ using the logarithm map $\exp^{-1}(\cdot) := \log(\cdot)$ of G, and we get

$$\varphi(\hat{\chi}, \xi) := \hat{\chi} \exp(\xi), \ \varphi_{\hat{\chi}}^{-1}(\chi) := \log\left(\hat{\chi}^{-1}\chi\right). \tag{4.18}$$

If we alternatively privilegiate right multiplications we have

$$\varphi(\hat{\chi}, \xi) := \exp(\xi)\hat{\chi}, \ \varphi_{\hat{\chi}}^{-1}(\chi) := \log\left(\chi\hat{\chi}^{-1}\right). \tag{4.19}$$

Algorithm 6: UKF on parallelizable manifolds

Input: $\hat{\chi}_{n-1}, \mathbf{P}_{n-1}, \omega_n, \mathbf{Q}_n, \mathbf{y}_n, \mathbf{N}_n, \alpha$;
Propagation
// propagate mean state
$\hat{\chi}_n = f(\hat{\chi}_{n-1}, \omega_n, \mathbf{0})$;
// propagate state error covariance
$\lambda, \{w_j\}_{j=0,\dots,2d} = \texttt{set_weights}(d, \alpha)$;
$\xi_j = \mathrm{col}(\sqrt{(\lambda+d)\mathbf{P}_{n-1}})_j,\ j = 1,\dots,d,$
$\xi_j = -\mathrm{col}(\sqrt{(\lambda+d)\mathbf{P}_{n-1}})_{j-d},\ j = d+1,\dots,2d;$
// use retraction onto manifold
$\chi_n^j = f(\varphi(\hat{\chi}_{n-1}, \xi_j), \omega_n, \mathbf{0}), j = 1,\dots,2d;$
// inverse retract to go back in $\mathbb{R}^d$
$\Sigma_n = \sum_{j=1}^{2d} w_j \varphi^{-1}_{\hat{\chi}_n}(\chi_n^j)\left(\varphi^{-1}_{\hat{\chi}_n}(\chi_n^j)\right)^T$;
// proceed similarly for noise
$\lambda, \{w_j\}_{j=0,\dots,2q} = \texttt{set_weights}(q, \alpha)$;
$\mathbf{w}^j = \mathrm{col}(\sqrt{(\lambda+q)\mathbf{Q}_n})_j,\ j = 1,\dots,q,$
$\mathbf{w}^j = -\mathrm{col}(\sqrt{(\lambda+q)\mathbf{Q}_n})_{j-d},\ j = q+1,\dots,2q;$
$\tilde{\chi}_n^j = f(\hat{\chi}_{n-1}, \omega_n, \mathbf{w}^j), j = 1,\dots,2q;$
$\mathbf{P}_n = \Sigma_n + \sum_{j=1}^{2q} w_j \varphi^{-1}_{\hat{\chi}_n}(\tilde{\chi}_n^j)(\varphi^{-1}_{\hat{\chi}_n}(\tilde{\chi}_n^j))^T$;
end
Update (when measurement $\mathbf{y}_n$ arrives)
Compute $\hat{\chi}_n^+, \mathbf{P}_n^+$ from Algorithm 5 with $\hat{\chi}_n, \mathbf{P}_n$;
end
Output: $\hat{\chi}_n^+, \mathbf{P}_n^+$;

3.1 3.1 Uncertainty on Lie Groups

Based on the above section and Section 2.2, it appears two natural manners to define random variables on Lie groups. We first define the probability distribution $\chi \sim \mathcal{N}_R(\hat{\chi}, \mathbf{P})$ for the random variable $\chi \in G$ as [42,86]

$$\chi = \exp(\xi)\hat{\chi},\ \xi \sim \mathcal{N}(\mathbf{0}, \mathbf{P}), \tag{4.20}$$

where $\mathbf{P} \in \mathbb{R}^{d\times d}$ is a covariance matrix. In (4.20), the original Gaussian ξ of the Lie algebra is moved over by right multiplication to be centered at $\hat{\chi} \in G$, hence the letter R which stands for "right", this type of uncertainty being also referred to as right-equivariant [60]. In (4.20), $\hat{\chi}$ may represent a large, noise-free and deterministic value, whereas $\mathbf{P}$ is the covariance of the small, noisy perturbation ξ. As (4.4), we have defined this probability density function directly in the vector space $\mathbb{R}^d$ such that $\mathcal{N}_R(\cdot)$ is not Gaussian distribution.

One can similarly define the distribution $\chi \sim \mathcal{N}_L(\hat{\chi}, \mathbf{P})$ for left multiplication of $\hat{\chi}$, as

$$\chi = \hat{\chi}\exp(\xi),\ \xi \sim \mathcal{N}(\mathbf{0}, \mathbf{P}). \tag{4.21}$$

The advantages of using (4.20) for SLAM, instead of (4.5) or a standard Euclidean error can be found in [10,59].

3.2 3.2 Applications in Mobile Robotics: the Group $SE_k(d)$

It is well known that orientations of body in spaces are described by elements of $SO(3)$. It is also well known that the use of $SE(3)$ is advantageous to describe the position

and the orientation of a robot (pose), especially for estimation, see [7,86–91]. In [37, 42] the group of double direct isometries $SE_2(3)$ was introduced to address estimation problems for robot navigation when the motion equations are based on an IMU. In [9,10] the group of multiple spatial isometries $SE_k(d)$ was introduced in the context of SLAM. The group $SE_k(d)$, allows recovering $SE(3)$ with $k = 1, d = 3$, $SE(2)$ with $k = 1, d = 2$ and $SO(3)$ with $k = 0, d = 3$. It seems to cover virtually all robotics applications where the Lie group methodology has been so far useful (along with trivial extensions to be mentioned in Section 3.3). Since it was introduced for navigation and SLAM, this group has been successfully used in various contexts, see [10,42,43,58,59,62,63,91–97, 100].

3.3 3.3 The Mixed Case

We call mixed the case where $\mathcal{M} = G \times \mathbb{R}^N$. This typically arises when one wants to estimate some additional parameters besides the state assumed to live in the group G, such as sensor biases, see Chapter 5, Chapter 6, Chapter 8 and Chapter 11. By decomposing the state as $\hat{\chi} = (\hat{\chi}_1, \hat{\chi}_2) \in G \times \mathbb{R}^N$ and letting $\xi = (\xi_1, \xi_2)$, we typically define $\varphi(\cdot)$ through right multiplication as

$$\varphi(\hat{\chi}, \xi) = (\exp(\xi_1)\hat{\chi}_1, \hat{\chi}_2 + \xi_2) \tag{4.22}$$

or if left multiplications are privilegiated $\varphi(\hat{\chi}, \xi) = (\hat{\chi}_1 \exp(\xi_1), \hat{\chi}_2 + \xi_2)$. This way, as many additional quantities as desired may be estimated along the same lines.

Remark 1. *When $G = SE(3)$ for example, it is tempting to let $G' = SO(3)$ and to treat $SE(3)$ as $SO(3) \times \mathbb{R}^3$ along the lines of mixed systems. However, in robotics contexts, it has been largely argued the Lie group structure of $SE(3)$ to treat poses is more relevant than $SO(3) \times \mathbb{R}^3$, as accounting for the coupling between orientation and position leads to important properties, see [7,86–91]. In the same way, $SE_k(3)$ resembles $SO(3) \times \mathbb{R}^{3k}$ but has a special noncommutative group structure having recently led to many successes in robotics, see [10,42,43,58,59,62,63,91–97,100].*

Example 14. *The state χ for fusing IMU with GNSS may be divided into the vehicle state $\chi_1 \in SE_2(3)$ (orientation, velocity and position of the vehicle) and IMU biases $\chi_2 = \mathbf{b} \in \mathbb{R}^6$, see e.g. our example on the KITTI dataset [101]. Further augmenting χ_2 with new parameters, e.g. time synchronization and force variables [102], is straightforward.*

4 4 UKF-M Implementation

We have released both open source Python package and Matlab toolbox *UKF-M* implementations of our method at `https://github.com/CAOR-MINES-ParisTech/ukfm`. Both implementations are wholly independent, and their design guidelines pursue simplicity, intuitiveness and easy adaptation rather than optimization. We adapt the code to the user preferences as follow: the Python code follows class-object paradigm and is heavily documented through the Sphinx documentation generator, whereas the Matlab toolbox contains equivalent functions without class as we believe choosing well function names is best suited for the Matlab use as compared to class definition. The following code snippets are based on the Python package that we recommend using.

Snippet 1: how to devise an UKF on manifolds

```
ukf = ukfm.UKF(
  f=model.f,                # propagation model
  h=model.h,                # observation model
  phi=user.phi,             # retraction
  phi_inv=user.phi_inv,     # inverse retraction
  Q=model.Q,                # process  cov.
  R=model.R,                # observation  cov.
  alpha=user.alpha          # sigma point param.
  state0=state0,            # initial state
  P0=P0)                    # initial covariance
```

Snippet 2: setting $\varphi(\cdot)$, $\varphi^{-1}(\cdot)$ for $\chi := (\mathrm{Rot} \in SO(3), \mathrm{v}, \mathrm{p})$

```
def phi(state, xi):
  return STATE(
  Rot=state.Rot.dot(SO3.exp(xi[0:3])),
  v=state.v + xi[3:6]
  p=state.p + xi[6:9])

def phi_inv(state, hat_state):
  return np.hstack([ # concatenate errors
  SO3.log(hat_state.Rot.T.dot(state.Rot)),
  state.v - hat_state.v,
  state.p - hat_state.p])
```

4.1 4.1 Recipe for Designing a UKF on Manifolds

To devise an UKF for any fusion problem on a parrallelizable manifold (or Lie group) $\mathcal{M}$ the ingredients required in terms of implementation are as follows, see Snippet 1.

1. A model that specifies the functions f and h used in the filter;
2. An uncertainty representation (4.5). This implies an expression for the function $\varphi(\cdot)$ and its inverse $\varphi^{-1}(\cdot)$, defined by the user;
3. Filter parameters, that define noise covariance matrices $\mathbf{Q}_n$, $\mathbf{N}_n$ and weights (λ, w_m, and w_j) through α. Noise covariance values are commonly guided by the model and tuned by the practitioner, whereas α is generally set between 10^{-3} and 1 [1].
4. Initial state estimates $\hat{\chi}_0$ and $\mathbf{P}_0$.

Example 15. *Consider a 3D model whose state contains a rotation matrix* $\mathrm{Rot} \in SO(3)$, *the velocity* $\mathrm{v} \in \mathbb{R}^3$ *and position* $\mathrm{p} \in \mathbb{R}^3$ *of a moving vehicle. Defining* $\varphi(\cdot)$ *and* $\varphi^{-1}(\cdot)$ *allows computing (respectively) a new state and a state error. One possibility is given in Snippet 2, where* $\chi \in SO(3) \times \mathbb{R}^6$, $\varphi(\hat{\chi}, \xi) = \left(\hat{\mathrm{Rot}} \exp(\xi^{(0:3)}), \hat{\mathrm{v}} + \xi^{(3:6)}, \hat{\mathrm{p}} + \xi^{(6:9)}\right)$ *and* $\varphi^{-1}_{\hat{\chi}}(\chi) = (\log(\hat{\mathrm{Rot}}^T \mathrm{Rot}), \mathrm{v} - \hat{\mathrm{v}}, \mathrm{p} - \hat{\mathrm{p}})$.

In the particular case where $\mathcal{M}$ is a Lie group we follow the rules above but we simplify step 2) as follows: we pick an uncertainty representation, either (4.18) or (4.19). This directly implies an expression for the map $\wedge$ and its inverse $\vee$, as well

as for the exponential exp and its (local) inverse log. Applying the present general methodology for the particular case of Lie groups, we recover the method of [60].

Example 16. *We may modify the representation used in Example 15 by viewing the state as an element $\chi \in SE_2(3)$ instead. This defines two alternative retractions. See e.g. implementation for corresponding $\varphi^{-1}(\cdot)$'s in Snippet 3. A quick comparison displayed in Figure 4.2 indicates the $SE_2(3)$-UKF with right multiplications* (4.19) *outperforms the other filters, notably the one based on the naive structure of Example 15.*

4.2 4.2 Unscented Kalman Filtering on Lie Groups

By representing the state error as a variable ξ of the Lie algebra, we can build two alternative unscented filters for any state living in Lie groups following Section 3.1.

Right-UKF-LG: the state is modeled as $\chi_n \sim \mathcal{N}_R(\hat{\chi}_n, \mathbf{P}_n)$, that is, using the representation (4.20) of the uncertainties. The mean state is thus encoded in $\hat{\chi}_n$ and dispersion in $\xi \sim \mathcal{N}(\mathbf{0}, \mathbf{P}_n)$. The sigma points are generated based on the ξ variables, and mapped to the group through the model (4.20). Note that, this is in slight contrast with [80,103], which generate sigma points through a distribution defined directly on the group. The filter consists of two steps along the lines of the conventional UKF: propagation and update, and compute estimates $\hat{\chi}_n$ and $\mathbf{P}_n$ at each n.

Left-UKF-LG: the state is alternatively modeled as $\chi_n \sim \mathcal{N}_L(\hat{\chi}_n, \mathbf{P}_n)$, that is, using the left-equivariant formulation (4.21) of the uncertainties.

4.3 4.3 Implemented Examples

In the code, we implement the frameworks on relevant vanilla robotics examples which are listed as follows:

- 2D vanilla robot localization tutorial based on odometry and GNSS measurements;
- 3D attitude estimation from an IMU equipped with gyro, accelerometer and magnetometer;
- 3D inertial navigation on flat Earth where the vehicle obtains observations of known landmarks;
- 2D SLAM where the UKFs follows [5] to limit computational complexity and adding new observed landmarks in the state;
- IMU-GNSS fusion on the KITTI dataset [101];
- an example where the state lives on the 2-sphere manifold, modeling e.g., a spherical pendulum [8].

We finally enhance code framework, documentation and examples with filter performance comparisons: for each example we simulate Monte-Carlo data and benchmark UKFs and EKFs based on different choices of uncertainty representation (4.5) through accuracy and consistency metrics.

Example 17. *Figure 4.2 displays two EKFs and two UKFs for inertial navigation in the setting of [42], where initial heading and position errors are large, respectively 45 degrees and 1 m. The second UKF, whose uncertainty representation* (4.5) *is based on $SE_2(3)$ exponential, see Section 3.2, clearly outperforms the EKF, the first UKF, and improves the EKF of [42] during the first 10 seconds of the trajectory.*

Snippet 3: defining $\varphi^{-1}(\cdot)$ via (4.18) or (4.19) for $\chi \in SE_2(3)$

```
def phi_inv(state, hat_state):
  chi = state2chi(state)
  hat_chi = state2chi(hat_state)
  # if left multiplication (3.18)
  return SEK3.log(SEK3.inv(hat_hat).dot(chi))
  # if right multiplication (3.19)
  return SEK3.log(chi.dot(SEK3.inv(hat_hat)))
```

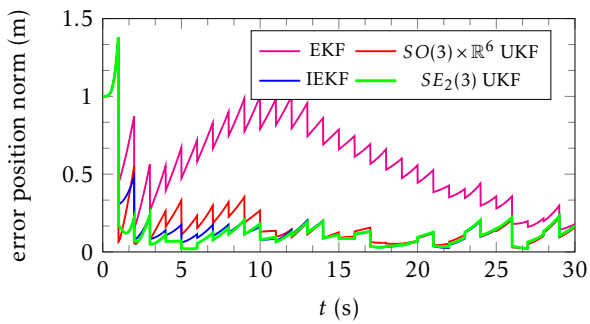


Figure 4.2: Inertial navigation with heavy initial errors in the setting of [42]. $SE_2(3)$-UKF obtains the best results.

5 5 Extension to General Manifolds

The main problem when $\mathcal{M}$ is not parallelizable is that one cannot define a global uncertainty representation through a map $\varphi(\cdot)$ as in (4.5). Indeed $\xi = (\xi^{(1)}, \cdots, \xi^{(d)})$ encodes at any $\chi \in \mathcal{M}$ coordinates in the tangent space related to a basis $(V_1(\chi), \cdots, V_d(\chi))$ of the tangent space. On general manifolds, though, it is always possible to cover the manifold with "patches" $\mathcal{M}_1, \cdots, \mathcal{M}_K$, such that on each patch i we have a set of vector fields $(V_1^{(i)}, \cdots, V_d^{(i)})$ allowing one to apply our methodology. For instance on the 2-sphere one could choose a North-East frame in between the polar circles, and then some other smooth set of frames beyond polar circles. However two main issues arise. First, we feel such a procedure induces discontinuities at the polar circles that will inevitably degrade the filter perfomances. Indeed by moving $\hat{\chi}$ slightly at the polar circle, one may obtain a jump in the distribution $\mathcal{N}_\varphi(\hat{\chi}, \mathbf{P})$ with fixed covariance $\mathbf{P}$, see Figure 4.3. Then, we see the obtained filter wholly depends on the way patches are chosen, which is undesirable.

5.1 5.1 The Lifting "Trick"

It turns out a number of manifolds of interest called homogeneous spaces may be "lifted" to a Lie group, hence a parallelizable manifold. By simplicity[1] we consider as a tutorial example the 2-sphere $\mathcal{M} = \mathbb{S}^2 = \{\mathbf{x} \in \mathbb{R}^3 \mid \|\mathbf{x}\| = 1\}$ with state $\mathbf{x}_n \in \mathbb{S}^2$. As $\mathbf{x}_{n+1}$ and $\mathbf{x}_n$ necessarily lie on the sphere, they are related by a rotation, that is,

$$\mathbf{x}_{n+1} = \mathbf{\Omega}_n \mathbf{x}_n \tag{4.23}$$

[1] Generalizations to the Stiefel manifold $St(p,n)$, that is, a set of p orthonormal vectors of $\mathbb{R}^n$, and hence to the set of p-dimensional subspaces of $\mathbb{R}^n$ called the Grassmann manifold are then straightforward.

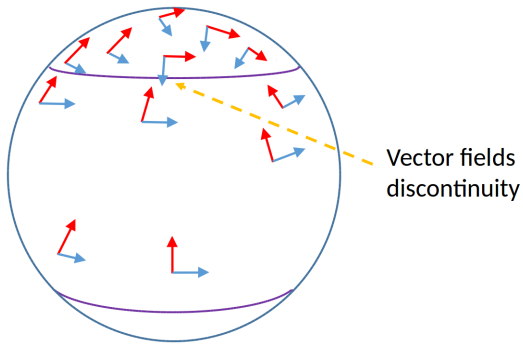


Figure 4.3: We see covering the 2-sphere with 3 parallelizable patches (in between polar circles, and beyond each) inevitably induces discontinuities that may degrade filtering performances. This is a consequence of the theorem that states it is not possible to "comb a hairy ball", see [104].

with $\mathbf{\Omega}_n \in SO(3)$ that may be written as $\exp_m(\omega_n^\wedge)\exp_m(\mathbf{w}_n^\wedge)$ where ω_n is a known input, and $\mathbf{w}_n \sim \mathcal{N}(0, \mathbf{Q}_n)$ represents a noise, see (2.22) for the definition of wedge operator, and $\exp_m(\cdot)$ is the usual matrix exponential of $SO(3)$. We assume $\mathbf{x}_n$ is measured through a linear observation, that is,

$$\mathbf{y}_n = \mathbf{H}\mathbf{x}_n + \mathbf{n}_n \in \mathbb{R}^p. \tag{4.24}$$

Example 18. *We provide a (novel) script which simulates a point of a pendulum with stiff wire living on a sphere, where two components are measured through e.g. a monocular camera, i.e.* $\mathbf{H} = [\mathbf{e}_1, \mathbf{e}_2]^T$.

The dynamics can be lifted into $SO(3)$ by writing $\mathbf{x}_n$ via a rotation matrix $\mathbf{R}_n$, that is, we posit $\mathbf{x}_n = \mathbf{R}_n\mathbf{L}$ with $\mathbf{L} \in \mathbb{R}^3$. In terms of $\mathbf{R}_n$, dynamics (4.23) may be lifted letting $\mathbf{R}_{n+1} = \mathbf{\Omega}_n\mathbf{R}_n$ as then $\mathbf{R}_n\mathbf{L}$ satisfies (4.23) indeed. Similarly, the output in terms of $\mathbf{R}_n$ writes $\mathbf{y}_n = \mathbf{H}\mathbf{R}_n\mathbf{L} + \mathbf{n}_n = \tilde{h}(\mathbf{R}_n) + \mathbf{n}_n$. Having transposed the problem into estimation on the parallelizable manifold $SO(3)$, we can then apply the two UKFs by setting $\varphi(\cdot)$ to either (4.18) or (4.19).

5.2 5.2 Covariance Retrieval

The practitioner may wonder how to retrieve the covariance in the original variables. Assume we have a Gaussian vector $\mathbf{x} \sim (\boldsymbol{\mu}, \Sigma)$, and we want to approximate $g(\mathbf{x})$ as a Gaussian. This might addressed resorting to the unscented transform but a more basic and direct approach is as follows. Consider $\mathbf{A}$ a matrix and $\mathbf{b}$ a vector. Then it is known from probability theory that

$$\mathbf{A}\mathbf{x} + \mathbf{b} \sim (\mathbf{A}\boldsymbol{\mu} + \mathbf{b}, \mathbf{A}\Sigma\mathbf{A}^T). \tag{4.25}$$

Then, we can write $\mathbf{x} = \boldsymbol{\mu} + \mathbf{e}$ with $e \sim \mathcal{N}(\mathbf{0}, \Sigma)$ and linearizing we find $g(\mathbf{x}) \approx g(\boldsymbol{\mu}) + \frac{\partial g}{\partial \mathbf{x}}(\boldsymbol{\mu})\mathbf{e}$ and applying linear Gaussian vectors transform yields approximately $g(\mathbf{x}) \sim (g(\boldsymbol{\mu}), \mathbf{A}\Sigma\mathbf{A}^T)$, where we let $\mathbf{A} := \frac{\partial g}{\partial \mathbf{x}}(\boldsymbol{\mu})$.

In the 2-sphere example of the present section, our uncertainty representation may be taken as $\mathbf{R}_n = \exp(\xi^\wedge)\hat{\mathbf{R}}_n$ with $\xi \sim \mathcal{N}(\mathbf{0}, \mathbf{P})$, see (4.19). As a result it is rather easy to compute the covariance matrix of $\mathbf{R}_n\mathbf{L}$ as follows. We may use linearizations to write

that $\exp(\xi^\wedge) \approx \mathbf{I} + \xi^\wedge$ and thus $\mathbf{R}_n\mathbf{L} = \exp(\xi^\wedge)\hat{\mathbf{R}}_n\mathbf{L} \approx \hat{\mathbf{R}}_n\mathbf{L} + \xi^\wedge\hat{\mathbf{R}}_n\mathbf{L} = \hat{\mathbf{R}}_n\mathbf{L} - (\hat{\mathbf{R}}_n\mathbf{L})^\wedge\xi = \hat{\mathbf{R}}_n\mathbf{L} + \mathbf{A}\xi$ with $\mathbf{A} = -(\hat{\mathbf{R}}_n\mathbf{L})^\wedge$. As a result, the probability distribution of $\mathbf{R}_n\mathbf{L}$ is under a linear approximation $\mathcal{N}(\hat{\mathbf{R}}_n\mathbf{L}, \mathbf{APA}^T)$.

6 6 Concluding Remarks

If we step back a little and look at the bigger picture, we see the main problem when designing filters on a manifold $\mathcal{M}$ is that we often lack coordinates to write down the filter equations on $\mathcal{M}$. Even if we do, e.g. longitude and latitude on the sphere, this implicitly defines probability distributions on the manifold in a way that may not suit the problem well, see Figure 4.3. Over the past decades, researchers have advocated the intrinsic approach based on the tangent space [105]. This way the filter becomes independent of a particular choice of coordinates on the manifold, but it depends on the way tangent spaces at different locations correspond. Notably, we see at lines 5, 6, 7, 9 of Algorithm 5 the covariance matrix $\mathbf{P}^+$ is computed using local information at $\hat{\chi}$, in total disregard of $\hat{\chi}^+$, although $\mathbf{P}^+$ is supposed to encode dispersion at $\hat{\chi}^+$! This means it is up to the user to define the way "Gaussians" are transported over $\mathcal{M}$ from $\hat{\chi}$ to $\hat{\chi}^+$, as early noticed in [80], see also [82]. The route we have followed herein consists in focusing on parallelizable manifolds where a global coordinate system of tangent spaces exists, and readily provides a transport operation over $\mathcal{M}$.

However, there are multiple choices for the parallel transport operation. In [80,82] the authors advocate using the Levi-Civita connection for parallel transport, which depends on the chosen metric, and argue its virtue is that it is torsion free. In the context of state estimation on Lie groups, though, the transport operations that lead to the best performances are not torsion free, see [43]. In cases where it is unclear to the user which transport operation (in our case parallelization+retraction) shall be best, we suggest using our code for quick benchmarking, as done in Figure 4.2. Indeed, the group structures $SE_2(3)$ versus $SO(3) \times \mathbb{R}^6$ actually boil down to particular choices of parallelization (hence transport), and the filter based on $SE_2(3)$ outperforms the other.

CHAPTER 5

Invariant Kalman Filtering for Visual Inertial SLAM

Résumé 🇫🇷

Dans ce chapitre, nous nous appuyons à la fois sur la théorie du filtrage de Kalman sans-parfum sur les groupes de Lie dévelopée dans le Chapitre 4 et plus généralement sur la théorie du filtrage de Kalman invariant, un UKF innovant est dérivé pour le problème de la la localisation et cartographie simultanées basée inertie et vision monoculaire. La position du véhicule, sa vitesse et les positions des points de repère 3D sont considérées comme un seul élément du groupe de Lie (de grande dimension) $SE_{2+p}(3)$, qui constitue l'état, et où les biais des accéléromètres et des gyromètres sont ajoutés à l'état et estimés également. L'efficacité de l'approche est validée à la fois sur des simulations et sur cinq séquences provenant d'un jeu de données réelles.

Chapter abstract

In this chapter, building upon both the theory of Unscented Kalman Filtering on Lie Groups developped in Chapter 4 and more generally the theory of invariant Kalman filtering, an innovative UKF is derived for the monocular visual SLAM problem. The body pose, velocity, and the 3D landmarks' positions are viewed as a single element of a (high dimensional) Lie group $SE_{2+p}(3)$, which constitutes the state, and where the accelerometers' and gyrometers' biases are appended to the state and estimated as well. The efficiency of the approach is validated both on simulations and on five real datasets.

1 1 Introduction

In this chapter, we come back to the problem of SLAM, but from a more practical viewpoint than in Chapter 3. Indeed, our main goal here is to deal with vision measurements.

We consider a robot equipped with inertial sensors, namely an IMU , and a monocular vision for SLAM for Micro Aerial Vehicles (MAVs). We seek to "fuse" those measurements in order to estimate the robot's state. To this aim, we propose a novel UKF that mainly builds upon two components. First, the recent Lie group structure of SLAM advocated in the field of invariant filtering, see [10,58,85] and the results of Chapter 3. Secondly, the UKF on Lie Groups (UKF-LG), whose general methodology has been

recently introduced in Chapter 4. The effectiveness of our algorithm is tested both on simulations and on real data [106]. The method, an UKF-LG visual SLAM, favorably compares to state-of-the-art Kalman filter based solutions.

Note that, in the present chapter we do not specifically address Visual Inertial Odometry (VIO) which is addressed in Chapter 6 and is a powerful technique where the features (i.e., the map) are not included in the state, saving execution time. As in SLAM, VIO estimates the sequential changes of the robot over time using an IMU and cameras, but there is no attempt to build a map [46]. Here, by contrast, we explicitly consider the probabilistic visual SLAM problem, where a consistent map of the environment is also pursued. Note that, even for navigation purposes only, building a map allows loop closures (contrary to visual inertial odometry), a powerful method to drastically reduce uncertainty on the state, when applicable.

1.1 1.1 Contributions and Links with Previous Literature

Over the last decades, tremendous progresses have been achieved in visual localization frameworks, whose estimation and robustness can be improved by tightly coupling visual and inertial informations, which is the major focus of this chapter. Most approaches combine data using filtering based solutions [46,59,80,107–110], or optimization/bundle adjustment techniques, e.g., [84,111,112]. Optimization based methods are more accurate but generally come with higher computational demands, and filtering approaches are well suited to real time applications.

Additionally, it has been shown that, by expressing the EKF estimation error directly on the Lie group and leveraging an IEKF, consistency guarantees can be obtained without ad hoc remedies both for wheel odometry SLAM [10,58] and VIO [59] filtering algorithms. Very recently [62] proposed a multi-state constrained Right IEKF (RIEKF) for 3D VINS. In Section 5 dedicated to experiments, we also apply the framework [10] to the inertial and vision fusion, to implement a RIEKF where the landmarks are part of the state. To our best knowledge this is the first published implementation of an RIEKF for visual inertial SLAM.

Our main contribution is a Right-UKF-LG, which can be viewed as an unscented-based transform alternative to the RIEKF, but which has the advantage of being much more versatile than the RIEKF. Indeed, it spares the user the computation of Jacobians, that can prove difficult, especially in the IEKF framework where Jacobians are defined with respect to the Lie structure, see e.g., [85]. As a result,

- the practitioner can *readily* implement our algorithm when using, e.g., a different camera model, or if one wants to add additional measurements such as GPS measurements outdoors, or a complementary depth sensor;
- should additional parameters/variables be estimated, such as IMU's scale factors and/or harmonization angles, the algorithm is straightforward to adapt following the state augmentation technique of Chapter 4.

Note that, the present chapter presents two new developments on the UKF-LG methodology, that are as follows: a square-root form implementation detailed in Appendix A and a modification to deal with large updates described in Appendix B.

In [80], the authors consider the same visual inertial fusion problem, and devise an UKF that takes advantage of the Lie group structure of the body pose $SE(3)$. The main differences are threefold. First, the Lie group we use $SE_{2+p}(3)$, introduced in [9,10], is much bigger than $SE(3)$, and includes the pose but also the velocity and the landmarks' positions. Then, and more generally, the UKF-LG presented in Chapter 4 generates

sigma points in the Lie algebra, and then uses concentrated Gaussian distributions (as in e.g., [86]) to map them onto the group. In contrast, [80] uses a probability distribution directly defined on the group [12] to generate the sigma points, which is akin to the general unscented Kalman filtering on manifolds of [98]. Moreover, while [80] uses parallel transport operations based on left multiplications, we explore two variants based both on left and right multiplications, and the right one proves to be actually much better.

1.2 1.2 Chapter's Organization

The chapter is organized as follows. Section 2 formulates the fusion problem. Section 2 contains mathematical preliminaries on unscented Kalman filtering on Lie groups. Section 3 describes the two proposed UKFs for monocular visual and inertial SLAM. Section 4 and 5 illustrate the performances of the proposed filters based both on Monte-Carlo simulations and on real datasets.

2 2 Visual Inertial SLAM Problem Modeling

We recall in this section the standard dynamic model for flying devices equipped with an IMU. We then detail the visual measurement model, and we finally pose the SLAM problem.

2.1 2.1 Variables of Interest and Dynamical Model

Let us consider an aerial body equipped with an IMU whose biases are modeled as random walks. Assume moreover that p fixed landmarks of the scene can be tracked visually, and that they constitute the map. The state we want to estimate consists of the position $\mathbf{p}^{\text{IMU}} \in \mathbb{R}^3$, velocity $\mathbf{v}^{\text{IMU}} \in \mathbb{R}^3$, orientation $\mathbf{R}^{\text{IMU}} \in SO(3)$ of the body, the IMU biases $\mathbf{b}^{\omega} \in \mathbb{R}^3$ and $\mathbf{b}^{\mathbf{a}} \in \mathbb{R}^3$, as well as the 3D positions $\mathbf{p}_1, \ldots, \mathbf{p}_p \in \mathbb{R}^3$ of the landmarks in the global frame. The dynamics of the system read:

$$\text{body state dynamics} \begin{cases} \dot{\mathbf{R}}^{\text{IMU}} = \mathbf{R}^{\text{IMU}} (\omega - \mathbf{b}^{\omega} + \mathbf{w}^{\omega})_{\times} \\ \dot{\mathbf{v}}^{\text{IMU}} = \mathbf{R}^{\text{IMU}} (\mathbf{a} - \mathbf{b}^{\mathbf{a}} + \mathbf{w}^{\mathbf{a}}) + \mathbf{g} \\ \dot{\mathbf{p}}^{\text{IMU}} = \mathbf{v}^{\text{IMU}} \end{cases}, \tag{5.1}$$

$$\text{IMU biases dynamics} \begin{cases} \dot{\mathbf{b}}^{\omega} = \mathbf{w}^{\mathbf{b}^{\omega}} \\ \dot{\mathbf{b}}^{\mathbf{a}} = \mathbf{w}^{\mathbf{b}^{\mathbf{a}}} \end{cases}, \tag{5.2}$$

$$\text{landmarks dynamics} \left\{ \dot{\mathbf{p}}_i = \mathbf{0},\ i = 1, \ldots, p \right. , \tag{5.3}$$

where $(\omega)_{\times}$ denotes the skew symmetric matrix associated with the cross product with vector $\omega \in \mathbb{R}^3$. The various white Gaussian continuous time noises can be stacked as

$$\mathbf{w} = \left[\mathbf{w}^{\omega T}\ \mathbf{w}^{\mathbf{a} T}\ \mathbf{w}^{\mathbf{b}^{\omega} T}\ \mathbf{w}^{\mathbf{b}^{\mathbf{a}} T} \right]^T, \tag{5.4}$$

where $\mathbf{w}$ is centered with autocorrelation $\mathcal{E}[\mathbf{w}(t)\mathbf{w}(s)] = \mathbf{W}\delta(t-s)$. These equations correspond to the equations of navigation, provided the earth is considered as locally flat. They model the dynamics of most of MAVs such as quadrotors where the IMU measurements ω and $\mathbf{a}$ in (5.1) are considered as noisy and biased inputs of the system.

2.2 2.2 Measurement Model

In addition to the IMU measurements used as input for the dynamics, the vehicle gets visual information from a calibrated monocular camera. It observes and tracks the p landmarks through a standard pinhole model [113]. Landmark $\mathbf{p}_i$ is observed through the camera as

$$\mathbf{y}_i = \frac{1}{y_w^i}\begin{bmatrix} y_i^u \\ y_i^v \end{bmatrix} + \mathbf{n}_i, \tag{5.5}$$

where $\mathbf{y}_i$ is the measured normalized pixel location of the landmark in the camera frame, that is,

$$\begin{bmatrix} y_i^u \\ y_i^v \\ y_w^i \end{bmatrix} = (\mathbf{R}^C)^T \left((\mathbf{R}^{\text{IMU}})^T (\mathbf{p}_i - \mathbf{p}^{\text{IMU}}) - \mathbf{p}^C \right), \tag{5.6}$$

where the right term corresponds to the distance from the landmark to the camera expressed in the camera frame. $\mathbf{R}^C$ and $\mathbf{p}^C$ are the known rotation matrix and the translation mapping from the body frame onto the camera frame. Finally, $\mathbf{n}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{N})$ represents the pixel image noise.

2.3 2.3 Estimation/Fusion Problem

Our goal is to compute the probability distribution of the high dimensional system's state $(\mathbf{R}^{\text{IMU}}, \mathbf{p}^{\text{IMU}}, \mathbf{v}^{\text{IMU}}, \mathbf{b}^{\omega}, \mathbf{b}^{\mathbf{a}}, \mathbf{p}_1, \ldots, \mathbf{p}_p)$ defined through an initial Gaussian prior and the probabilistic dynamic model (5.1)-(5.3), *conditionally* on the visual landmarks measurements of the form (5.5) for $1 \leq i \leq p$. This is the standard probabilistic formulation of the visual 3D SLAM problem with an IMU.

3 3 Proposed Algorithms

To apply the methodology of UKF on Lie groups, the dynamics must first be discretized, and the state space must be (partly) embedded in a matrix Lie group.

3.1 3.1 Time Discretization

Equations (5.1) are standard navigation equations, and their discretization is well established. In this chapter, we implemented the method of pre-integration on manifolds of [84].

3.2 3.2 Lie Group Embedding

The state space can be partially embedded into a Lie group, by letting χ_n be the matrix of the group $G = SE_{2+p}(3)$ that represents the state variables $(\mathbf{R}, \mathbf{v}, \mathbf{p}, \mathbf{p}_1, \cdots, \mathbf{p}_p)$ at time step n through representation (2.35). Using this embedding, the state at time n can in turn be represented as $(\chi_n, \mathbf{b}_n)$, letting the bias vector be $\mathbf{b} = \left[\mathbf{b}^{\omega T}\ \mathbf{b}^{\mathbf{a}T}\right]^T \in \mathbb{R}^6$. The dispersion on χ_n can be encoded using the left uncertainty or the right one, leading to two alternative filters (see Section 4.2). In the following, we detail the Right-UKF-LG which adopts the right-equivariant uncertainties of χ_n and conventional additive uncertainties on the biases $\mathbf{b}$. We leave to the reader the derivation of the Left-UKF-LG, based upon left-equivariant uncertainties (4.5).

Algorithm 7: Left and Right UKF on Lie groups

Input: $\hat{\chi}, \hat{\mathbf{b}}, \mathbf{S} = \text{chol}(\mathbf{P}), \mathbf{u}, \mathbf{Q}, \mathbf{Y}, \mathbf{N}$;

1 $\hat{\chi}, \hat{\mathbf{b}}, \mathbf{S} \leftarrow \texttt{Propagation}\left(\hat{\chi}, \hat{\mathbf{b}}, \mathbf{S}, \mathbf{u}, \mathbf{Q}\right)$;

if *received measurement* **then**

2 $\quad \hat{\chi}, \hat{\mathbf{b}}, \mathbf{S} \leftarrow \texttt{Update}\left(\hat{\chi}, \hat{\mathbf{b}}, \mathbf{S}, \mathbf{Y}, \mathbf{N}\right)$;

end

Output: $\hat{\chi}, \hat{\mathbf{b}}, \mathbf{S}$;

3.3 3.3 Final Retained Model and Filter Architecture

Defining the input vector $\mathbf{u} = [\omega^T, \mathbf{a}^T]^T$, and gathering the results of the two preceding subsections, we obtain the following uncertainty representation and discrete time model associated to the Right-UKF-LG:

$$\text{uncertainty rep.} \begin{cases} \chi_n = \exp(\xi)\hat{\chi}_n \\ \mathbf{b}_n = \hat{\mathbf{b}}_n + \tilde{\mathbf{b}} \end{cases}, \begin{bmatrix} \xi \\ \tilde{\mathbf{b}} \end{bmatrix} \sim \mathcal{N}(\mathbf{0}, \mathbf{P}_n), \tag{5.7}$$

$$\text{dynamics} \left\{ \chi_n, \mathbf{b}_n = f(\chi_{n-1}, \mathbf{u}_n - \mathbf{b}_{n-1}, \mathbf{w}_n), \right. \tag{5.8}$$

$$\text{observations} \begin{cases} \mathbf{Y}_n = \left[\mathbf{y}_1^T \cdots \mathbf{y}_p^T\right]^T := \mathbf{Y}(\chi_n, \mathbf{n}_n) \\ \quad \mathbf{y}_i \text{ given in (5.5), } i = 1, \ldots, p \end{cases}, \tag{5.9}$$

such that $\left(\hat{\chi}_n, \hat{\mathbf{b}}_n\right) \in \mathbb{R}^{15+3p}$ represents the mean (estimated) state at time n, $\mathbf{P}_n \in \mathbb{R}^{(15+3p)\times(15+3p)}$ is the covariance matrix that defines the state uncertainties $\left(\xi, \tilde{\mathbf{b}}\right)$, and the vector $\mathbf{Y}_n$ contains the observations of the p landmarks with associated discrete Gaussian noise $\mathbf{w}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$. The filter consists of two steps: propagation and update; as shown in Algorithm 7. We detail these two main steps in the following with the formalism of square-root implementation [114] where $\mathbf{S}$ is the Cholesky decomposition of the covariance, sparing the computation of covariance matrices and being numerically more stable.

Remark 2. *for the Left-UKF-LG, we define* $\chi_n = \hat{\chi}_n \exp(\xi)$ *and substitute it in* (5.7). *This results in quite different filters, though. In particular, consistence properties for EKF SLAM are characteristics of the right-invariant formalism, see [10] and also [58,59].*

3.4 3.4 Propagation Step

The propagation step is described in Algorithm 8 and operates as follow. The filter first computes the propagated mean state, and then the $2J$ sigma points obtained at line 5 are propagated at lines 6-7. It is then convenient to view the propagated Cholesky factors $\mathbf{S}$ as an output of the function $\text{qr}(\cdot)$. Details are provided in Appendix A along with the definitions of J and γ. Although more details on the methodology can be found in [60] (see also [86] regarding propagation), line 7 deserves a few explanations. According to uncertainty model, dispersion around the mean is modeled as $\exp(\xi)\chi$, so if $\hat{\chi}$ denotes the propagated mean, and χ_j denotes a propagated sigma point, then the corresponding sigma point in the Lie algebra is defined through $\exp(\xi_j)\hat{\chi} = \chi_j$, i.e., $\xi_j = \log\left(\chi_j \hat{\chi}^{-1}\right)$.

Algorithm 8: `Propagation` function for Right-UKF-LG

Input: $\hat{\chi}, \hat{\mathbf{b}}, \mathbf{S}, \mathbf{u}, \mathbf{Q}$;

$\mathbf{u} \leftarrow \mathbf{u} - \hat{\mathbf{b}}$; `// unbiased input`

$\mathbf{S}^a = \mathrm{blkdiag}(\mathbf{S}, \mathrm{chol}(\mathbf{Q}))$;

$\chi = \hat{\chi}$; `// save non propagated state`

$\hat{\chi}, \hat{\mathbf{b}} = f(\chi, \mathbf{u}, \mathbf{0})$; `// propagate mean`

`// step 5: sigma points generation`

$\left[\xi_j^{\mp}\ \mathbf{b}_j^{\mp}\ \mathbf{n}_j^{\mp}\right] = \mp\gamma\,\mathrm{col}_j(\mathbf{S}^a),\ j = 1,\ldots,J$;

`// steps 6-7: sigma point propagation`

$\chi_j^{\mp}, \mathbf{b}_j^{\mp} \leftarrow f\left(\exp\left(\xi_j^{\mp}\right)\chi, \mathbf{u} - \mathbf{b}_j^{\mp}, \mathbf{n}_j^{\mp}\right),\ j = 1,\ldots,J$;

$\xi_j^{\mp} \leftarrow \log\left(\chi_j^{\mp}\hat{\chi}^{-1}\right),\ j = 1,\ldots,J$;

$\mathbf{S} \leftarrow \mathrm{qr}\left(\xi_j^{\mp}, \mathbf{b}_j^{\mp}, j = 1,\ldots,J, \mathbf{Q}\right)$;

`// see Appendix for definition of qr`

`// the notation` $\mathbf{p}^{\mp}$ `is used to denote the two variables` $+\mathbf{p}$ `and` $-\mathbf{p}$

Output: $\hat{\chi}, \hat{\mathbf{b}}, \mathbf{S}$;

Algorithm 9: `Update` function for the Right-UKF-LG

Input: $\hat{\chi}, \hat{\mathbf{b}}, \mathbf{S}, \mathbf{Y}, \mathbf{N}$;

1 $\mathbf{Y}_0 = \mathbf{Y}(\hat{\chi}, \mathbf{0})$; `// see (5.9) and (5.5)-(5.6)`

2 $\left[\xi_j^{\mp}\ \mathbf{b}_j^{\mp}\right] = \mp\gamma\,\mathrm{col}_j(\mathbf{S}),\ j = 1,\ldots,J'$;

3 $\chi_j^{\mp} = \exp\left(\xi_j^{\mp}\right)\hat{\chi},\ j = 1,\ldots,J'$;

4 $\mathbf{Y}_j = \mathbf{Y}\left(\chi_j^{\mp}, \mathbf{w}_j^{\mp}\right),\ j = 1,\ldots,J'$;

5 $\delta\bar{\xi}, \delta\hat{\mathbf{b}}, \mathbf{S} \leftarrow \mathrm{qr}'\left(\mathbf{Y}_n, \mathbf{Y}_0, \mathbf{Y}_j^{\mp}, \xi_j^{\mp}, j = 1,\ldots,J', \mathbf{N}\right)$;

6 $\hat{\chi} \leftarrow \exp\left(\delta\bar{\xi}\right)\hat{\chi}, \hat{\mathbf{b}} \leftarrow \hat{\mathbf{b}} + \delta\hat{\mathbf{b}}$; `// update mean`

`// See Appendix for definition of qr'`

Output: $\hat{\chi}, \hat{\mathbf{b}}, \mathbf{S}$;

3.5 3.5 Update Step

The update step incorporates the observation of the p landmarks at time n and is described in Algorithm 9. It operates as follow. The sigma points generated in the Lie algebra at line 2 are mapped to the group at line 3, and used to compute $2J'+1$ measurement sigma points at line 4. The function $\mathrm{qr}'(\cdot)$ then evaluates the updated Cholesky factors and the correction term $\left(\delta\bar{\xi}, \delta\hat{\mathbf{b}}\right)$ used to update the mean state, along the lines of the conventional UKF methodology, and it is detailed in the Appendix A. Line 6 is the update of [60] as concerns the Lie group part of the state, and conventional update as concerns the biases, see next subsection for more details.

Remark 3. *following [115], the square-root implementation can add or remove landmarks, initializing landmark position as inverse depth point and allows computationally efficient propagation steps.*

3.6 3.6 Discussion on the Final Update

Let $\hat{\chi}$ denote the *propagated* mean, and $\mathbf{P} = \mathbf{S}\mathbf{S}^T$ the *propagated* covariance matrix of the state error after the propagation step, i.e. the outputs of Algorithm 7. According to

right uncertainty model (4.20), it means the propagated state is described by

$$\chi \approx \exp(\xi)\hat{\chi},\ \xi \sim \mathcal{N}(\mathbf{0},\mathbf{P})$$

before measurement $\mathbf{Y}_n$. At the update step, the UKF methodology takes into account the observation $\mathbf{Y}_n$ to update the element $\xi \in \mathbb{R}^{9+3p}$ as $\xi \sim \mathcal{N}(\delta\bar{\xi},\mathbf{P}^+)$, i.e., $\xi = \delta\bar{\xi} + \xi^+$ with $\xi^+ \sim \mathcal{N}(\mathbf{0},\mathbf{P}^+)$. Back to the Lie group this implies

$$\chi \approx \exp\left(\xi^+ + \delta\bar{\xi}\right)\hat{\chi},\ \text{where } \xi^+ \sim \mathcal{N}(\mathbf{0},\mathbf{P}^+). \tag{5.10}$$

Following [60], and assuming both correction $\delta\bar{\xi}$ and uncertainty ξ^+ in (5.10) are small, we end up with the following posterior that matches with the uncertainty representation (4.20):

$$\chi \simeq \exp(\xi^+)\hat{\chi}^+,\ \text{where} \tag{5.11}$$

$$\xi^+ \sim \mathcal{N}(\mathbf{0},\mathbf{P}^+), \tag{5.12}$$

$$\hat{\chi}^+ = \exp\left(\delta\bar{\xi}\right)\hat{\chi}. \tag{5.13}$$

This approximation is based indeed upon the Baker-Campbell-Hausdorff (BCH) formula that states that $\exp\left(\xi^+ + \delta\bar{\xi}\right) = \exp(\xi^+)\exp\left(\delta\bar{\xi}\right) + O\left(\delta\bar{\xi}^+, (\xi^+)^2, \xi^+\delta\bar{\xi}\right)$.

Remark 4. *when the correction terms are large and the BCH based approximation does not hold true, we propose an alternative method in Appendix B.*

4 4 Simulation Results

Five different filters are compared on Monte-Carlo simulations:

- an UKF that considers the attitude as an element $SO(3)$ and the remaining variables as a vector space;
- the $SE(3)$-based UKF recently introduced in [80]. This filter is an UKF which encodes body attitude and position in $SE(3)$ and uses parallel transport associated to left-invariant vector fields of $SE(3)$;
- the Right-Invariant visual EKF SLAM (RIEKF) of [10,59] (where the biases are appended to the state and treated as in the conventional EKF);
- the proposed Right-UKF-LG described in Section 3;
- the proposed Left-UKF-LG, as an alternative to Right-UKF-LG based on the left uncertainty representation (4.5).

4.1 4.1 Simulation Setting

We generate a noise-free trajectory displayed on Figure 5.1. This trajectory is realistic since it is inspired by true quadrotor trajectories from [106]. Noises and slowly drifting small biases are added, and a standard deviation of 2 pixels is set for the observation noise. We define the number of landmarks in the state as $p = 60$ and at each observation, we observe a subset of 10 of these landmarks. 100 Monte Carlo simulations are then run.

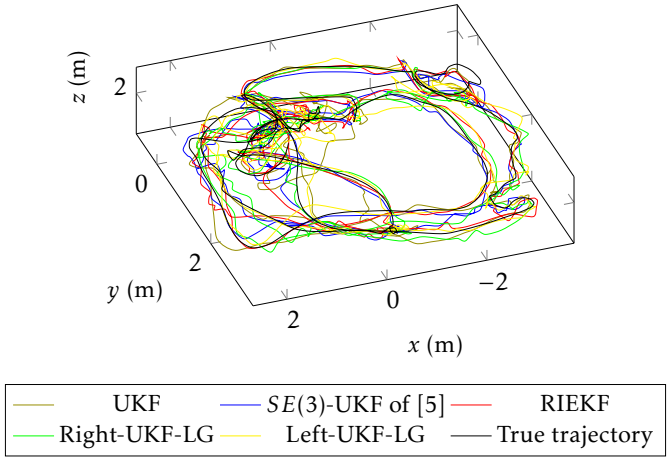


Figure 5.1: Simulation trajectory used in Section 4, and trajectories estimated by the various filters.

	p (cm)	**R** (°)
Conventional UKF	9.3	1.3
$SE(3)$-UKF of [80]	7.8	1.2
Left-UKF-LG	7.5	1.2
RIEKF	6.8	1.1
Right-UKF-LG	6.7	1.1

Figure 5.2: Root Mean Squared Error averaged on 100 Monte Carlo simulations, on the body position and attitude, for the various filters, as described in Section 4. The proposed Right-UKF-LG and RIEKF achieve the best results.

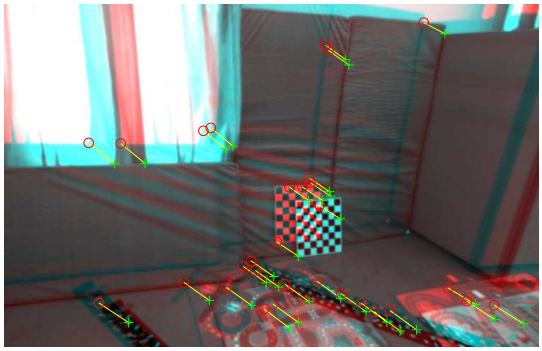


Figure 5.3: Landmark tracking in the experiment. Green crosses are the current pixel locations of the landmarks and red circles are the pixel locations of the landmarks five images (i.e., 1 s) earlier. Picture comes from the EuRoC dataset [106].

4.2 4.2 Results

The Root Mean Squared Error (RMSE) for the entire trajectory, averaged over 100 Monte Carlo runs, is displayed in Figure 5.2. From these results, we observe that:

- three groups appear: the RIEKF and Right-UKF-LG achieve the best results. This confirms that the right-invariant errors on $SE_{2+p}(3)$ are best suited to SLAM, as explained in [10,58]. Then, the Left-UKF-LG and the $SE(3)$-based UKF of [80] run second, and the conventional UKF runs last;
- The discrepancy between RIEKF and Right-UKF-LG is low at this noise level. Both algorithms are based on the right-invariant error on the Lie group $SE_{2+p}(3)$, but the first one uses the EKF methodology and the second one the UKF methodology.

5 5 Experimental Results

To further validate then the two proposed filters (Right and Left UKF-LG) on real data, we evaluate them on the EuRoC dataset [106]. The five compared filters are the same as in the previous section. We selected five sequences in [106] in which landmarks can be well tracked in order to minimize the influence of the frontend image processor.

5.1 5.1 Experimental Setting

Owing to the number of landmarks that keeps growing, the state may grow unboundedly and the filters become intractable for real time implementation. We thus propose to marginalize out landmarks that are not seen anymore, and add new landmarks to the state as they arrive, along the lines of [80]. This way, we conserve a constant number of 30 observed landmarks in the state, and the experimental results to come can be viewed as preliminary regarding our visual SLAM algorithm.

In our implementation, the filter tracks features via the KLT tracker using minimum eigenvalue feature detection [116] for its efficiency, and points are undistorted with the furnished camera parameters. The different filters are configured with the

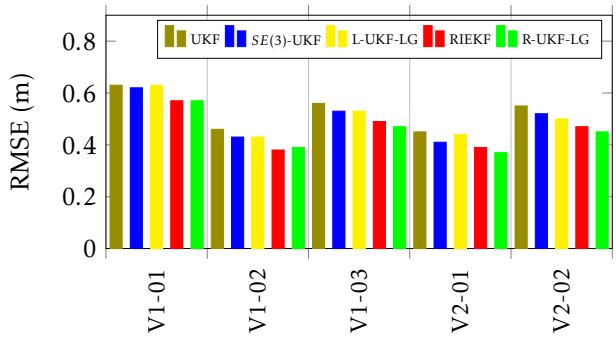


Figure 5.4: Root Mean Squared Error on position with respect to ground truth, on five different experiments.

same parameters, where we set 2 pixels standard deviation for the landmark observations and IMU noise provided by [106]. The initial state corresponds to the ground truth.

5.2 5.2 Results

The different filters are thus launched on the real data and we plot the position errors with respect to ground truth for five experiments in Figure 5.4. On this set of experiments, we see that as for the previous section, two groups appear: the RIEKF and Right-UKF-LG achieve the best results. However, the differences between the approaches are smaller than in the simulation section. This is mainly due to the small time presence of each landmark in the state, such that the different error representations have less influence on the results. This preliminary experiment confirms the potential of Right-UKF-LG and RIEKF over conventional UKF and $SE(3)$-UKF.

5.3 5.3 Comparison of Execution Times

We compare in this section the execution time of the filters, both for the propagation and update steps. Figure 5.5 summarizes the results (frontend execution time is excluded). From this table, we observe that

- UKFs approaches require much more computational power than RIEKF during the propagation step. This is reinforced by the computation of logarithm at line 7 of Algorithm 2;
- the propagation necessitates much more calculus than the update for each UKF solutions, since the IMU (propagation) frequency (200 Hz) is ten times the camera (update) frequency (20 Hz);
- the differences between UKF-based approaches and RIEKF solutions for the update step appear as negligible compared to the propagation step.

Note that, the various UKFs' propagation step seems particularly long, owing in part to the non-optimal use of Matlab. However, an alternative solution we advocate for low powerful devices is merely to implement an hybrid R-UKF-LG that combines the RIEKF propagation and the R-UKF-LG update, in which we preserve the versatility (and fast prototyping benefits) of the Unscented approach with respect to the addition

	propagation (s)	update (s)
Conventional UKF	28	2.5
$SE(3)$-UKF of [80]	33	4.2
Left-UKF-LG	35	3.4
RIEKF	2.1	2.2
Right-UKF-LG	36	3.4

Figure 5.5: Execution times of the different filters, both for the propagation and the update step. The indicated times correspond to the time execution of all sequences averaged for 10 s of flight.

of other sensors' measurements (such as GPS) and/or variations in the measurement camera model. We implemented this solution, and we obtained similar results as the (full) R-UKF-LG on those datasets. Finally, in practice, we note that the front image processing is anyway generally much higher than the execution time of the filter.

6 6 Conclusion

Two novel UKFs for data fusion of inertial sensors and monocular vision in the context of visual SLAM have been proposed. They build upon the very recent theory of UKF on Lie groups of [60], and have the merit of exploiting the full Lie group structure underlying the SLAM problem introduced in [9,10]. Another advantage is that the UKF approach spares the user the computation of Jacobians inherent to EKF implementation, and thus the proposed filters can be readily adapted to small modifications in the model, estimation of additional parameters, and/or addition of one or several sensors. Results from simulations and real experimental data have shown the relevance of the approach based on invariance, and notably the Right-UKF-LG.

Appendix A

We give here the details of parameters and functions used for L-UKF-LG and R-UKF-LG. We set the scale parameters γ and γ' with the scaled unscented transform [117], such that they depend on the augmented covariance size $J = 27 + 3p$, $J' = 15 + 3p$ and

$$\gamma = \sqrt{J/(1-W_0)}, W_0 = 1 - J/3, W_j = \frac{1-W_0}{2J}, \tag{5.14}$$

$$\gamma' = \sqrt{J'/(1-W_0')}, W_0' = 1 - J'/3, W_j' = \frac{1-W_0'}{2J'}. \tag{5.15}$$

The function $\mathrm{qr}(\cdot)$ operates as taking the $\mathcal{QR}$ decomposition of

$$\mathcal{QR} \leftarrow \sqrt{W_1}\begin{bmatrix} \xi_1^+ & \cdots & \xi_J^+ & \xi_1^- & \cdots & \xi_J^- \\ \mathbf{b}_1^+ & \cdots & \mathbf{b}_J^+ & \mathbf{b}_1^- & \cdots & \mathbf{b}_J^- \\ \mathbf{0} & \mathrm{chol}(\mathbf{Q}) & & \mathbf{0} & -\mathrm{chol}(\mathbf{Q}) & \end{bmatrix}, \tag{5.16}$$

from which we can extract the Cholesky factor as

$$\mathcal{R} = \begin{bmatrix} \mathbf{S} \\ \mathbf{0} \end{bmatrix}. \tag{5.17}$$

The function $\mathrm{qr}'(\cdot)$ operates as follow: first, compute the mean measurement and weighted deviation

$$\bar{\mathbf{Y}} = W_0'\mathbf{Y}_0 + \sum_{j=1}^{J'} W_i'\left(\mathbf{Y}_j^+ + \mathbf{Y}_j^-\right), \tag{5.18}$$

$$\mathbf{e}_0 = \sqrt{|W_0'|}\left(\mathbf{Y}_0 - \bar{\mathbf{Y}}\right), \tag{5.19}$$

$$\mathbf{e}_j^{\mp} = \sqrt{W_j'}\left(\mathbf{Y}_j^{\mp} - \bar{\mathbf{Y}}\right),\ j = 1,\ldots,J', \tag{5.20}$$

and compute the Cholesky factors of the measurement covariance and the cross covariance as

$$\mathcal{QR} \leftarrow \begin{bmatrix} \mathbf{e}_1^+ & \cdots & \mathbf{e}_{J'}^+ & \mathbf{e}_1^- & \cdots & \mathbf{e}_{J'}^- & \mathbf{R}' \end{bmatrix}, \tag{5.21}$$

$$\mathcal{R} = \begin{bmatrix} \mathbf{S}' \\ \mathbf{0} \end{bmatrix}, \tag{5.22}$$

$$\mathbf{S}' \leftarrow \texttt{CholUpdate}(\mathbf{S}', \mathrm{sign}(W_0'), \mathbf{e}_0), \tag{5.23}$$

$$\mathbf{P}' = \sum_{j=1}^{J'} \sqrt{W_j'}\left(\begin{bmatrix} \xi_j^+ \\ \mathbf{b}_j^+ \end{bmatrix}^T \mathbf{e}_j^+ + \begin{bmatrix} \xi_j^- \\ \mathbf{b}_j^- \end{bmatrix}^T \mathbf{e}_j^-\right), \tag{5.24}$$

$\mathbf{R}'$ is a block diagonal matrix containing p times the matrix $\mathrm{chol}(\mathbf{N})$ along its diagonal, and then compute gain, innovation and covariance as

$$\mathbf{K} = \mathbf{P}'\left(\mathbf{S}'^T\mathbf{S}'\right)^{-1} \tag{5.25}$$

$$= \mathbf{P}'\mathbf{S}'^{-1}\mathbf{S}'^{-T} \tag{5.26}$$

$$\begin{bmatrix} \delta\bar{\xi} \\ \delta\hat{\mathbf{b}} \end{bmatrix} = \mathbf{K}\left(\mathbf{Y} - \bar{\mathbf{Y}}\right), \tag{5.27}$$

$$\mathbf{S} \leftarrow \texttt{SeqCholUpdate}\left(\mathbf{S}, -1, \mathbf{K}\mathbf{S}'^T\right), \tag{5.28}$$

where $\texttt{SeqCholUpdate}$ denotes repeated Cholesky updating $\texttt{CholUpdate}$ using successive columns of $\mathbf{K}\mathbf{S}'^T$ as the updating vector [114]. To finally consider the Jacobian (see Section 3.6), we compute

$$\mathbf{S} \leftarrow \mathbf{S}\mathbf{J}^T, \tag{5.29}$$

letting $\mathbf{S}$ no longer triangular, but $\mathbf{S}$ keeps a valid matrix square root which could be used to define the next set of sigma points [115].

Remark 5. *since we consider observation* $\mathbf{Y}$ *that lives in vector space,* (5.27) *is always valid and we do not have to compute any logarithm operation.*

Appendix B

As concerns the update step, when the innovation $\delta\bar{\xi}$ is important, we propose to possibly use the more accurate approximation of [11,12]

$$\exp\left(\xi^+ + \delta\bar{\xi}\right) = \exp(\mathbf{J}\xi^+)\exp\left(\delta\bar{\xi}\right) + o(\xi^+), \tag{5.30}$$

where $\mathbf{J}$ is the left Jacobian. In this case we compute the updated parameters as

$$\hat{\chi}^{+} = \exp\left(\delta\bar{\xi}\right)\hat{\chi}, \tag{5.31}$$

$$\mathbf{P}^{+} \leftarrow \mathbf{J}\mathbf{P}^{+}\mathbf{J}^{T}, \tag{5.32}$$

When $\delta\bar{\xi}$ remains small, $\mathbf{J} \approx \mathbf{I}$ such that we can discard $\mathbf{J}$ in (5.32) for computational efficiency, recovering the update [60].

CHAPTER 6

Unscented Kalman Filter on Lie Groups for Visual Inertial Odometry

Résumé 🇫🇷

Dans ce chapitre, nous présentons un filtre innovant pour l'odométrie basée inertie et vision stéréo en s'appuyant sur: *i*) le filtre de Kalman à contraintes multi-états stéréo récemment introduit; *ii*) la théorie du filtrage invariant; et *iii*) le filtre de Kalman sans-parfum sur les variétés parallélisables (plus précisément les groupes de Lie). Notre solution allie précision, robustesse et polyvalence de l'UKF. Nous comparons ensuite notre approche des solutions de pointe en termes de précision, de robustesse et de complexité de calcul sur le jeu de données EuRoC et ainsi que sur un jeu de donnée acquis en extérieur difficile.

Chapter abstract

In this chapter, we present an innovative filter for stereo visual inertial odometry building on: *i*) the recently introduced stereo multi-state constraint Kalman filter; *ii*) the invariant filtering theory; and *iii*) the UKF on parallelizable manifolds (more precisely Lie groups). Our solution combines accuracy, robustness and versatility of the UKF. We then compare our approach to state-of-art solutions in terms of accuracy, robustness and computational complexity on the EuRoC dataset and a challenging MAV outdoor dataset.

1 1 Introduction

In this chapter, we tackle the problem of fusing IMU signals with stereo vision. To this respect the problem is similar to the previous chapter. However, there are two main differences. First, we use stereo vision. Indeed, adopting a stereo configuration provides higher robustness compared to the popular monocular configuration (but it comes at the cost of an additional sensor and the need for further calibration). Then, we adopt the VIO framework, where landmarks are not kept in the state. This is desirable indeed as keeping all the landmarks in the state makes the state variable grow indefinitely and sooner or later leads to an intractable problem. Because of this, VIO allows for efficient real-time implementation over arbitrary long time periods.

More precisely, we propose a novel algorithm whose implementation mainly builds on the very recent Stereo Multi-State Constraint Kalman Filter (S-MSCKF) [108]. In this chapter:

1. We benefit from the UKF methodology as compared to the standard EKF. Since the unscented transform spares the computation of Jacobians, the algorithm is versatile and allows fast prototyping in the presence variations in the model (e.g., the camera model).

2. We build upon the theory of the UKF on Lie Groups [60], and leverage the Lie group structure of the SLAM problem introduced in [10].

We demonstrate that the proposed stereo VIO filter is able to achieve similar or even higher accuracy than state-of-art solutions on two distinct datasets with high efficiency.

1.1 1.1 Contributions

Among popular visual inertial navigation solutions, the MSCKF developed by Mourikis and Roumeliotis [109] and its state-of-the-art variants, notably [46], offer an efficient compromise between accuracy and computational complexity, which was applied e.g. to the application of spacecraft descent and landing [118] and fast UAV autonomous flight [108].

In this chapter, we propose an UKF-based stereo VIO solution that leverages the MSCKF methodology and the Lie group structure of the state space $SE(3)_{2+p}$. Our main contributions are:

- an embedding of the state and the uncertainties into a matrix Lie group which additionally considers the unknown IMU to camera transformation;

- the derivation of a Kalman filter that combines an IEKF propagation for computational efficiency and an UKF-LG update, in which our choice is motivated by the UKF superiority performance compared with the EKF for many non-linear problems, and its ease of implementation for the practitioner, allowing him to readily handle additional measurements (such as GPS measurements) or variations in the output model (the camera model) since the update is derivative free;

- since the computational demands of a standard UKF update is generally greater than those of the EKF, we provide: i) a computationally efficient strategy for computing our UKF-LG update in the formalism of an IEKF update, inspired by [4,5]; and ii) a closed-form expression for alternatively performing the update in a full IEKF manner;

- the publicly available C++/ROS source code used for this chapter, available at `https://github.com/mbrossar/msckf_vio.git`. It uses building blocks from the code of [108].

Finally, the accuracy and computational complexity of the proposed filter is validated and compared with state-of-the-art VIO solutions on two challenging real-world MAV datasets [106,108].

1.2 1.2 Chapter's Organization

Section 2 formulates the filtering problem. Section 3 contains mathematical preliminaries for unscented Kalman filtering on Lie groups. Section 4 describes the proposed

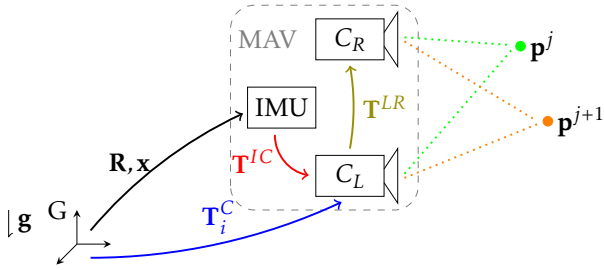


Figure 6.1: The coordinate systems that are used in the chapter. The IMU pose $(\mathbf{R}, \mathbf{p})$ maps vectors expressed in the IMU frame to the global frame (G). The transformations $\mathbf{T}^{IC}$ and $\mathbf{T}^{LR}$ pass, respectively, from the IMU frame to the left camera (C_L) frame, and from the left camera frame to the right camera (C_R) frame. The pose $\mathbf{T}_i^C$ consists of the position of C_L in the global frame and the rotation mapping vectors expressed in the C_L frame to vectors expressed in the global frame. Unknown 3D features $\mathbf{p}^j$ (expressed in the global frame) are tracked across a stereo camera system.

filter for stereo VIO. Section 5 illustrates the performances of the proposed filter based on two publicly available datasets.

2 2 Problem Modeling

We define in this section the kinematic model for flying devices equipped with an IMU where N past cloned camera poses make up the state [109]. We then detail the stereo visual measurement model, and we finally pose the filtering problem we seek to address.

2.1 2.1 Variables of Interest and Dynamical Model

Let us consider an aerial body navigating on flat earth equipped with an IMU. The dynamics of the system read

$$\text{IMU-related state} \begin{cases} \dot{\mathbf{R}}^{\text{IMU}} = \mathbf{R}^{\text{IMU}} (\boldsymbol{\omega} - \mathbf{b}^{\omega} + \mathbf{n}^{\omega})_{\times} \\ \dot{\mathbf{v}}^{\text{IMU}} = \mathbf{R}^{\text{IMU}} (\mathbf{a} - \mathbf{b}^{\mathbf{a}} + \mathbf{n}^{\mathbf{a}}) + \mathbf{g} \\ \dot{\mathbf{p}}^{\text{IMU}} = \mathbf{v}^{\text{IMU}} \\ \dot{\mathbf{b}}_{\omega} = \mathbf{n}^{\mathbf{b}^{\omega}} \\ \dot{\mathbf{b}}_{\mathbf{a}} = \mathbf{n}^{\mathbf{b}^{\mathbf{a}}} \end{cases} \tag{6.1}$$

$$\text{camera-related state} \begin{cases} \dot{\mathbf{T}}^{IC} = \mathbf{0} \\ \dot{\mathbf{T}}_i^C = \mathbf{0}, \ i = 1, \ldots, N \end{cases} \tag{6.2}$$

where the state we want to estimate consists of the current orientation $\mathbf{R}^{\text{IMU}} \in SO(3)$ of the body frame (referred to as the IMU frame), that is, the rotation matrix that maps the IMU frame to the global frame, velocity $\mathbf{v}^{\text{IMU}} \in \mathbb{R}^3$, position $\mathbf{p} \in \mathbb{R}^3$, IMU biases $\mathbf{b}^{\omega} \in \mathbb{R}^3$ and $\mathbf{b}^{\mathbf{a}} \in \mathbb{R}^3$, as well as the relative transformation between the IMU frame and the left camera frame $\mathbf{T}^{IC} \in SE(3)$ and an arbitrary number of N recorded left camera poses $\mathbf{T}_i^C \in SE(3)$, $i = 1, \ldots, N$, see Figure 6.1. Finally, $(\boldsymbol{\omega})_{\times}$ denotes the skew symmetric matrix associated with the cross product with vector $\boldsymbol{\omega} \in \mathbb{R}^3$, and the various white Gaussian noises can be stacked as

$$\mathbf{w} = \begin{bmatrix} \mathbf{w}^{\omega T} & \mathbf{w}^{\mathbf{a} T} & \mathbf{w}^{\mathbf{b}^{\omega} T} & \mathbf{w}^{\mathbf{b}^{\mathbf{a}} T} \end{bmatrix}^T \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}). \tag{6.3}$$

These equations model the dynamics of small MAVs such as quadrotors where the IMU measurements $\boldsymbol{\omega}$ and $\mathbf{a}$ in (6.1) are considered as noisy and biased inputs of the system.

2.2 2.2 Measurement Model

In addition to the IMU measurements used as inputs for the dynamics, the vehicle observes and tracks static landmarks in the global frame from a calibrated stereo camera. A landmark $\mathbf{p}^j \in \mathbb{R}^3$ is observed through both the left and right cameras corresponding to the recorded i-th pose as

$$\mathbf{y}_i^j = h\left(\mathbf{T}_i^C, \mathbf{p}^j\right) + \mathbf{n}^{\mathbf{y}} \in \mathbb{R}^4, \tag{6.4}$$

where the non-linear stereo measurement model $h(\cdot)$ is given as [108]

$$h(\mathbf{T}, \mathbf{p}) = \begin{bmatrix} \frac{1}{z_l}\mathbf{I} & \mathbf{0} \\ \mathbf{0} & \frac{1}{z_r}\mathbf{I} \end{bmatrix} \begin{bmatrix} x_l \\ y_l \\ x_r \\ y_r \end{bmatrix}, \tag{6.5}$$

in which $\mathbf{p}_l = [\,x_l \; y_l \; z_l\,]^T$ and $\mathbf{p}_r = [\,x_r \; y_r \; z_r\,]^T$ refer to the landmark coordinates expressed in the left and in the right camera frames (i.e., $[\,\mathbf{p}_l^T \; 1\,]^T = \mathbf{T}^{-1}[\,\mathbf{p}^{\text{IMU}T} \; 1\,]^T$ and $[\,\mathbf{p}_r^T \; 1\,]^T = \left(\mathbf{T}\mathbf{T}^{LR}\right)^{-1}[\,\mathbf{p}^{\text{IMU}T} \; 1\,]^T$). Note that the stereo cameras have different poses at the same time instance, represented as $\mathbf{T}_i^C$ for the left camera and $\mathbf{T}_i^C\mathbf{T}^{LR}$ for the right camera, although the state only contains the pose of the left camera, since using the assumed known extrinsic parameters $\mathbf{T}^{LR} \in SE(3)$ leads to an expression for the pose of the right camera, see Figure 6.1.

2.3 2.3 Estimation/Fusion Problem

We would like to compute the probability distribution of the system's state ($\mathbf{R}^{\text{IMU}}, \mathbf{p}^{\text{IMU}}, \mathbf{v}^{\text{IMU}}, \mathbf{b}^{\omega}, \mathbf{b}^{\mathbf{a}}, \mathbf{T}^{IC}, \mathbf{T}_1^C, \ldots, \mathbf{T}_N^C$) defined through an initial Gaussian prior and the probabilistic evolution model (6.1)-(6.2), *conditionally* on the measurements of the form (6.4). This is the standard probabilistic formulation of the (Stereo-)MSCKF [109].

3 3 Unscented Based Inferred Jacobian for UKF-LG Update

In [4], the authors interpret the conventional UKF as a linear regression Kalman filter and show how the propagation and update steps in UKF can be performed in a similar fashion as an EKF, which can save execution time [5]. The method is here straightforwardly adapted for the case of the Right-UKF-LG update. Within this interpretation, the filter seeks to find the optimal linear approximation to the nonlinear function

$$\mathbf{y} = h\left(\exp_G(\xi)\hat{\mathcal{X}}\right) = g(\xi) \tag{6.6}$$

$$\simeq \mathbf{H}\xi + \hat{\mathbf{y}} \tag{6.7}$$

given a weighted discrete representation (the so-called sigma-points) of the distribution $\xi \sim \mathcal{N}(\mathbf{0}, \mathbf{P})$. The objective is thus to find the regression matrix $\mathbf{H}$ and vector $\hat{\mathbf{y}}$ that minimize the linearization error $\mathbf{e} = \mathbf{y} - (\mathbf{H}\xi + \hat{\mathbf{y}})$. The optimal linear regression matrix is given as [4]

$$\mathbf{H} = \mathbf{P}_{\mathbf{y}\xi}\mathbf{P}^{-1}, \tag{6.8}$$

where $\mathbf{P}_{\mathbf{y}\xi}$ is the cross-correlation between $\mathbf{y}$ and ξ, and $\hat{\mathbf{y}}$ is the estimated mean of $\mathbf{y}$, both computed from the unscented transform of the UKF-LG. The numerically inferred Jacobian $\mathbf{H}$ serves as a linear approximation to $\mathbf{y}-\hat{\mathbf{y}} = g(\xi)-\hat{\mathbf{y}}$ and can then be used for the Right-EKF-LG update.

4 4 Proposed Filters

In this section, we propose the Stereo-UKF-LG (S-UKF-LG), a VIO filtering solution which embeds the state in a specially defined and high dimensional Lie group. Our solution operates in two steps, as for any Kalman filter-based algorithm:

- a propagation step that propagates both the mean state and the error covariance, where the matrix covariance is computed with IEKF linearization [42] for computational efficiency.
- an update step that considers the visual information obtained from the feature tracking, in which we used as a basis the UKF-LG [60]. We additionally provide Jacobian expressions to alternatively perform IEKF update.

4.1 4.1 State and Error Embedding on Lie Groups

Based on Chapter 4, we embed the state into a high dimensional Lie group, by letting $\chi \in G$ be the matrix that represents:

- the IMU variables $\mathbf{R}^{\text{IMU}}$, $\mathbf{v}^{\text{IMU}}$ and $\mathbf{p}^{\text{IMU}}$ through $\chi^I \in SE_2(3)$, a group obtained by letting $p = 0$ in (2.35);
- the IMU bias $\chi^{\mathbf{b}} = [\,\mathbf{b}^{\omega T}\ \mathbf{b}^{\mathbf{a}T}\,] \in \mathbb{R}^6$;
- the IMU to left camera transformation $\mathbf{T}^{IC} \in SE(3)$;
- the N left camera poses $\mathbf{T}_i^C \in SE(3),\ i = 1,\dots,N$.

The dispersion on the state

$$\chi^I, \chi^{\mathbf{b}}, \mathbf{T}^{IC}, \mathbf{T}_1^C, \dots, \mathbf{T}_N^C \triangleq \chi \sim \mathcal{N}_R(\hat{\chi}, \xi), \tag{6.9}$$

where $\triangleq$ stands for "identifiable to" and $(\bar{\cdot})$ for estimated mean value, is partitioned into

$$\xi = \begin{bmatrix} \xi_{\mathbf{R}}^T & \xi_{\mathbf{v}}^T & \xi_{\mathbf{p}}^T & \xi_{\omega}^T & \xi_{\mathbf{a}}^T & \xi_{IC}^T & \xi_{C_1}^T & \cdots & \xi_{C_N}^T \end{bmatrix}^T \tag{6.10}$$

and encoded using the right uncertainty (4.20), i.e., the uncertainty representation is defined as

$$\exp(\xi)\hat{\chi} \triangleq \begin{cases} \exp_{SE_2(3)}\left(\begin{bmatrix} \xi_{\mathbf{R}}^T & \xi_{\mathbf{v}}^T & \xi_{\mathbf{p}}^T \end{bmatrix}^T\right)\chi^I \\ \begin{bmatrix} \xi_{\omega}^T & \xi_{\mathbf{a}}^T \end{bmatrix}^T + \chi^{\mathbf{b}} \\ \exp_{SE(3)}(\xi_{IC})\mathbf{T}^{IC} \\ \exp_{SE(3)}\left(\xi_{C_i}\right)\mathbf{T}_i^C, i = 1,\dots,N \end{cases} \tag{6.11}$$

and we define for convenience the IMU error as $\xi_I = [\,\xi_{\mathbf{R}}^T\ \xi_{\mathbf{v}}^T\ \xi_{\mathbf{p}}^T\ \xi_{\omega}^T\ \xi_{\mathbf{a}}^T\,]^T$. Any unknown feature $\mathbf{p}^j$, albeit not explicitly considered in the state, appear in the measurement (6.4) and consequently we have to define an error on this feature. In the following we propose to identify each $\left(\mathbf{T}_1^C, \mathbf{p}^j\right)$ as a element of the Lie group $SE_2(3)$ [10]. Note that, error $\xi_{\mathbf{p}^j}$ on landmark j then differs from the standard Euclidean error.

Remark 6. *Using another camera pose than* $\mathbf{T}_1^C$ *does not influence the performances of the filter in our experiments.*

4.2 4.2 Propagation Step

Let us now present the proposed filter's mechanics. To deal with discrete time measurement from the IMU, we essentially proceed along the lines of [109]. We apply a 4-th order Runge-Kutta numerical integration of the model dynamic (6.1) to propagate the estimated state $\hat{\chi}$. To propagate the uncertainty of the state, let us consider the dynamic of the IMU linearized error as

$$\dot{\xi}_I = \mathbf{F}\xi_I + \mathbf{G}\mathbf{n}, \tag{6.12}$$

where

$$\mathbf{F} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathbf{R}^{\text{IMU}} & \mathbf{0} \\ \mathbf{g}_\times & \mathbf{0} & \mathbf{0} & -\mathbf{v}_\times\mathbf{R}^{\text{IMU}} & -\mathbf{R}^{\text{IMU}} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} & -\mathbf{p}_\times\mathbf{R}^{\text{IMU}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}, \tag{6.13}$$

$$\mathbf{G} = \begin{bmatrix} \mathbf{0} & \mathbf{R}^{\text{IMU}} & \mathbf{0} & \mathbf{0} \\ \mathbf{R}^{\text{IMU}} & \mathbf{v}_\times\mathbf{R}^{\text{IMU}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{p}_\times\mathbf{R}^{\text{IMU}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix}, \tag{6.14}$$

and first compute the discrete time state transition matrix

$$\mathbf{\Phi}_n = \mathbf{\Phi}(t_{n+1}, t_n) = \exp_m\left(\int_{t_n}^{t_{n+1}} \mathbf{F}(\tau)d\tau\right) \tag{6.15}$$

and discrete time noise covariance matrix

$$\mathbf{Q}_n = \int_{t_n}^{t_{n+1}} \mathbf{\Phi}(t_{n+1}, \tau)\mathbf{G}\mathbf{Q}\mathbf{G}^T\mathbf{\Phi}(t_{n+1}, \tau)^T\, d\tau. \tag{6.16}$$

The covariance matrix from t_n to t_{n+1} is propagated as the combination of partitioned covariance matrix as follows. The propagated covariance of the IMU state becomes

$$\mathbf{P}_{n+1}^{II} = \mathbf{\Phi}_n\mathbf{P}_n^{II}\mathbf{\Phi}_n + \mathbf{Q}_n, \tag{6.17}$$

and the full uncertainty propagation can be computed as

$$\mathbf{P}_{n+1} = \begin{bmatrix} \mathbf{P}_{n+1}^{II} & \mathbf{\Phi}_n\mathbf{P}_n^{IC} \\ \mathbf{P}_n^{CI}\mathbf{\Phi}_n^T & \mathbf{P}_n^{CC} \end{bmatrix}. \tag{6.18}$$

When new images are received, the state should be augmented with the new camera state. The new augmented covariance matrix becomes

$$\mathbf{P}_{n+1} = \begin{bmatrix} \mathbf{I} \\ \mathbf{J} \end{bmatrix} \mathbf{P}_{n+1} \begin{bmatrix} \mathbf{I} \\ \mathbf{J} \end{bmatrix}^T. \tag{6.19}$$

To obtain the expression of $\mathbf{J}$ in (6.19), let us denote $\mathbf{T} \in SE(3)$ as the sub-matrix of χ^I that contain only the IMU pose $(\mathbf{R}, \mathbf{p})$ with $\xi_{\mathbf{T}} = [\, \xi_{\mathbf{R}}^T \; \xi_{\mathbf{p}}^T \,]^T$ its corresponding uncertainties and thus write the current camera pose as

$$\begin{aligned}
\mathbf{T}^C &= \mathbf{T}\mathbf{T}^{IC} \\
&= \exp_{SE(3)}(\xi_{\mathbf{T}})\bar{\mathbf{T}}\exp_{SE(3)}(\xi_{IC})\bar{\mathbf{T}}^{IC} \\
&= \exp_{SE(3)}(\xi_{\mathbf{T}})\exp_{SE(3)}(\mathrm{Ad}_{\bar{\mathbf{T}}}\,\xi_{IC})\bar{\mathbf{T}}\bar{\mathbf{T}}^{IC} \\
&\simeq \exp_{SE(3)}(\xi_C)\bar{\mathbf{T}}^C,
\end{aligned} \tag{6.20}$$

in which $\bar{\mathbf{T}}^C = \bar{\mathbf{T}}\bar{\mathbf{T}}^{IC}$ and $\xi_C \simeq \xi_{\mathbf{T}} + \mathrm{Ad}_{\bar{\mathbf{T}}}\,\xi_{IC}$ after using a first order BCH approximation, and where $\mathrm{Ad}_{\bar{\mathbf{T}}}$ is the adjoint notation of $SE(3)$, which finally leads to

$$\mathbf{J} = \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0}_{3\times 6} & \mathbf{R}^{\text{IMU}} & \mathbf{0} & \mathbf{0} & \mathbf{0}_{3\times 6N} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0}_{3\times 6} & \mathbf{p}_\times\mathbf{R}^{\text{IMU}} & \mathbf{0} & \mathbf{R}^{\text{IMU}} & \mathbf{0}_{3\times 6N} \end{bmatrix}. \tag{6.21}$$

4.3 4.3 Update Step in the MSCKF Methodology

Let us first consider the observation of a single feature $\mathbf{p}^j$, in which the estimated unbiased feature position ${}^{\text{LS}}\mathbf{p}^j$ is computed using least squares estimate based on the current estimated camera poses [109]. Linearizing the measurement model at the obtained estimates $\hat{\chi}$, ${}^{\text{LS}}\mathbf{p}^j$, the residual of the measurement is approximated as

$$\mathbf{r}_i^j = \mathbf{y}_i^j - {}^{\text{UKF}}\mathbf{y}_i^j = {}^{\text{UKF}}\mathbf{H}_i^j\xi + \mathbf{H}_{\mathbf{p}^j}^i\xi_{\mathbf{p}^j} + \mathbf{n}_i^j, \tag{6.22}$$

where ${}^{\text{UKF}}\mathbf{H}_i^j$, $\mathbf{H}_{\mathbf{p}^j}^i$ and ${}^{\text{UKF}}\mathbf{y}_i^j$ are computed in Section 4.4 and are *not* the usual Jacobians appearing in [109] (beyond the fact they are computed using the unscented transform) since we use here alternative state errors ξ, $\xi_{\mathbf{p}^j}$ related to the Lie group structure we have endowed the state space with. By stacking multiple observations of the same feature $\mathbf{p}^j$, we then dispose of

$$\mathbf{r}^j = \mathbf{y}^j - {}^{\text{UKF}}\mathbf{y}^j = {}^{\text{UKF}}\mathbf{H}^j\xi + \mathbf{H}_{\mathbf{p}^j}\xi_{\mathbf{p}^j} + \mathbf{n}^j. \tag{6.23}$$

Then, along the lines of [109], to eliminate the landmark errors from the residual, measurements are projected onto the null space $\mathbf{V}$ of $\mathbf{H}_{\mathbf{p}^j}^i$, i.e.,

$$\mathbf{r}_o^j = \mathbf{V}^T\mathbf{r}^j = \mathbf{V}^T\,{}^{\text{UKF}}\mathbf{H}^j\xi + \mathbf{V}^T\mathbf{n}^j = \mathbf{H}_o^j\xi + \mathbf{n}_o^j. \tag{6.24}$$

Based on (6.24) and after stacking residual and Jacobians for multiple landmarks in, respectively, $\mathbf{r}_o$ and $\mathbf{H}_o$, the update step is carried out by first computing $\mathbf{S} = \mathbf{R}^{\text{IMU}} + \mathbf{H}_o\mathbf{P}_n\mathbf{H}_o^T$ and the gain matrix $\mathbf{K} = \mathbf{P}_n\mathbf{H}_o^T/\mathbf{S}$. We then compute the innovation to update the mean state as [42,60]

$$\hat{\chi}^+ = \exp(\mathbf{K}\mathbf{r}_o)\hat{\chi}, \tag{6.25}$$

which is an update that is consistent with our uncertainties (6.11) defined using the right multiplication based representation (4.20). The associated covariance matrix writes

$$\mathbf{P}_n^+ = \mathbf{P}_n(\mathbf{I} - \mathbf{K}\mathbf{H}_o). \tag{6.26}$$

The filter concludes the update step with a marginalization of the last camera pose.

4.4 4.4 Proposed Lie Group Based Update

In this section, we describe the computation of ${}^{\mathsf{UKF}}\mathbf{H}_i^j$, $\mathbf{H}_{\mathbf{p}^j}^i$ and ${}^{\mathsf{UKF}}\mathbf{y}_i^j$ in (6.22) following Section 3.

Computation of $\mathbf{H}_{\mathbf{p}^j}^i$: since the covariance of $\xi_{\mathbf{p}}^j$ is unknown, we can not apply UKF-LG update and consequently we compute $\mathbf{H}_{\mathbf{p}^j}^i$ in closed-form, as $\mathbf{H}_{\mathbf{p}^j}^1 = \mathbf{0}$, and for $i > 1$, as

$$\mathbf{H}_{\mathbf{p}^j}^i = \mathbf{J}_i^j \begin{bmatrix} \left(\bar{\mathbf{R}}_i^L\right)^T \\ \left(\bar{\mathbf{R}}_i^R\right)^T \end{bmatrix}, \tag{6.27}$$

where $\bar{\mathbf{R}}_i^L$ and $\bar{\mathbf{R}}_i^R$ are the orientations of the i-th left and right cameras, and where

$$\mathbf{J}_i^j = \begin{bmatrix} 1/\bar{z}_l & 0 & -\bar{x}_l/\bar{z}_l^2 \\ 0 & 1/\bar{z}_l & -\bar{\mathbf{y}}_l/\bar{z}_l^2 \\ 1/\bar{z}_r & 0 & -\bar{x}_r/\bar{z}_r^2 \\ 0 & 1/\bar{z}_r & -\bar{\mathbf{y}}_r/\bar{z}_r^2 \end{bmatrix} \tag{6.28}$$

is the intermediate Jacobian after applying the chain rule in (6.4). We stress that even if (6.27) is the same expression as the Jacobians appearing in [109], they are associated to the alternative state errors $\xi_{\mathbf{p}^j}$.

Computation of ${}^{\mathsf{UKF}}\mathbf{H}_i^j$ and ${}^{\mathsf{UKF}}\mathbf{y}_i^j$: we apply the UKF-LG [60] to numerically infer the Jacobian ${}^{\mathsf{UKF}}\mathbf{H}_i^j$ and estimated measurement ${}^{\mathsf{UKF}}\mathbf{y}_i^j$ (see Section 3). This computation allows us to then project the residual in (6.24) and can be computed efficiently as follow. First, the filter stacks multiple observations w.r.t. the same camera pose in a vector $\mathbf{y}_i$ that can be written in the form $\mathbf{y}_i = g_i\left(\xi_{C_1}, \xi_{C_i}\right)$, when noise and landmark errors are marginalized, as follow. Since $\mathbf{y}_i$ is the concatenation of (6.4) for multiple landmarks, it is sufficient to write $\mathbf{y}_i^j$ as a function of ξ_{C_1} and ξ_{C_i} only. Let us first write our alternative state error as

$$\exp_{SO(3)}(\xi_{\mathbf{R}_{C_i}}) = \mathbf{R}_{C_i}\bar{\mathbf{R}}_{C_i}^T, \tag{6.29}$$

$$\xi_{\mathbf{p}_{C_i}} = \mathbf{R}_{C_i}\bar{\mathbf{R}}_{C_i}^T\mathbf{p}_{C_i} - \bar{\mathbf{x}}_{C_i}, \tag{6.30}$$

$$\xi_{\mathbf{p}^j} = \mathbf{R}_{C_1}\bar{\mathbf{R}}_{C_1}^T\mathbf{p}^j - {}^{\mathsf{LS}}\mathbf{p}^j, \tag{6.31}$$

and consider the measurement function of the stereo camera such that

$$\mathbf{y}_i^j = l\left(\mathbf{R}_{C_i}^T\left(\mathbf{p}^j - \mathbf{p}_{C_i}\right)\right), \tag{6.32}$$

where $l(\cdot)$ is the projection function. After inserting the uncertainties (6.29)-(6.31) in (6.32), we obtain

$$\begin{aligned} \mathbf{y}_i^j = l(\bar{\mathbf{R}}_{C_i}^T(\exp_{SO(3)}(\xi_{\mathbf{R}_{C_i}})\exp_{SO(3)}(-\xi_{\mathbf{R}_{C_1}})^{\mathsf{LS}}\mathbf{p}^j \\ -\bar{\mathbf{R}}_{C_i}^T(\mathbf{p}_{C_i} + \xi_{\mathbf{p}_{C_i}})), \end{aligned} \tag{6.33}$$

which depends on $\xi_{\mathbf{R}_{C_1}}$ and $\xi_{C_i} = [\,\xi_{\mathbf{R}_{C_i}}\ \xi_{\mathbf{p}_{C_i}}\,]$ only, such that we can write $\mathbf{y}_j^i$ and by extension $\mathbf{y}^i$ as function of $\xi_{\mathbf{R}_{C_1}}$ and ξ_{C_i} only. Following then [5], we sample only sigma-points from variables the observations depend on, i.e., from the rotational part of ξ_{C_1} and ξ_{C_i}, such that the complexity remains dominated by (6.26) and comparable to the S-MSCKF, which is illustrated in Section 5.

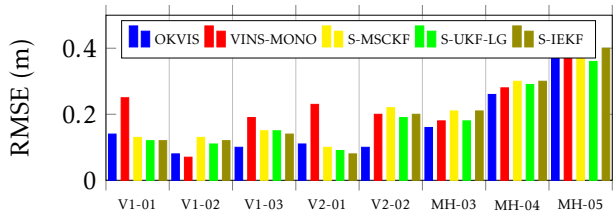


Figure 6.2: Root Mean Square Error of the proposed S-UKF-LG and S-IEKF compared to various methods on the EuRoC dataset [106]. Statistics are averaged over ten runs on each sequence.

4.5 4.5 Alternative IEKF Update

Albeit the UKF-LG update is computed efficiently, an IEKF update remains slightly more computationally efficient and can be adopted for computationally restricted platforms. Thus, as an alternative, we provide the closed-form expression for an IEKF update, yielding a stereo extension of [59], and where moreover the IMU to camera transformation is also estimated. The mean $\mathbf{y}_i^j = \left(\bar{\mathbf{T}}_i^C, \tilde{\mathbf{p}}^j\right)$ is alternatively computed with estimated state, $\mathbf{H}^i_{\mathbf{p}^j}$ in (6.27) and $\mathbf{H}_i^j$ as

$$\mathbf{H}_i^j = \mathbf{J}_i^j \begin{bmatrix} \mathbf{0} & \mathbf{H}^j_{C_1} & \mathbf{0} & \mathbf{H}^j_{C_i} & \mathbf{0} \end{bmatrix}, \tag{6.34}$$

where $\mathbf{J}_i^j$ is computed in (6.28), for $i > 1$,

$$\mathbf{H}^j_{C_1} = \begin{bmatrix} -\left(\bar{\mathbf{R}}_i^L\right)^T {}^{\mathsf{LS}}\mathbf{p}^j_\times & \mathbf{0} \\ -\left(\bar{\mathbf{R}}_i^R\right)^T {}^{\mathsf{LS}}\mathbf{p}^j_\times & \mathbf{0} \end{bmatrix}, \tag{6.35}$$

$$\mathbf{H}^j_{C_i} = \begin{bmatrix} \left(\bar{\mathbf{R}}_i^L\right)^T {}^{\mathsf{LS}}\mathbf{p}^j_\times & -\left(\bar{\mathbf{R}}_i^L\right)^T \\ \left(\bar{\mathbf{R}}_i^R\right)^T {}^{\mathsf{LS}}\mathbf{p}^j_\times & -\left(\bar{\mathbf{R}}_i^R\right)^T \end{bmatrix}, \tag{6.36}$$

and, for $i = 1$,

$$\mathbf{H}^j_{C_1} = \mathbf{H}^j_{C_i} = \begin{bmatrix} \mathbf{0} & -\left(\bar{\mathbf{R}}_1^L\right)^T \\ \mathbf{0} & -\left(\bar{\mathbf{R}}_1^R\right)^T \end{bmatrix}. \tag{6.37}$$

The rest of the update follows Section 4.3.

4.6 4.6 Filter Update Mechanism and Image Processing Frontend

Our implementation builds upon [108], and preserves its original methodology both for the filter update mechanism, marginalization of camera poses, outlier removal and the image processing frontend.

5 5 Experimental Results

In this section, we compare the performances of the proposed S-UKF-LG and its full IEKF variant, that we call S-IEKF, with state-of-the-art VIO algorithms including S-MSCKF [108], OKVIS (stereo-optimization) [111] and VINS-MONO (monocular-optimization)

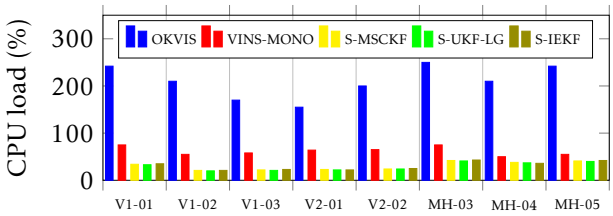


Figure 6.3: Average CPU load of the proposed S-UKF-LG and S-IEKF compared to various methods on the EuRoC dataset [106].

[112], i.e., with different combinations of monocular, stereo, filter-based and optimization-based solutions. We first evaluate the algorithms on the EuRoC dataset [106], and then on a runway environment [108] with high speed flights. In both of the experiments, VINS-MONO considers only the images from the left camera and has its loop closure functionality disabled. The three filters (S-MSCKF, S-UKF-LG and S-IEKF) use the same frontend and the same parameters provided from the S-MSCKF github repository, with $N = 20$ camera poses. Finally, we provide to each algorithms the off-line estimating extrinsic parameters between the IMU and camera frames.

Summary of the results can be found in Figure 6.2 and Figure 6.4. They reveal good performances of our proposed S-UKF-LG, which favorably compares to its conventional Stereo-MSCKF counterpart in terms of RMSE both on position and orientation. Note that, optimization based OKVIS achieves best estimation accuracy, but at the cost of extended CPU load.

5.1 5.1 EuRoC Dataset

The EuRoC [106] dataset includes synchronized 20 Hz stereo images and 200 Hz IMU messages collected on a MAV. The dataset contains sequences of flights with different level of flight dynamics. Figure 6.2 shows the Root Mean Square Error (RMSE) and Figure 6.3 the average CPU load of the different algorithms. As in [108], the filter-based methods do not work properly on `V2_03_difficult` due to their same KLT optical flow algorithm. In terms of accuracy, the filters compete with the stereo-optimization approach OKVIS, whereas the results of VINS-MONO are affected by its monocular camera frontend. The S-UKF-LG and S-IEKF compare favorably to the recent S-MSCKF. Since filters use the same frontend, differences come from the filters' backends. In terms of computational complexity, filter-based solutions reclaims clearly less computational resources than optimization-based methods, in which 80 % of the computation is caused by the frontend including feature detection, tracking and matching, on Precision Tower 7910 with CPU E5-2630 v4 2.20 Hz. The filters themselves take about 10 % of one core when the camera rate is set at 20 Hz.

5.2 5.2 Fast Flight Dataset

To further test the accuracy and the robustness of the proposed S-UKF-LG, the algorithms are evaluated on four outdoor flight datasets with different top speeds of 5, 10, 15 5 m/s, 10 m/s, 15 m/s, and 17.5 m/s [108]. During each sequence, the quadrotor goes 300 m straight and returns to the starting point. The configuration includes two cameras running at 40 Hz and one IMU running at 200 Hz. Figure 6.4 compares the accuracy of the different VIO solutions on the fast flight datasets. The accuracy is evaluated by computing the RMSE of estimates w.r.t. GPS positions only in the xy

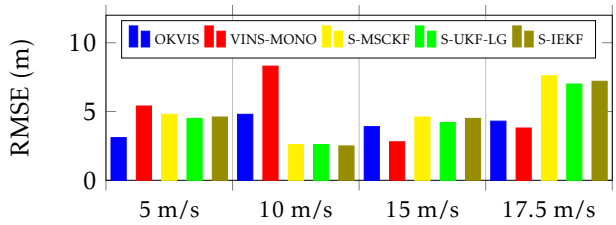


Figure 6.4: Root Mean Square Error of the proposed S-UKF-LG and S-IEKF compared to various methods on the dataset [108]. Statistics are averaged over ten runs on each dataset.

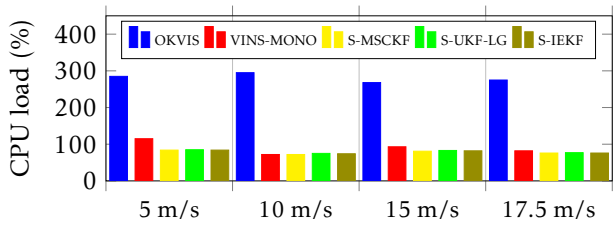


Figure 6.5: Average CPU load of the proposed S-UKF-LG and S-IEKF compared to various methods on the dataset [108].

directions after making corrections on both time and yaw offsets. In this experiment, OKVIS obtains the best results in terms of accuracy, while VINS-MONO generally obtains an accurate estimation with higher variance than than the other methods which increase its RMSE. The S-UKF-LG appears to be slightly more robust than S-MSCKF and S-IEKF. From both experiments, it can be observed that the three filters achieve the lowest CPU usage while maintaining comparable accuracy regarding optimization solutions. However, compared to the experiments with the EuRoC dataset, the image processing frontend spends more computational effort, since the image frequency and resolution are higher, and the fast flight requires more detections of new features.

6 6 Conclusion

In this chapter, we introduced a novel filter-based stereo visual inertial state estimation algorithm that is a stereo multi-state constraint Kalman filter variant. It has the merit of using the UKF approach while achieving execution times that are akin to the standard EKF-based solution, namely S-MSCKF. The UKF approach is generally more robust to non-linearities, and allows fast prototyping since Jacobian explicit computation is not required (and thus readily adapts to model modifications, estimation of additional parameters, and fusion with other sensors). We exploited an efficient inference of the Jacobian that leads to similar computational complexity between our S-UKF-LG solution and the S-MSCKF. We also provided the closed-forms for the measurement Jacobian that lead to an alternative S-IEKF, that can be considered as a stereo version of [59] extending it also with an on-line estimation of the extrinsic parameters, resulting in a consistent filter with high level of accuracy and robustness. Accuracy, efficiency, and robustness of our filters are demonstrated using two challenging datasets.

Part II

Measurement Noise Estimation for Kalman Filter Tuning

CHAPTER 7

Introduction to Part II

The Kalman filter and more generally virtually all state estimation algorithms (particle filter, smoothers) must be *tuned*: given the state transition function $f(\cdot)$ that encodes the dynamical model at play, and observation function $h(\cdot)$ (see Chapter 2), the extent of uncertainty that corrupts both the dynamical model and the observations must be assessed. In the Kalman filter-like approach, uncertainties are modelled as Gaussian (centered) noises with covariance matrices $\mathbf{Q}$ and $\mathbf{N}$ that encode the magnitude of the noises (and their inner correlations). Those matrices appear as tuning parameters of the filter: their value is based on the knowledge one has of the statistics of the noise when available. Wether noise statistics be available or not, their fine tuning often requires a deal of manual "tweaking" from the engineers.

The noise terms encoded in matrices $\mathbf{Q}$ and $\mathbf{N}$ capture what the deterministic model fails to [119], i.e. unmodeled perturbations on the system, which is usually the result of various effects:

- mis-modeled dynamics, e.g. an unfixed level arm between IMU and camera;
- the existence of unmodelled hidden state in the environment, e.g. temperature and its effect on inertial sensors;
- the integration scheme, as time discretization, which introduces additional error;
- and the algorithmic approximations itself, such as the Taylor approximation commonly used for linearization in EKF.

All these effects are commonly characterized as noise. Furthermore, the noise is assumed to be Gaussian and independent over time whereas the phenomena described above cause highly correlated noises. The magnitude of the noise in an EKF is therefore extremely difficult to estimate. This difficulty was yet early noticed in the seminal paper [120], where Kalman himself commented on the difficulty of identifying such parameters:

"In real life, however, the situation is usually reversed. One is given the covariance matrix and the problem is to get the statistical properties [of the perturbation]. This is a subtle and presently largely unsolved problem in experimentation and data reduction."

As a result, many have proposed methods for tuning the parameters in Kalman filters. One of the first methods proposed was to jointly learn the parameters and state/output sequence using expectation-maximization [121]. More recent approaches

employ different optimization approaches, including the simplex algorithm [122], coordinate descent [119], genetic algorithms [123], Bayesian optimization [124], and least squares [125].

1 1 Illustrative Example

To illustrate the importance and difficulties of optimizing the Kalman filter parameters (either by machine learning or manual tuning), consider the practical problem of estimating the variance parameter for a GNSS unit that is being used to estimate the position $\mathbf{x}_n$ of a robot or more generally a vehicle. A standard Kalman filter approach models the GPS readings measured as the true position plus noise as

$$\mathbf{y}_n = \mathbf{x}_n + \mathbf{n}_n \tag{7.1}$$

where $\mathbf{n}_n$ is a zero mean noise term with variance $\sigma^2\mathbf{I}$. The manufacturer specifications of the sensor generally give σ^2 for the unit; otherwise, one can also straightforwardly estimate σ^2 by placing the vehicle at a fixed position and measuring the variability of the GNSS readings. However, in practice either of these choices work very poorly if it is the parameter used in the Kalman filter. This is because GNSS errors are often correlated over time, whereas the straightforward implementation of the Kalman filter assumes that the errors are independent. Thus, if the vehicle is stationary and we average k GNSS readings, the filter assumes that the variance of the resulting estimate is $\sigma^2/k\mathbf{I}$ (obviously this estimate may drop below the actual limit on the sensor's precision). However, if the errors are correlated over time, then the true variance of the resulting position state estimate can be significantly larger. The extreme of this case would be full correlation: if the errors were perfectly correlated so that all k readings are identical, then the variance of the average would be $\sigma^2\mathbf{I}$ instead of $\sigma^2/k\mathbf{I}$. Additionally, if $\sigma^2\mathbf{I}$ is the parameter used in the filter, it tends to make the filter inconsistent. Consequently, in practice, significant human time was expended to try to "tweak" the variance parameter to what they guessed [126].

2 2 Content of Part II

In this part we focus on the problem of assessing noise parameters of the state estimator in the context of navigation. This route is explored in two quite different directions.

In Chapter 9, we focus on a very specific problem. Some recent sensors may generate scans of the environment in the form of clouds of points, and by matching scans one may find the rigid body transformation (translation+rotation) that aligns them best. This may be used to evaluate relative displacements of any vehicle equipped with such sensors between different instants (as long as there is sufficient overlap between the scans). In turn, this information may be used to navigate, either as a means for pure odometry, or to fuse it with other sensors such as inertial sensors. The scan matching algorithm that is used the most is called Iterative Closest Point (ICP). Even if one knows well the precision of the scanner, that is, the uncertainty associated with the depth of each point in the cloud, it is not obvious to deduce the associated uncertainty on the relative displacement. But the knowledge of this extent of uncertainty the ICP estimate bears is paramount to use ICP/scan matching for navigation. Although there has been a number of papers devoted to the subject, we believe our approach clarifies a number of important points, and yields a novel efficient algorithm for assessment of the covariance of the ICP estimate.

In chapter 8, we pursue a different approach to a different problem. Indeed, we consider a wheeled vehicle that navigates with an IMU. Wheeled vehicles have a specific way of moving as they are linked to a surface, and their velocity is approximately aligned with the body frame. This side information about the motion may be used by the Inertial Navigation System (INS) to improve the accuracy of localization. To do so, one may use the standard IMU equations as a dynamical model (exactly as what would be done in an aerial or underwater vehicle), and incorporate the side information in the form of pseudo-measurements, that is, "tell" the Kalman filter it measures a null lateral velocity for instance. However, the lateral velocity is not null, as the vehicle slips. It is thus important to "let the filter know" this information is approximate. The measurement noise matrix $\mathbf{N}$ encodes the magnitude of the uncertainty, that is, essentially the "extent of slip" in the present case. Thus it should be larger when maneuvers are performed, and smaller when the vehicle is moving at constant speed in straight line. The interesting point is that the IMU does measure information related to the vehicle's motion, and notably the gyroscopes detect turns. Our idea is thus to use the IMU measurements to assess in real time an appropriate value for $\mathbf{N}$. However, the precise relation between the most appropriate $\mathbf{N}$ and the inertial sensors, is far from obvious, either it be theoretically or experimentally. Hence we propose to rely on machine (deep) learning to automatically identify a function that relates both in a relevant way.

CHAPTER 8

AI-IMU Dead-Reckoning

The present chapter has been published in IEEE Transactions on Intelligent Vehicles.

Résumé 🇫🇷

Dans ce chapitre, nous proposons une nouvelle méthode précise pour la navigation des véhicules roulant basée uniquement sur une centrake inertielle. Dans le contexte des véhicules autonomes et intelligents, un calcul à la fois robuste et précis basé sur la centrale inertielle peut s'avérer utile pour corréler les flux provenant des capteurs d'imagerie, pour naviguer en toute sécurité à travers lesdifférents obstacles ou pour des arrêts d'urgence sûrs dans le cas extrême de défaillance des capteurs extéroceptifs. Les composants clés de la méthode sont le filtre de Kalman et l'utilisation de réseaux de neurones profonds pour adapter dynamiquement les paramètres de bruit du filtre. La méthode est testée sur le jeu de données KITTI, et notre méthode inertielle d'estimation basée *uniquement* sur l'IMU estime avec précision la position 3D, la vitesse, l'orientation du véhicule et calibre automatiquement les biais de la centrale inertielle. Nous obtenons en moyenne une erreur de translation de 1,10 % et l'algorithme rivalise avec les méthodes les mieux classées qui, en revanche, utilisent un LiDAR ou de la vision stéréo. Nous mettons notre implémentation open source à l'adresse:

`https://github.com/mbrossar/ai-imu-dr`

Chapter abstract

In this chapter we propose a novel accurate method for dead-reckoning of wheeled vehicles based only on an IMU. In the context of intelligent vehicles, robust and accurate dead-reckoning based on the IMU may prove useful to correlate feeds from imaging sensors, to safely navigate through obstructions, or for safe emergency stops in the extreme case of exteroceptive sensors failure. The key components of the method are the Kalman filter and the use of deep neural networks to dynamically adapt the noise parameters of the filter. The method is tested on the KITTI odometry dataset, and our dead-reckoning inertial method based *only* on the IMU accurately estimates 3D position, velocity, orientation of the vehicle and self-calibrates the IMU biases. We achieve on average a 1.10% translational error and the algorithm competes with top-ranked methods which, by contrast, use LiDAR or stereo vision. We make our implementation open-source at:

`https://github.com/mbrossar/ai-imu-dr`

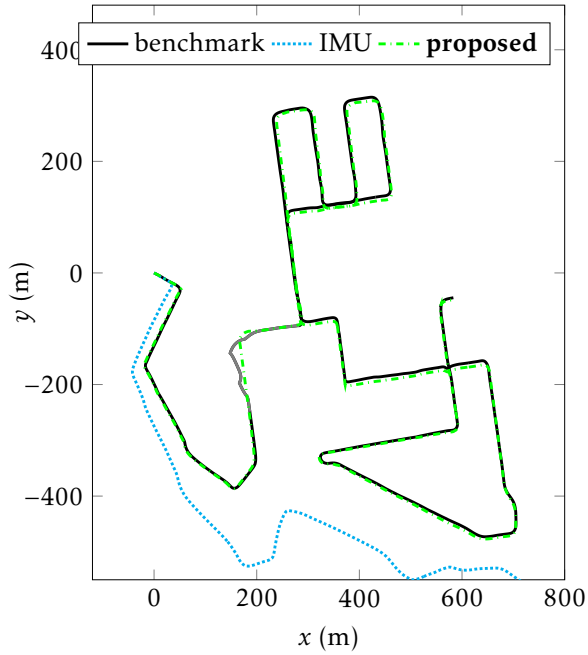


Figure 8.1: Trajectory results on seq. 08 (drive #28, 2011/09/30) [101] of the KITTI dataset. The proposed method (green) accurately follows the benchmark trajectory for the entire sequence (4.2 km, 9 min), whereas the pure integration of (calibrated) IMU signals (cyan) quickly diverges. Both methods use only IMU signals and are initialized with the benchmark pose and velocity. We see during the GPS outage which occurs in this sequence that our solution keeps estimating accurately the trajectory.

1 1 Introduction

Intelligent vehicles need to know where they are located in the environment, and how they are moving through it. An accurate estimate of vehicle dynamics allows validating information from imaging sensors such as lasers, ultrasonic systems and video cameras, correlating the feeds, and also ensuring safe motion throughout whatever may be seen along the road [127]. Moreover, in the extreme case where an emergency stop must be performed owing to severe occlusions, lack of texture, or more generally imaging system failure, the vehicle must be able to assess accurately its dynamical motion. For all those reasons, the IMU appears as a key component of intelligent vehicles [128]. Note that Global Navigation Satellite System (GNSS) allows for global position estimation but it suffers from phase tracking loss in densely built-up areas or through tunnels, is sensitive to jamming, and may not be used to provide continuous accurate and robust localization information, as exemplified by a GPS outage in the well known KITTI dataset [101], see Figure 8.1.

Kalman filters are routinely used to integrate the outputs of IMUs. When the IMU is mounted on a car, it is common practice to make the Kalman filter incorporate side information about the specificity of wheeled vehicle dynamics, such as approximately null lateral and upward velocity assumption in the form of pseudo-measurements [129–134]. However, the degree of confidence the filter should have in this side information is encoded in a covariance noise parameter which is difficult to

set manually, and moreover which should dynamically adapt to the motion, e.g., lateral slip is larger in bends than in straight lines. Using the recent tools from the field of Artificial Intelligence (AI), namely deep neural networks, we propose a method to automatically learn those parameters and their dynamic adaptation for IMU only dead-reckoning. Our contributions, and the chapter's organization, are as follows:

- we introduce a state-space model for wheeled vehicles as well as simple assumptions about the motion of the car;
- we implement a state-of-the-art Kalman filter [42,43] that exploits the kinematic assumptions and combines them with the IMU outputs in a statistical way in Section 4.3. It yields accurate estimates of position, orientation and velocity of the car, as well as IMU biases, along with associated uncertainty (covariance matrices);
- we exploit deep learning for dynamic adaptation of covariance noise parameters of the Kalman filter in Section 5.1. This module greatly improves filter's robustness and accuracy, see Section 6.4;
- we demonstrate the performances of the approach on the KITTI dataset [101] in Section 6. Our approach solely based on the IMU produces accurate estimates and competes with top-ranked LiDAR and stereo camera methods [135,136]; and we do not know of IMU based dead-reckoning methods capable to compete with such results;
- the approach is not restricted to inertial only dead-reckoning of wheeled vehicles. Thanks to the versatility of the Kalman filter, it can easily be coupled with GNSS which is the backbone for IMU self-calibration, applied for railway vehicles [137], or for using IMU as a speedometer in path-reconstruction and map-matching methods [138–141].

2 2 Relation to Previous Literature

Autonomous vehicle must robustly self-localize with their embarked sensor suite which generally consists of odometers, IMUs, radars or LiDARs, and cameras [127,128,141]. SLAM based on inertial sensors, cameras, and/or LiDARs have enabled robust real-time localization systems, see e.g., [135,136]. Although these highly accurate solutions based on those sensors have recently emerged, they may drift when the imaging system encounters troubles.

As concerns wheeled vehicles, taking into account vehicle constraints and odometer measurements are known to increase the robustness of localization systems [130, 131,142,143]. Although quite successful, such systems continuously process a large amount of data which is computationally demanding and energy consuming. Moreover, an autonomous vehicle should run in parallel its own robust IMU-based localization algorithm to perform maneuvers such as emergency stops in case of other sensors failures, or as an aid for correlation and interpretation of image feeds [128].

High precision aerial or military inertial navigation systems achieve very small localization errors but are too costly for consumer vehicles. By contrast, low and medium-cost IMUs suffer from errors such as scale factor, axis misalignment and random walk noise, resulting in rapid localization drift [144]. This makes the IMU-based positioning unsuitable, even during short periods.

Inertial navigation systems have long leveraged virtual and pseudo-measurements from IMU signals, e.g. the widespread Zero velocity UPdaTe (ZUPT) [132,145,146], and covariance adaptation [147]. In parallel, deep learning (more generally machine learning) are gaining much interest for inertial navigation [148,149]. In [148] velocity is estimated using support vector regression whereas [149] use recurrent neural networks for end-to-end inertial navigation (this means the entire traditional pipeline is replaced with a single learning algorithm that learns the whole input to output process from examples). This is promising but restricted to pedestrian dead-reckoning since they generally assume slow horizontal planar motion, and must infer velocity directly from a small sequence of IMU measurements, whereas we can afford using larger sequences.

General "deep" Kalman filter theories [150–153] consist in learning a full state-space model from inputs, measurements, and ground-truth: learning the state space structure, learning the propagation function, and learning the observation function, from scratch. Our method is quite different as it builds upon well established dynamical and measurements equations, and achieves better performances. In particular, the end-to-end learning approach [153], albeit promising, obtain large translational error > 30% in their stereo odometry experiment. [154] combines IMU and GNSS for learning a specific propagation model of IMU errors. In contrast, our method, once parameters are optimized (learned), does not require any GNSS signal. Finally, [155] uses deep learning for estimating covariance of a local odometry algorithm that is fed into a global optimization procedure, and in [156] we used Gaussian processes to learn a wheel encoders error.

Dynamic adaptation of noise parameters in the Kalman filter is standard in the tracking literature [126] and generally based on Auto covariance Least Squares (ALS) [157] and adaptive Kalman filter methods [147,158,159], where adaptation rules are application dependent and are generally the result of manual "tweaking" by engineers. Our approach is wholly different. In the above covariance adaptation is based on statistical tests about measurement error residuals (simply put, a large discrepancy between estimates and actual measurements indicates the parameters are not optimal and prompts the filter to adapt the covariance), whereas in our framework it is solely based on raw IMU data: it is independent of the current state estimates and the observed measurement errors (see Figure 8.8).

Finally, in [119] the authors propose to use classical machine learning techniques to learn static noise parameters (without adaptation) of the Kalman filter, and apply it to the problem of IMU-GNSS fusion.

3 3 IMU and Problem Modelling

An inertial navigation system uses accelerometers and gyrometers provided by the IMU to track the orientation $\mathbf{R}_n^{\text{IMU}}$, velocity $\mathbf{v}_n^{\text{IMU}} \in \mathbb{R}^3$ and position $\mathbf{p}_n^{\text{IMU}} \in \mathbb{R}^3$ of a moving platform relative to a starting configuration $(\mathbf{R}_0^{\text{IMU}}, \mathbf{v}_0^{\text{IMU}}, \mathbf{p}_0^{\text{IMU}})$. The orientation is encoded in a rotation matrix $\mathbf{R}_n^{\text{IMU}} \in SO(3)$ whose columns are the axes of a frame attached to the vehicle.

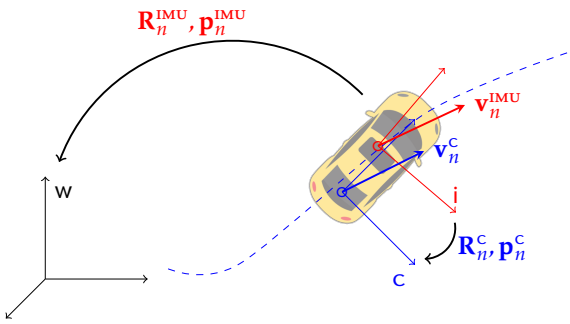


Figure 8.2: The coordinate systems that are used in the chapter. The IMU pose $(\mathbf{R}_n^{\text{IMU}}, \mathbf{p}_n^{\text{IMU}})$ maps vectors expressed in the IMU frame i (red) to the world frame w (black). The IMU frame is attached to the vehicle and misaligned with the car frame c (blue). The pose between the car and inertial frames $(\mathbf{R}_n^{\text{c}}, \mathbf{p}_n^{\text{c}})$ is unknown. IMU velocity $\mathbf{v}_n^{\text{IMU}}$ and car velocity $\mathbf{v}_n^{\text{c}}$ are respectively expressed in the world frame and in the car frame.

3.1 3.1 IMU Modelling

The IMU provides noisy and biased measurements of the instantaneous angular velocity vector ω_n and specific forces $\mathbf{a}_n$ as follows [144]

$$\omega_n^{\text{IMU}} = \omega_n + \mathbf{b}_n^{\omega} + \mathbf{w}_n^{\omega}, \tag{8.1}$$
$$\mathbf{a}_n^{\text{IMU}} = \mathbf{a}_n + \mathbf{b}_n^{\mathbf{a}} + \mathbf{w}_n^{\mathbf{a}}, \tag{8.2}$$

where $\mathbf{b}_n^{\omega}$, $\mathbf{b}_n^{\mathbf{a}}$ are quasi-constant biases and $\mathbf{w}_n^{\omega}$, $\mathbf{w}_n^{\mathbf{a}}$ are zero-mean Gaussian noises. The biases follow a random walk

$$\mathbf{b}_{n+1}^{\omega} = \mathbf{b}_n^{\omega} + \mathbf{w}_n^{\mathbf{b}_\omega}, \tag{8.3}$$
$$\mathbf{b}_{n+1}^{\mathbf{a}} = \mathbf{b}_n^{\mathbf{a}} + \mathbf{w}_n^{\mathbf{b}_\mathbf{a}}, \tag{8.4}$$

where $\mathbf{w}_n^{\mathbf{b}_\omega}$, $\mathbf{w}_n^{\mathbf{b}_\mathbf{a}}$ are zero-mean Gaussian noises.

The kinematic model is governed by the following equations

$$\mathbf{R}_{n+1}^{\text{IMU}} = \mathbf{R}_n^{\text{IMU}} \exp(\omega_n dt), \tag{8.5}$$
$$\mathbf{v}_{n+1}^{\text{IMU}} = \mathbf{v}_n^{\text{IMU}} + (\mathbf{R}_n^{\text{IMU}} \mathbf{a}_n + \mathbf{g})\, dt, \tag{8.6}$$
$$\mathbf{p}_{n+1}^{\text{IMU}} = \mathbf{p}_n^{\text{IMU}} + \mathbf{v}_n^{\text{IMU}} dt, \tag{8.7}$$

between two discrete time instants sampling at dt, where we let the IMU velocity be $\mathbf{v}_n^{\text{IMU}} \in \mathbb{R}^3$ and its position $\mathbf{p}_n^{\text{IMU}} \in \mathbb{R}^3$ in the world frame. $\mathbf{R}_n^{\text{IMU}} \in SO(3)$ is the 3×3 rotation matrix that represents the IMU orientation, i.e. that maps the IMU frame to the world frame, see Figure 8.2. Finally $\exp(\cdot)$ denotes the $SO(3)$ exponential map. The true angular velocity $\omega_n \in \mathbb{R}^3$ and the true specific acceleration $\mathbf{a}_n \in \mathbb{R}^3$ are the inputs of the system (8.5)-(8.7). In our application scenarios, the effects of earth rotation and Coriolis acceleration are ignored, Earth is considered flat, and the gravity vector $\mathbf{g} \in \mathbb{R}^3$ is a known constant.

All sources of error displayed in (8.1) and (8.2) are harmful since a simple implementation of (8.5)-(8.7) leads to a triple integration of raw data, which is much more harmful than the unique integration of differential wheel speeds [141]. Indeed, a bias of order ϵ on the accelerometer measurements has an impact of order $\epsilon t^2/2$ on the position after t seconds, potentially leading to a huge drift.

3.2 3.2 Problem Modelling

We distinguish between three different frames, see Figure 8.2: *i*) the static world frame, w; *ii*) the IMU frame, i, where (8.1)-(8.2) are measured; and *iii*) the car frame, c. The car frame is an ideal frame attached to the car, that will be estimated online and plays a key role in our approach. Its orientation w.r.t. i is denoted $\mathbf{R}_n^{\mathsf{c}} \in SO(3)$ and its origin denoted $\mathbf{p}_n^{\mathsf{c}} \in \mathbb{R}^3$ is the car to IMU level arm. In the rest of the chapter, we tackle the following problem:

IMU Dead-Reckoning Problem. *Given an initial known configuration* $\left(\mathbf{R}_0^{IMU}, \mathbf{v}_0^{IMU}, \mathbf{p}_0^{IMU}\right)$, *perform in real-time IMU dead-reckoning, i.e. estimate the IMU and car variables*

$$\chi_n := (\mathbf{R}_n^{IMU},\ \mathbf{v}_n^{IMU},\ \mathbf{p}_n^{IMU},\ \mathbf{b}_n^{\omega},\ \mathbf{b}_n^{\mathbf{a}},\ \mathbf{R}_n^{\mathsf{c}},\ \mathbf{p}_n^{\mathsf{c}}) \tag{8.8}$$

using only the inertial measurements $\boldsymbol{\omega}_n^{IMU}$ *and* $\mathbf{a}_n^{IMU}$.

4 4 Kalman Filtering with Pseudo-Measurements

The EKF presented in Chapter 3 starts from a dynamical discrete-time non-linear law of the form

$$\chi_{n+1} = f(\chi_n,\ \mathbf{u}_n,\ \mathbf{w}_n) \tag{8.9}$$

where χ_n denotes the state to be estimated, $\mathbf{u}_n$ is a general input, and $\mathbf{w}_n$ is the process noise which is assumed Gaussian with zero mean and covariance matrix $\mathbf{Q}_n$. Assume side information is in the form of loose equality constraints $h(\chi_n) \approx \mathbf{0}$ is available. It is then customary to generate a fictitious observation from the constraint function:

$$\mathbf{y}_n = h(\chi_n) + \mathbf{n}_n, \tag{8.10}$$

and to feed the filter with the information that $\mathbf{y}_n = \mathbf{0}$ (pseudo-measurement) as first advocated by [160], see also [133,142] for application to visual inertial localization and general considerations. The noise is assumed to be a centered Gaussian $\mathbf{n}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{N}_n)$ where the covariance matrix $\mathbf{N}_n$ is set by the user and reflects the degree of validity of the information: the larger $\mathbf{N}_n$ the less confidence is put in the information.

Starting from an initial Gaussian belief about the state, $\chi_0 \sim \mathcal{N}(\hat{\chi}_0, \mathbf{P}_0)$ where $\hat{\chi}_0$ represents the initial estimate and the covariance matrix $\mathbf{P}_0$ the uncertainty associated to it, the EKF alternates between two steps. At the propagation step, the estimate $\hat{\chi}_n$ is propagated through model (8.9) with noise turned off, $\mathbf{w}_n = \mathbf{0}$, and the covariance matrix is updated through

$$\mathbf{P}_{n+1} = \mathbf{F}_n \mathbf{P}_n \mathbf{F}_n^T + \mathbf{G}_n \mathbf{Q}_n \mathbf{G}_n^T, \tag{8.11}$$

where $\mathbf{F}_n$, $\mathbf{G}_n$ are Jacobian matrices of $f(\cdot)$ at current estimate w.r.t. χ_n and $\mathbf{u}_n$. At the update step, pseudo-measurement is taken into account, and Kalman equations allow updating the estimate $\hat{\chi}_{n+1}$ and its covariance matrix $\mathbf{P}_{n+1}$ accordingly.

To implement an EKF, the engineer needs to determine the functions $f(\cdot)$ and $h(\cdot)$, and the associated noise matrices $\mathbf{Q}_n$ and $\mathbf{N}_n$. In this chapter, noise parameters $\mathbf{Q}_n$ and $\mathbf{N}_n$ will be wholly learned by a neural network.

4.1 4.1 Defining the Dynamical Model $f(\cdot)$

We now need to assess the evolution of state variables (8.8). The evolution of $\mathbf{R}_n^{\text{IMU}}$, $\mathbf{p}_n^{\text{IMU}}$, $\mathbf{v}_n^{\text{IMU}}$, $\mathbf{b}_n^{\omega}$ and $\mathbf{b}_n^{\mathbf{a}}$ is already given by the standard equations (8.3)-(8.7), leading to the input $\mathbf{u}_n = \left[\omega_n^T, \mathbf{a}_n^T\right]^T$. The additional variables $\mathbf{R}_n^{\text{c}}$ and $\mathbf{p}_n^{\text{c}}$ represent the car frame with respect to the IMU. This car frame is rigidly attached to the car and encodes an unknown point where the pseudo-measurements of Section 4.2 are most advantageously made. In [134] this point is located in the rear wheel, but in our approach we let the algorithm find its location to enhance performances. As IMU is also rigidly attached to the car, and $\mathbf{R}_n^{\text{c}}$, $\mathbf{p}_n^{\text{c}}$ represent misalignment between IMU and car frame, they are approximately constant:

$$\mathbf{R}_{n+1}^{\text{c}} = \mathbf{R}_n^{\text{c}} \exp(\mathbf{w}_n^{\mathbf{R}^{\text{c}}}), \tag{8.12}$$

$$\mathbf{p}_{n+1}^{\text{c}} = \mathbf{p}_n^{\text{c}} + \mathbf{w}_n^{\mathbf{p}^{\text{c}}}. \tag{8.13}$$

where we let $\mathbf{w}_n^{\mathbf{R}^{\text{c}}}$, $\mathbf{w}_n^{\mathbf{p}^{\text{c}}}$ be centered Gaussian noises with small covariance matrices $\sigma^{\mathbf{R}^{\text{c}}}\mathbf{I}$, $\sigma^{\mathbf{p}^{\text{c}}}\mathbf{I}$ that will be learned during training. Although those variables are bounded in practice, we model them as random walks to ensure good tracking by the Kalman filter, as is usually done in the navigation literature regarding IMU biases (see (8.3)-(8.4), that are also bounded in practice). Noises $\mathbf{w}_n^{\mathbf{R}^{\text{c}}}$ and $\mathbf{w}_n^{\mathbf{p}^{\text{c}}}$ encode possible small variations through time of level arm due to the lack of rigidity stemming from dampers and shock absorbers.

4.2 4.2 Defining the Pseudo-Measurements $h(\cdot)$

Consider the different frames depicted on Figure 8.2. The velocity of the origin point of the car frame, expressed in the car frame, writes

$$\mathbf{v}_n^{\text{c}} = \begin{bmatrix} v_n^{\text{for}} \\ v_n^{\text{lat}} \\ v_n^{\text{up}} \end{bmatrix} = (\mathbf{R}_n^{\text{c}})^T \left((\mathbf{R}_n^{\text{IMU}})^T \mathbf{v}_n^{\text{IMU}} + (\omega_n)_\times \mathbf{p}_n^{\text{c}} \right), \tag{8.14}$$

from basic screw theory. In the car frame, we consider that the car lateral and vertical velocities are roughly null, that is, we generate two scalar pseudo observations of the form (8.10)

$$\mathbf{y}_n = \begin{bmatrix} y_n^{\text{lat}} \\ y_n^{\text{up}} \end{bmatrix} = \begin{bmatrix} h^{\text{lat}}(\mathcal{X}_n) + \mathrm{n}_n^{\text{lat}} \\ h^{\text{up}}(\mathcal{X}_n) + \mathrm{n}_n^{\text{up}} \end{bmatrix} = \begin{bmatrix} v_n^{\text{lat}} \\ v_n^{\text{up}} \end{bmatrix} + \mathbf{n}_n, \tag{8.15}$$

where the noises $\mathbf{n}_n = \left[\mathrm{n}_n^{\text{lat}}, \mathrm{n}_n^{\text{up}}\right]^T$ are assumed centered and Gaussian with covariance matrix $\mathbf{N}_n \in \mathbb{R}^{2\times 2}$. The filter is then fed with the pseudo-measurement that $y_n^{\text{lat}} = y_n^{\text{up}} = 0$.

Assumptions that v_n^{lat} and v_n^{up} are roughly null are common for cars moving forward on human made roads or wheeled robots moving indoor. Treating them as loose constraints, i.e., allowing the uncertainty encoded in $\mathbf{N}_n$ to be non strictly null, leads to much better estimates than treating them as hard constraints, that is, $\mathbf{N}_n = \mathbf{0}$ [142]. These assumptions are *statistically* true but does not hold for a given instant.

It should be duly noted the vertical velocity v_n^{up} is expressed in the car frame, and thus the assumption it is roughly null generally holds for a car moving on a road even if the motion is 3D. It is quite different from assuming null vertical velocity in the world frame, which then boils down to planar horizontal motion.

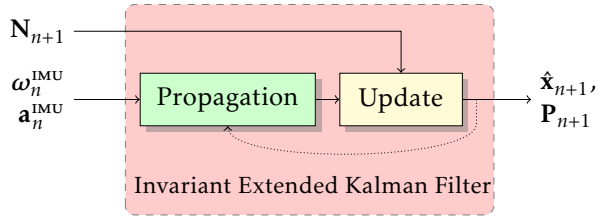


Figure 8.3: Structure of the IEKF. The filter uses the noise parameter $\mathbf{N}_{n+1}$ of pseudo-measurements (8.15) to yield a real time estimate of the state $\hat{\chi}_{n+1}$ along with covariance $\mathbf{P}_{n+1}$, identically to the conventional EKF.

Figure 8.4: Structure of the proposed system for inertial dead-reckoning. The measurement noise adapter feeds the filter with noise parameter $\mathbf{N}_n$ computed from raw IMU signals only.

The main point of the present work is that the validity of the null lateral and vertical velocity assumptions widely vary depending on what maneuver is being performed: for instance, v_n^{lat} is much larger in turns than in straight lines. The role of the noise parameter adapter of Section 5.1, based on AI techniques, will be to dynamically assess the parameter $\mathbf{N}_n$ that reflects confidence in the assumptions, as a function of past and present IMU measurements.

4.3 4.3 The Invariant Extended Kalman Filter

For inertial navigation, we advocate the use of the IEKF, see [42,43], that has recently given raise to a commercial aeronautics product [43,85] and to various successes in the field of visual inertial odometry [62,91,95].

We opt for an IEKF to perform the fusion between the IMU measurements (8.1)-(8.2) and (8.15) treated as pseudo-measurements we advocate IEKF for inertial navigation, see Chapter 2. Its architecture, which is identical to the conventional EKF's, is recapped in Figure 8.3. The interested reader is referred to the Appendix where the exact equations of the filter are provided.

5 5 Proposed AI-IMU Dead-Reckoning

This section describes our system for recovering all the variables of interest from IMU signals only. Figure 8.4 illustrates the approach which consists of two main blocks summarized as follows:

- the filter integrates the inertial measurements (8.1)-(8.2) with dynamical model $f(\cdot)$ given by (8.3)-(8.7) and (8.12)-(8.13), and exploits (8.15) as measurements $h(\cdot)$ with covariance matrix $\mathbf{N}_n$ to refine its estimates;
- the noise parameter adapter determines in real-time the most suitable covariance noise matrix $\mathbf{N}_n$. This deep learning based adapter converts directly raw IMU signals (8.1)-(8.2) into covariance matrices $\mathbf{N}_n$ without requiring knowledge of any state estimate nor any other quantity.

The amplitude of process noise parameters $\mathbf{Q}_n$ are considered fixed by the algorithm, and are learned during training.

Note that the adapter computes covariances meant to improve localization accuracy, and thus the computed values may broadly differ from the actual statistical covariance of $\mathbf{y}_n$ (8.15), see Section 6.4 for more details. In this respect, our approach is related to [153] but the considered problem is more challenging: our state-space is of dimension 21 whereas [153] has a state-space of dimension only 3. Moreover, we compare our results in the sequel to state-of-the-art methods based on stereo cameras and LiDARs, and we show we may achieve similar results based on the moderately precise IMU only.

5.1 5.1 AI-based Measurement Noise Parameter Adapter

The measurement noise parameter adapter computes at each instant n the covariance $\mathbf{N}_{n+1}$ used in the filter update, see Figure 8.3. The base core of the adapter is a Convolutional Neural Network (CNN) [161]. The networks takes as input a window of N inertial measurements and computes

$$\mathbf{N}_{n+1} = \mathrm{CNN}\Big(\Big\{\boldsymbol{\omega}_i^{\mathrm{IMU}},\ \mathbf{a}_i^{\mathrm{IMU}}\Big\}_{i=n-N}^{n}\Big). \tag{8.16}$$

Our motivations for the above simple CNN-like architecture are threefold:

i) avoiding over-fitting by using a relatively small number of parameters in the network and also by making its outputs independent of state estimates;

ii) obtaining an interpretable adapter from which one can infer general and safe rules using reverse engineering, e.g. to which extent must one inflate the covariance during turns, for e.g., generalization to all sorts of wheeled and commercial vehicles, see Section 6.4;

iii) letting the network be trainable. Indeed, as reported in [153], training is quite difficult and slow. Setting the adapter with a recurrent architecture [161] would make the training even much harder.

The complete architecture of the adapter consists of CNN layers (we use 2 layers, see Section 5.2) followed by a full layer outputting a vector $\mathbf{z}_n = \left[z_n^{\mathrm{lat}}, z_n^{\mathrm{up}}\right]^T \in \mathbb{R}^2$. Based on the latter, the covariance $\mathbf{N}_{n+1} \in \mathbb{R}^{2\times 2}$ is then computed as

$$\mathbf{N}_{n+1} = \mathrm{diag}\Big(\sigma_{\mathrm{lat}}^2 10^{\beta \tanh(z_n^{\mathrm{lat}})},\ \sigma_{\mathrm{up}}^2 10^{\beta \tanh(z_n^{\mathrm{up}})}\Big), \tag{8.17}$$

with $\beta \in \mathbb{R}_{>0}$, and where σ_{lat} and σ_{up} correspond to our initial guess for the noise parameters. The network thus may inflate covariance up to a factor 10^{β} and squeeze it up to a factor $10^{-\beta}$ with respect to its original values. Additionally, as long as the network is disabled or barely reactive (e.g. when starting training), we get $\mathbf{z}_n \approx \mathbf{0}$ and recover the initial covariance $\mathrm{diag}\big(\sigma_{\mathrm{lat}},\ \sigma_{\mathrm{up}}\big)^2$. We manually remove the part of the sequences where outlier are present. Handling outlier with robust uncertainty estimation [11] is set for future work.

Regarding process noise parameter $\mathbf{Q}_n$, we choose to fix it to a value $\mathbf{Q}$ and leave its dynamic adaptation for future work. However its entries are optimized during training, see Section 5.3.

5.2 5.2 Implementation Details

We provide in this section the setting and the implementation details of our method. We implement the full approach in Python with the PyTorch[1] library for the noise parameter adapter part. We set as initial values before training

$$\mathbf{P}_0 = \mathrm{diag}\left(\sigma_0^{\mathbf{R}}\mathbf{I}_2, 0, \sigma_0^{\mathbf{v}}\mathbf{I}_2, \mathbf{0}_4, \sigma_0^{\mathbf{b}^\omega}\mathbf{I}, \sigma_0^{\mathbf{b}^\mathbf{a}}\mathbf{I}, \sigma_0^{\mathbf{R}^\mathrm{c}}\mathbf{I}, \sigma_0^{\mathbf{p}^\mathrm{c}}\mathbf{I}\right)^2, \tag{8.18}$$

$$\mathbf{Q} = \mathrm{diag}\left(\sigma_\omega\mathbf{I},\ \sigma_\mathbf{a}\mathbf{I},\ \sigma_{\mathbf{b}^\omega}\mathbf{I},\ \sigma_{\mathbf{b}^\mathbf{a}}\mathbf{I},\ \sigma_{\mathbf{R}^\mathrm{c}}\mathbf{I},\ \sigma_{\mathbf{p}^\mathrm{c}}\mathbf{I}\right)^2, \tag{8.19}$$

$$\mathbf{N}_n = \mathrm{diag}\left(\sigma_{\mathrm{lat}},\ \sigma_{\mathrm{up}}\right)^2, \tag{8.20}$$

where $\mathbf{I} = \mathbf{I}_3$, $\sigma_0^{\mathbf{R}} = 10^{-3}\,\mathrm{rad}$, $\sigma_0^{\mathbf{v}} = 0.3\,\mathrm{m/s}$, $\sigma_0^{\mathbf{b}^\omega} = 10^{-4}\,\mathrm{rad/s}$, $\sigma_0^{\mathbf{b}^\mathbf{a}} = 3.10^{-2}\,\mathrm{m/s^2}$, $\sigma_0^{\mathbf{R}^\mathrm{c}} = 3.10^{-3}\,\mathrm{rad}$, $\sigma_0^{\mathbf{p}^\mathrm{c}} = 10^{-1}\,\mathrm{m}$ in the initial error covariance $\mathbf{P}_0$, $\sigma_\omega = 1.4.10^{-2}\,\mathrm{rad/s}$, $\sigma_\mathbf{a} = 3.10^{-2}\,\mathrm{m/s^2}$, $\sigma_{\mathbf{b}^\omega} = 10^{-4}\,\mathrm{rad/s}$, $\sigma_{\mathbf{b}^\mathbf{a}} = 10^{-3}\,\mathrm{m/s^2}$, $\sigma_{\mathbf{R}^\mathrm{c}} = 10^{-4}\,\mathrm{rad}$, $\sigma_{\mathbf{p}^\mathrm{c}} = 10^{-4}\,\mathrm{m}$ for the noise propagation covariance matrix $\mathbf{Q}$, $\sigma_{\mathrm{lat}} = 1\,\mathrm{m/s}$, and $\sigma_{\mathrm{up}} = 3\,\mathrm{m/s}$ for the measurement covariance matrix. The zero values in the diagonal of $\mathbf{P}_0$ in (8.18) corresponds to a perfect prior of the initial yaw, position and zero vertical speed, as (IMU-based) dead reckoning methods can only estimate a trajectory and a yaw relative to the starting point.

The adapter is a 1D temporal convolutional neural network with 2 layers. The first layer has kernel size 5, output dimension 32, and dilatation parameter 1. The second has kernel size 5, output dimension 32 and dilatation parameter 3, thus it set the window size equal to $N = 15$. The CNN is followed by a fully connected layer that output the scalars z^{lat} and z^{up}. Each activation function between two layers is a ReLU unit [161]. We define $\beta = 3$ in the right part of (8.17) which allows for each covariance element to be 10^3 higher or smaller than its original values.

5.3 5.3 Training

We seek to optimize the relative translation error t_{rel} computed from the filter estimates $\hat{\chi}_n$, which is the averaged increment error for all possible sub-sequences of length 100 m to 800 m.

Toward this aim, we first define the learnable parameters. It consists of the 6210 parameters of the adapter, along with the parameter elements of $\mathbf{P}_0$ and $\mathbf{Q}$ in (8.18)-(8.19), which add 12 parameters to learn. We then choose an Adam optimizer [162] with learning rate 10^{-4} that updates the trainable parameters. Training consists of repeating for a chosen number of epochs the following iterations:

i) sample a part of the dataset;

ii) get the filter estimates for then computing loss and gradient w.r.t. the learnable parameters;

ii) update the learnable parameters with gradient and optimizer.

We train the networks for 400 epochs and then applying continual training strategies [163]. This makes sense for online training in a context where the vehicle gathers accurate ground-truth poses from e.g. its LiDAR system or precise GNSS. It requires careful procedures for avoiding over-fitting, such that we use dropout and data augmentation [161]. Dropout refers to ignoring units of the adapter during training, and

test seq.	length (km)	dura -tion (s)	IMLS t_{rel} (%)	IMLS r_{rel} (deg/km)	ORB-SLAM2 t_{rel} (%)	ORB-SLAM2 r_{rel} (deg/km)	IMU t_{rel} (%)	IMU r_{rel} (deg/km)	**proposed** t_{rel} (%)	**proposed** r_{rel} (deg/km)
01	2.6	110	**0.82**	**1.0**	1.41	1.9	5.35	1.2	1.56	1.2
03	-	80	-	-	-	-	-	-	-	-
04	0.4	27	**0.33**	1.2	0.47	2.2	0.97	1.0	1.22	**0.4**
06	1.2	110	**0.33**	**0.8**	0.73	2.2	5.78	1.9	1.57	1.9
07	0.7	110	**0.33**	**1.5**	0.91	4.9	12.6	3.0	1.32	3.0
08	3.2	407	**0.80**	**1.8**	1.03	3.0	549	5.6	1.08	3.2
09	1.7	159	**0.55**	**1.2**	0.81	2.3	23.4	3.5	0.82	2.2
10	0.9	120	**0.53**	1.7	**0.66**	3.1	4.58	2.5	1.05	2.5
average scores			**0.64**	**1.2**	0.99	2.6	171	3.1	0.97	2.3

Table 8.1: Results on [101]. IMU integration tends to drift or diverge, whereas the proposed method may be used as an alternative to LiDAR based (IMLS [135]) and stereo vision based (ORB-SLAM2 [136]) methods, using only IMU information. Indeed, on average, our dead-reckoning solution performs better than ORB-SLAM2 and achieves a translational error being close to that of the LiDAR based method IMLS, which is ranked 3rd on the KITTI online benchmarking system. Data from seq. 03 was unavailable for testing algorithms, and sequences 00, 02 and 05 are discussed separately in Section 6.3. It should be duly noted, though, that IMLS, ORB-SLAM2, and the proposed AI-IMU algorithm all use different sensors. The interest of ranking algorithms based on different information is debatable. Our goal here is rather to evidence that using data from a moderately precise IMU only, one can achieve similar results as state-of-the art systems based on imaging sensors, which is a rather surprising feature.

we set the probability $p = 0.5$ of any CNN element to be ignored (set to zero) during a sequence iteration.

Regarding i), we sample a batch of nine 1 min sequences, where each sequence starts at a random arbitrary time. We add to data a small Gaussian noise with standard deviation 10^{-4}, a.k.a. data augmentation technique. We compute ii) with standard backpropagation, and we finally clip the gradient norm to a maximal value of 1 to avoid gradient explosion at step iii).

We stress the loss function consists of the relative translation error t_{rel}, i.e. we optimize parameters for improving the filter accuracy, disregarding the values actually taken by $\mathbf{N}_n$, in the spirit of [153].

6 6 Experimental Results

We evaluate the proposed method on the KITTI dataset [101], which contains data recorded from LiDAR, cameras and IMU, along with centimeter accurate ground-truth pose from different environments (e.g., urban, highways, and streets). The dataset contains 22 sequences for benchmarking odometry algorithms, 11 of them contain publicly available ground-truth trajectory, raw and synchronized IMU data. We download the raw data with IMU signals sampled at 100 Hz ($dt = 10^{-2}$ s) rather than the synchronized data sampled at 10 Hz, and discard seq. 03 since we did not find raw data for this sequence. The RT3003[2] IMU has announced gyro and accelerometer bias stabil-

[1] https://pytorch.org/
[2] https://www.oxts.com/

ity of respectively 36 deg/h and 1 mg. The KITTI dataset has an online benchmarking system that ranks algorithms. However we could not submit our algorithm for online ranking since sequences used for ranking do not contain IMU data, which is reserved for training only. Our implementation is made open-source at:

`https://github.com/mbrossar/ai-imu-dr`.

6.1 6.1 Evaluation Metrics and Compared Methods

To assess performances we consider the two error metrics proposed in [101]:

Relative Translation Error (t_{rel}): which is the averaged relative translation increment error for all possible sub-sequences of length 100 m, ..., 800 m, in percent of the traveled distance;

Relative Rotational Error (r_{rel}): that is the relative rotational increment error for all possible sub-sequences of length 100 m, ..., 800 m, in degree per kilometer.

We compare four methods which alternatively use LiDAR, stereo vision, and IMU-based estimations:

- **IMLS** [135]: a recent state-of-the-art LiDAR-based approach ranked 3rd in the KITTI benchmark. The author provided us with the code after disabling the loop-closure module;
- **ORB-SLAM2** [136]: a popular and versatile library for monocular, stereo and RGB-D cameras that computes a sparse reconstruction of the map. We got the open-source code, disabled loop-closure and we then evaluate the stereo algorithm without modifying any parameter;
- **IMU**: the direct integration of the IMU measurements based on (8.4)-(8.5), that is, pure inertial navigation;
- **proposed**: the proposed approach, that uses only the IMU signals and involves no other sensor.

6.2 6.2 Trajectory Results

We follow the same protocol for evaluating each sequence: i) we initialize the filter with parameters described in Section 5.2; ii) we train then the noise parameter adapter following Section 5.3 for 400 epochs without the evaluated sequence (e.g. for testing seq. 10, we train on seq. 00-09) so that the noise parameter has *never* been confronted with the evaluated sequence; iii) we run the IMU-based methods on the full raw sequence with ground-truth initial configuration ($\mathbf{R}_0^{\text{IMU}}, \mathbf{v}_0^{\text{IMU}}, \mathbf{p}_0^{\text{IMU}}$), whereas we initialize remaining variables at zero ($\mathbf{b}_0^{\omega} = \mathbf{b}_0^{\mathbf{a}} = \mathbf{p}_0^{\text{c}} = \mathbf{0}$, $\mathbf{R}_0^{\text{c}} = \mathbf{I}$); and iv) we get the estimates only on time corresponding to the odometry benchmark sequence. LiDAR and visual methods are directly evaluated on the odometry sequences.

Results are averaged in Table 8.1 and illustrated in Figures 8.1, 8.5 and 8.6, where we exclude sequences 00, 02 and 05 which contain problems with the data, and will be discussed separately in Section 6.3. From these results, we see that:

- LiDAR and visual methods perform generally well in all sequences, and the LiDAR method achieves slightly better results than its visual counterpart;

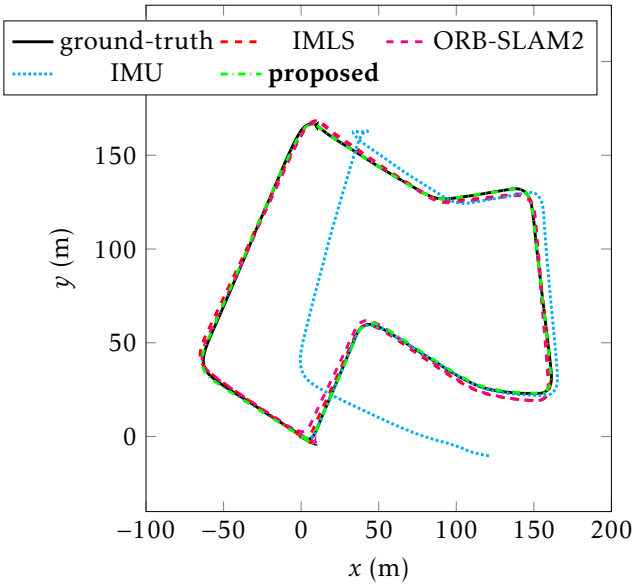


Figure 8.5: Results on seq. 07 (drive #27, 2011/09/30) [101]. The proposed method competes with LiDAR and visual odometry methods, whereas the IMU integration broadly drifts after the car's stop.

- our method competes on average with the latter imaging based methods, see Table 8.1;
- directly integrating the IMU signals leads to rapid drift of the estimates, especially for the longest sequences but even for short periods;
- Our method looks unaffected by stops of the car, as in seq. 07, see Figure 8.5.

The results are remarkable as we use none of the vision sensors, nor wheel odometry. We only use the IMU, which moreover has moderate precision.

We also sought to compare our method to visual inertial odometry algorithms. However, we could not find open-source code for such methods that perform well on the full KITTI dataset. We tested [164] but the code in still under development (results sometimes diverge), and the authors in [165] evaluate their not open-source method for short sequences (≤ 30 s). The chapter [62,166] evaluate their visual inertial odometry methods on seq. 08, both get a final error around 20 m, which is four times what our method achieves (final distance to ground-truth is as low as 5 m). This clearly evidences that methods taylored for ground vehicles [130,142] may achieve higher accuracy and robustness than general methods designed for smartphones, drones and aerial vehicles.

6.3 6.3 Results on Sequences 00, 02 and 05

Following the procedure described in Section 6.2, the proposed method seems to have degraded performances on seq. 00, 02 and 05, see Figure 8.9. However, the behavior is wholly explainable: data are missing for a couple of seconds due to logging problems which appear both for IMU and ground-truth. This is illustrated in Figure 8.10 for seq.

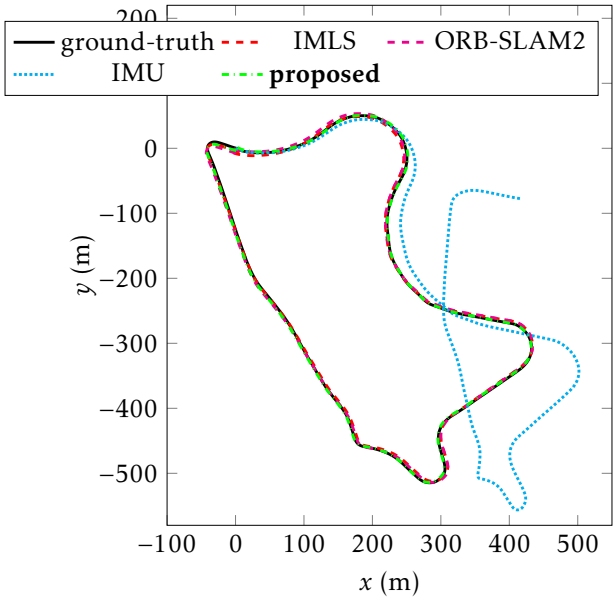


Figure 8.6: Results on seq. 09 (drive #33, 2011/09/30) [101]. The proposed method competes with LiDAR and visual odometry methods, whereas the IMU integration drifts quickly after the first turn.

02 where we plot available data over time. We observe a jump in the IMU and ground-truth signals, evidencing that data are missing between $t = 1$ and $t = 3$. The problem was corrected manually when using those sequences in the training phase described in Section 5.3.

Although those sequences could have been discarded due to logging problems, we used them for testing without correcting their problems. This naturally results in degraded performance, but also evidences our method is remarkably robust to such errors in spite of their inherent harmfulness. For instance, the 2 s time jump of seq. 02 results in estimate shift, but no divergence occurs for all that, see Figure 8.9.

6.4 6.4 Further Results

The performances may be explained by: i) the use of a recent IEKF that has been proved to be well suited for IMU based localization; ii) incorporation of side information in the form of pseudo-measurements with dynamic noise parameter adaptation learned by a neural network; and iii) accounting for a “loose" misalignment between the IMU and the car.

As concerns i), it should be stressed the method is perfectly suited to the use of a conventional EKF and is easily adapted if need be. However we advocate the use of an IEKF owing to its accuracy and convergence properties [42].

To illustrate the benefits of points ii) and iii), we consider two sub-versions of the proposed algorithm. One without alignment, i.e. where $\mathbf{R}_n^c$ and $\mathbf{p}_n^c$ are not included in the state and fixed at their initial values $\mathbf{R}_n^c = \mathbf{I}$, $\mathbf{p}_n^c = \mathbf{0}$, and a second one that uses the static filter parameters (8.18)-(8.20). End trajectory results for the highway seq. 01 are plotted in Figure 8.7, where we see that the two sub-version methods have trouble when the car is turning. Therefore their respective translational errors t_{rel} are

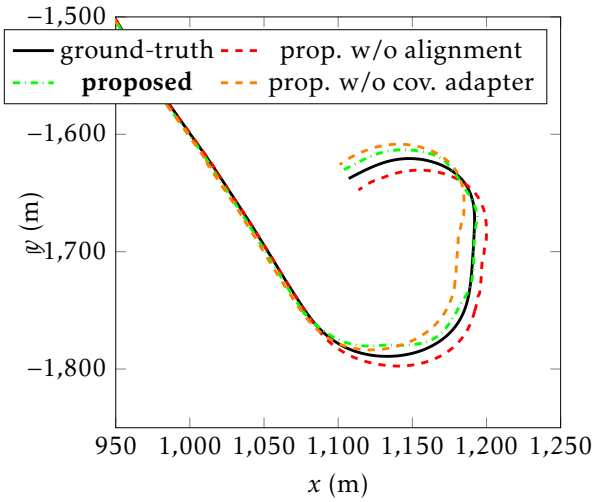


Figure 8.7: End trajectory results on the highway seq. 01 (drive #42, 2011/10/30) [101]. Dynamically adapting the measurement covariance and considering misalignment between car and inertial frames makes the translational error drop from 1.94% to 1.11%, see Section 6.4.

higher than the full version of the proposed method: the proposed method achieves 1.11%, the method without alignment level arm achieves 1.65%, and the absence of covariance adaptation yields 1.94% error. All methods have the same rotational error $r_{rel} = 0.12\,\mathrm{deg/m}$. This could be anticipated for the considered sequence since the full method has the same rotational error than standard IMU integration method.

6.5 6.5 Limitations and Discussion

Limitations regarding the approach revolve around the following points: 1) one may not want to depend on AI; 2) generalization; and 3) computational complexity and real-time embedded applications. Regarding the first point, it may be objected AI is only as good as the data available and hence has weaknesses. Moreover, when vision sensors fail, IMU is critical to provide a refuge trajectory but in case of emergency one might want to depend on more classical and robust IMU algorithms, s AI is still difficult to certify [167]. To this respect, note that our deep network only affects the Kalman filter tuning parameters. If one wants to remove AI while achieving good performances, it is possible to infer some adaptation rules from the obtained results. To this end, we plot the covariances computed by the adapter for seq. 01 in Figure 8.8. The adapter clearly increases the covariances during the bend, i.e. when the gyro yaw rate is important. This is especially the case for the zero velocity measurement (8.15): its associated covariance is inflated by a factor of 10^2 between $t = 90\,\mathrm{s}$ and $t = 110\,\mathrm{s}$. This illustrates the kind of information the adapter has learned and how this behavior may be emulated in a more traditional pipeline.

6.6 6.6 Discussion

As our deep network only affects the Kalman filter tuning parameters, it is possible to infer some adaptation rules from the obtained results. These rules may substitute the deep network while achieving good performances, if one wants to remove AI for

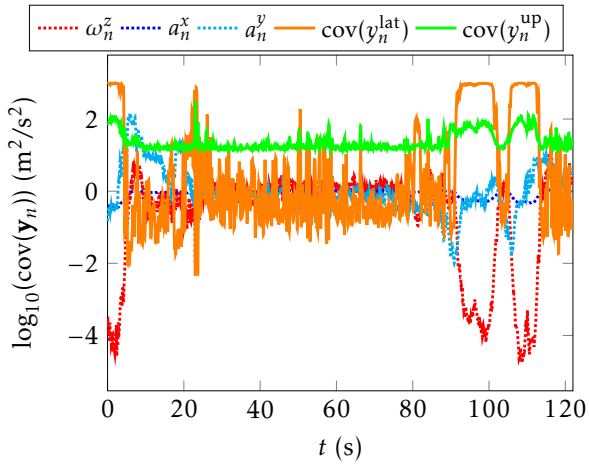


Figure 8.8: Covariance values computed by the adapter on the highway seq. 01 (drive #42, 2011/10/30) [101]. We clearly observe a large increase in the covariance values when the car is turning between $t = 90\,\mathrm{s}$ and $t = 110\,\mathrm{s}$.

safety and commercial purposes (as AI is still difficult to certify [167]), or simply due to hardware limitations. To this end, we plot the covariances computed by the adapter for seq. 01 in Figure 8.8. The adapter clearly increases the covariances during the bend, i.e. when the gyrometer's yaw rate is larger. This is especially the case for the zero velocity measurement (8.15): its associated covariance is inflated by a factor of 10^2 between $t = 90\,\mathrm{s}$ and $t = 110\,\mathrm{s}$. Indeed, such a large noise parameter inflation indicates the AI-based part of the algorithm has learned and recognizes that pseudo-measurements have no value for localization at those precise moments, so the filter should barely consider them. This illustrates the kind of information the adapter has learned and how this behavior may be emulated in a more traditional pipeline. An alternative consists in using these pseudo-measurement with robust cost [11], e.g. Huber loss in the filter update or in the loss for training neural networks. Learning to adapt with robust (non Gaussian) uncertainty is a promishing perspectives of future works.

Figure 8.8 gives also a striking illustration of the role of observation noise autocorrelation in Kalman filter tuning. We lack space to get into the theory of effective sample size [168] but intuitively, with a minimal time τ between 2 independent observations of 1 s, measurements at 100 Hz do not bring much more information than at 1 Hz. As a consequence, if the frequency f_s of the Kalman updates (100 Hz here) is high, i.e., $f_s\tau \gg 1$, then the *instant* statistical uncertainty of the observation is not the optimal value for the matrix $\mathbf{N}_n$ and a good rule of thumb is multiplying it by $f_s\tau$. These theoretical considerations become very concrete on Figure 8.8: we see the optimal value of the "covariance" of observation $\mathbf{y}_n^{\mathrm{up}}$ to be given to the Kalman filter is $100\,\mathrm{m}^2/\mathrm{s}^2$, which models for instance a noise of *instant* covariance $1\,\mathrm{m}^2/\mathrm{s}^2$ with autocorrelation time 1 s, which is coherent for vertical variations of velocity. However if for some reason the practitioner wants to keep the noise covariance parameter $\mathbf{N}_n$ at bounded values at all times, it is always possible to turn off pseudo-measurements or enforce a predefined upper bound on $\mathbf{N}_n$ that feeds the Kalman filter when the AI-based adapter goes above the predefined bound.

Interestingly, we see large statistical uncertainty (which should clearly be below $100\,\mathrm{m}^2/\mathrm{s}^2$) and the inflated covariances whose values are computed for the sole pur-

test seq.	length (km)	dura -tion (s)	IMLS t_{rel} (%)	IMLS r_{rel} (deg/km)	ORB-SLAM2 t_{rel} (%)	ORB-SLAM2 r_{rel} (deg/km)	IMU t_{rel} (%)	IMU r_{rel} (deg/km)	**proposed** t_{rel} (%)	**proposed** r_{rel} (deg/km)
00	3.7	454	**0.50**	**1.8**	0.84	2.9	426	46.8	6.81	24.8
02	5.1	466	**0.53**	**1.4**	0.79	2.7	346	8.7	3.37	3.8
05	2.2	278	**0.32**	**1.4**	0.55	2.2	189	5.2	3.05	8.7

Table 8.2: Results on [101] on seq. 00, 02 and 05. The degraded results of the proposed method are wholly explained by a problem of missing data, see Section 6.3 and Figure 8.10.

pose of filter's performance enhancement. Indeed, such a large noise parameter inflation indicates the AI-based part of the algorithm has learned and recognizes that pseudo-measurements have no value for localization at those precise moments, so the filter should barely consider them. Note that, if for some reason the practitioner wants to keep the noise covariance parameter $\mathbf{N}_n$ at more realistic values at all times, for instance in the context of fusion with other sensors, it is always possible to turn-off pseudo-measurements or enforce a predefined upper bound on $\mathbf{N}_n$ that feeds the Kalman filter when the AI-based adapter goes above the predefined bound.

Finally, in terms of execution time in view of real time applications, once the network is trained, using the network is very cheap computationally speaking. Indeed the network takes the raw IMU data and outputs a covariance for the Kalman filter. This process takes 15% only of the overall algorithm time execution, while the Kalman filter takes 85%. As Kalman filter routinely run on embedded code in inertial navigation in aerospace engineering, and the method is marginally more computationally demanding than a Kalman filter there shall be no problem. Note that, however, training the network may take much time but of course this should be done once beforehand.

7 7 Conclusion

This chapter proposes a novel approach for inertial only dead-reckoning for wheeled vehicles which builds upon deep neural networks to dynamically adapt the parameters of a Kalman filter. We have shown the following facts. 1) It is possible to obtain surprisingly accurate results using only a moderate cost IMU, thanks to the use of a Kalman filter that combines standard IMU equations with side information about dynamics of wheeled vehicles. The algorithm competes with vision based methods, although only the IMU is used (and not a single other sensor, such as GNSS). 2) Deep neural networks are a powerful tool for dynamic adaptation of Kalman filter tuning parameters (noise covariance matrices). 3) Beyond deep neural nets, correctly assessing measurement covariance dynamically allows Kalman filters to achieve much better performance, and this opens avenues for fusion with other sensors. The subject of generalization, and notably how the network architecture may be reused in similar applications, is left for future research, as tuning CNNs represents in itself a current research field. That said, the code we made publicly available may be used as is, and adapted to other vehicles. Moreover, as mentioned in Section 6.6, the chapter proves that dynamic covariance adaptation plays a huge role for accurate localization, and simple practical engineered adaptation rules might be pursued instead of AI-based ones. Futures works would adress learning to adapt uncertainties in a robust (non-Gaussian) context and the question of generalization, i.e. how a neural network trained on a platform may be used in another platform.

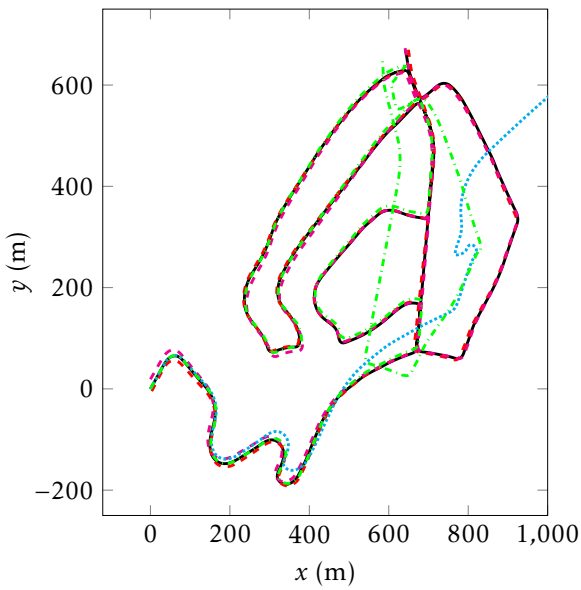


Figure 8.9: Results on seq. 02 (drive #34, 2011/09/30) [101]. The proposed method competes with LiDAR and visual odometry methods until a problem in data occurs (2 seconds are missing). It is remarkable that the proposed method be robust to such trouble causing a shift estimates but no divergence.

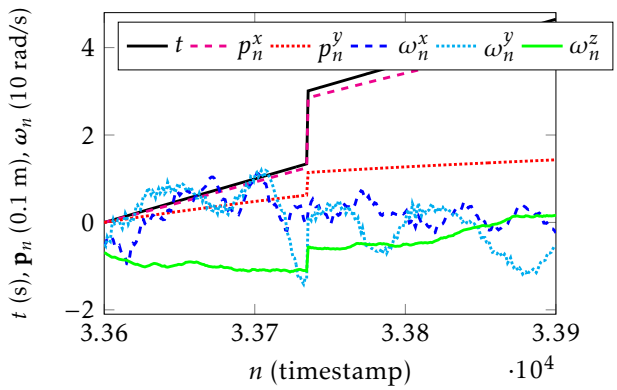


Figure 8.10: Data on seq. 02 (drive #34, 2011/09/30) [101], as function of timestamps number n. A 2 s time jump happen around $n = 33750$, i.e. data have not been recorded during this jump. It leads to jump in ground-truth and IMU signals, causing estimate drift.

Appendix A

The IEKF [42,43] is an EKF based on an alternative state error, see Chapter 2. One must define a linearized error, an underlying group to derive the exponential map, and then the methodology is akin to the EKF's, see Chapter 2.

Linearized Error: the filter state χ_n is given by (8.8). The state evolution is given by the dynamics (8.5)-(8.7) and (8.12)-(8.13), see Section 3. Along the lines of [42], variables $\chi_n^{\text{IMU}} := (\mathbf{R}_n^{\text{IMU}}, \mathbf{v}_n^{\text{IMU}}, \mathbf{p}_n^{\text{IMU}})$ are embedded in the Lie group $SE_2(3)$. Then biases vector $\mathbf{b}_n = [\mathbf{b}_n^{\omega T}, \mathbf{b}_n^{\mathbf{a}T}]^T \in \mathbb{R}^6$ is merely treated as a vector, that is, as an element of $\mathbb{R}^6$ viewed as a Lie group endowed with standard addition, $\mathbf{R}_n^{\text{c}}$ is treated as element of Lie group $SO(3)$, and $\mathbf{p}_n^{\text{c}} \in \mathbb{R}^3$ as a vector. Once the state is broken into several Lie groups, the linearized error writes as the concatenation of corresponding linearized errors, that is,

$$\mathbf{e}_n = \begin{bmatrix} \xi_n^{\text{IMU}T} & \mathbf{e}_n^{\mathbf{b}T} & \xi_n^{\mathbf{R}^{\text{c}}T} & \mathbf{e}_n^{\mathbf{p}^{\text{c}}T} \end{bmatrix}^T \sim \mathcal{N}(\mathbf{0}, \mathbf{P}_n), \tag{8.21}$$

where state uncertainty $\mathbf{e}_n \in \mathbb{R}^{21}$ is a zero-mean Gaussian variable with covariance $\mathbf{P}_n \in \mathbb{R}^{21\times 21}$. As (8.15) are measurements expressed in the robot's frame, they lend themselves to the Right IEKF methodology. This means each linearized error is mapped to the state using the corresponding Lie group exponential map, and multiplying it *on the right* by elements of the state space. This yields:

$$\chi_n^{\text{IMU}} = \exp_{SE_2(3)}(\xi_n^{\text{IMU}})\hat{\chi}_n^{\text{IMU}}, \tag{8.22}$$

$$\mathbf{b}_n = \hat{\mathbf{b}}_n + \mathbf{e}_n^{\mathbf{b}}, \tag{8.23}$$

$$\mathbf{R}_n^{\text{c}} = \exp_{SO(3)}\left(\xi_n^{\mathbf{R}^{\text{c}}}\right)\hat{\mathbf{R}}_n^{\text{c}}, \tag{8.24}$$

$$\mathbf{p}_n^{\text{c}} = \hat{\mathbf{p}}_n^{\text{c}} + \mathbf{e}_n^{\mathbf{p}^{\text{c}}}, \tag{8.25}$$

where $\hat{(\cdot)}$ denotes estimated state variables.

Propagation Step: we apply (8.5)-(8.7) and (8.12)-(8.13) to propagate the state and obtain $\hat{\chi}_{n+1}$ and associated covariance through the Riccati equation (8.11) where the Jacobians $\mathbf{F}_n$, $\mathbf{G}_n$ are related to the evolution of error (8.21) and write:

$$\mathbf{F}_n = \mathbf{I}_{21} + \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathbf{R}_n & \mathbf{0} & \mathbf{0}_{3\times 6} \\ \mathbf{g}_\times & \mathbf{0} & \mathbf{0} & -(\mathbf{v}_n^{\text{IMU}})_\times \mathbf{R}_n & -\mathbf{R}_n & \mathbf{0}_{3\times 6} \\ \mathbf{0} & \mathbf{I}_3 & \mathbf{0} & -(\mathbf{p}_n^{\text{IMU}})_\times \mathbf{R}_n & \mathbf{0} & \mathbf{0}_{3\times 6} \\ & & & \mathbf{0}_{12\times 21} & & \end{bmatrix} dt, \tag{8.26}$$

$$\mathbf{G}_n = \begin{bmatrix} \mathbf{R}_n & \mathbf{0} & \mathbf{0}_{3\times 12} \\ (\mathbf{v}_n^{\text{IMU}})_\times \mathbf{R}_n & \mathbf{R}_n & \mathbf{0}_{3\times 12} \\ (\mathbf{p}_n^{\text{IMU}})_\times \mathbf{R}_n & \mathbf{0} & \mathbf{0}_{3\times 12} \\ \mathbf{0}_{12\times 3} & \mathbf{0}_{12\times 3} & \mathbf{I}_{12} \end{bmatrix} dt, \tag{8.27}$$

with $\mathbf{R}_n = \mathbf{R}_n^{\text{IMU}}$, $\mathbf{0} = \mathbf{0}_{3\times 3}$, and $\mathbf{Q}_n$ denotes the classical covariance matrix of the process noise as in Section 5.2.

Update Step: the measurement vector $\mathbf{y}_{n+1}$ is computed by stacking the motion information

$$\mathbf{y}_{n+1} = \begin{bmatrix} v_{n+1}^{\text{lat}} \\ v_{n+1}^{\text{up}} \end{bmatrix} = \mathbf{0}, \tag{8.28}$$

with assessed uncertainty a zero-mean Gaussian variable with covariance $\mathbf{N}_{n+1} = \text{cov}(\mathbf{y}_{n+1})$. We then compute an updated state $\hat{\chi}_{n+1}^{+}$ and updated covariance $\mathbf{P}_{n+1}^{+}$ following the IEKF methodology, i.e. we compute

$$\mathbf{S} = \left(\mathbf{H}_{n+1}\mathbf{P}_{n+1}\mathbf{H}_{n+1}^{T} + \mathbf{N}_{n+1}\right), \tag{8.29}$$

$$\mathbf{K} = \mathbf{P}_{n+1}\mathbf{H}_{n+1}^{T}/\mathbf{S}, \tag{8.30}$$

$$\mathbf{e}^{+} = \mathbf{K}(\mathbf{y}_{n+1} - \hat{\mathbf{y}}_{n+1}), \tag{8.31}$$

$$\hat{\chi}_{n+1}^{\text{IMU}+} = \exp_{SE_2(3)}(\xi^{\text{IMU}+})\,\hat{\chi}_{n+1}^{\text{IMU}}, \tag{8.32}$$

$$\mathbf{b}_{n+1}^{+} = \mathbf{b}_{n+1} + \mathbf{e}^{\mathbf{b}+} \tag{8.33}$$

$$\hat{\mathbf{R}}_{n+1}^{\text{c}+} = \exp_{SO(3)}\left(\xi^{\mathbf{R}^{\text{c}}+}\right)\hat{\mathbf{R}}_{n+1}^{\text{c}}, \tag{8.34}$$

$$\hat{\mathbf{p}}_{n+1}^{\text{c}+} = \hat{\mathbf{p}}_{n+1}^{\text{c}} + \mathbf{e}^{\mathbf{p}^{\text{c}}+}, \tag{8.35}$$

$$\mathbf{P}_{n+1}^{+} = (\mathbf{I}_{21} - \mathbf{K}\mathbf{H}_{n+1})\,\mathbf{P}_{n+1}, \tag{8.36}$$

summarized as Kalman gain (8.30), state innovation (8.31), state update (8.32)-(8.35) and covariance update (8.36), where $\mathbf{H}_{n+1}$ is the measurement Jacobian matrix with respect to linearized error (8.21) and thus given as:

$$\mathbf{H}_n = \mathbf{A}\begin{bmatrix}\mathbf{0} & \mathbf{R}_n^{\text{IMU}T} & \mathbf{0} & -(\mathbf{p}_n^{\text{c}})_\times & \mathbf{0} & \mathbf{B} & \mathbf{C}\end{bmatrix}, \tag{8.37}$$

where $\mathbf{A} = [\mathbf{I}_2\ \mathbf{0}_2]\,\mathbf{R}_n^{\text{c}T}$ selects the two first row of the right part of (8.37), $\mathbf{B} = \mathbf{R}_n^{\text{IMU}T}\,(\mathbf{v}_n^{\text{IMU}} + (\omega_n^{\text{IMU}} - \mathbf{b}_n^{\omega})\,\mathbf{p}_n^{\text{c}})_\times$ and $\mathbf{C} = -(\omega_n^{\text{IMU}} - \mathbf{b}_n^{\omega})_\times$.

CHAPTER 9

A New Approach to 3D ICP Covariance Estimation

The present chapter has been published in IEEE Robotics and Automation Letters.

Résumé 🇫🇷

Dans ce chapitre, nous adressons le problème d'estimer la covariance de l'algorithme ICP, covariance qui peut ensuite être donnée comme covariance de l'observation à un filtre de Kalman où un algorithme d'optimisation pour la fusion de données En effet, en robotique mobile, la recherche de correspondance entre des nuages de points en utilisant l'algorithme itératif ICP permet d'estimer les déplacements succesifs du capteur, et il peut s'avérer important d'évaluer l'incertitude associée à la transformation obtenue, notamment à des fins de fusion de capteurs. Dans ce chapitre, nous proposons une nouvelle approche de l'estimation de l'incertitude 3D de l'ICP qui tient compte de toutes les sources d'erreur énumérées dans les travaux pionniers de Censi [169], à savoir une convergence vers un minimum local, des situations sous faibles contraintes, et le bruit du capteur. Notre approche repose sur deux faits. Premièrement, l'incertitude de l'ICP dépend entièrement de l'incertitude sur la transformation initiale. Ainsi parler de la covariance de l'ICP n'a de sens que par rapport à l'incertitude d'initialisation, qui découle généralement des erreurs d'odométrie. Nous capturons cela en utilisant la transformation sans-parfum, qui reflète également la corrélation entre les incertitudes initiales et finales. Ensuite, le bruit supossé blanc du capteur conduit à une incertitude car l'ICP est biaisé en raison par exemple du biais du capteur que nous expliquons. Notre solution est testée sur des données réelles accessible mellant des environnements structurés et non structurés, où notre algorithme prédit des résultats cohérents avec une incertitude réelle et se compare favorablement aux méthodes précédentes.

Chapter abstract

In the present chapter we address the problem of estimating the covariance of the Iterative Closest Point (ICP) algorithm, which can be fed in a Kalman filter or an optimization-based sensor-fusion algorithm as the measurement covariance. Indeed, in mobile robotics, scan matching of point clouds using ICP allows estimating sensor displacements, and it may prove important to assess the associated uncertainty about the obtained rigid transformation, especially for sensor fusion purposes. In this chapter we propose a novel approach to 3D uncertainty of ICP that accounts for all

the sources of error as listed in Censi's pioneering work [169], namely wrong convergence, underconstrained situations, and sensor noise. Our approach builds on two facts. First, the uncertainty about the ICP's output fully depends on the initialization accuracy. Thus speaking of the covariance of ICP makes sense only in relation to the initialization uncertainty, which generally stems from odometry errors. We capture this using the unscented transform, which also reflects correlations between initial and final uncertainties. Then, assuming white sensor noise leads to overoptimism as ICP is biased owing to e.g. calibration biases, which we account for. Our solution is tested on publicly available real data ranging from structured to unstructured environments, where our algorithm predicts consistent results with actual uncertainty, and compares favorably to previous methods.

1 1 Introduction

Point clouds and the Iterative Closest Point (ICP) algorithm play a crucial role for localization and mapping in modern mobile robotics [170,171]. ICP computes an estimate of the 3D rigid transformation that aligns a reading point cloud to a reference point cloud (or more generally a model or a surface). The algorithm starts with a first transformation estimate, and repeats - until convergence - point association and least-square minimization, where initialization is naturally provided in mobile robotics by odometry [172,173] based on wheel speeds, inertial sensors, or vision. The point association matches points between the two clouds by generally associating each point of the second cloud to its closest point in the first one. Then, the algorithm minimizes a user-chosen metric between the matched points that provides an update of the current estimate. In spite of robust filtering that are broadly used during the alignment of point clouds, a.k.a. registration, ICP is subject to errors stemming from sensor noises, underconstrained environments that result in unobservable directions, and local minima [169,174,175].

1.1 1.1 Sources of ICP Uncertainty

The pioneering work of Censi [169] identifies the following sources of error for ICP registration: wrong convergence (not handled by Censi's formula), underconstrained situations, and sensor noise. As indicated by preliminary remarks in [178,179] we believe a fourth important source is missing: the one that stems from sensor biases. In the present chapter we consider indeed the following sources of error:

Initial Transformation: ICP is subject to error due to wrong initialization that makes the algorithm converge to a local minimum out of the attraction basin of the true solution, as largely observed in practice, see e.g. [175,180] and Figure 9.1. In practice it often proves to be the dominant error, where very recent works adress this problem for obtaining "certifiable" solvers, such as TEASER [181], which claims to be able to either find the global optimum alignment between two pointclouds or report an error if it fails to do so, even with a very high rate of outlier measurements.

Sensor White Noise: each *point* measured in a point cloud is affected by an *independent* random sensor noise of centimetric magnitude which is a function of point depth and beam angle [174,182].

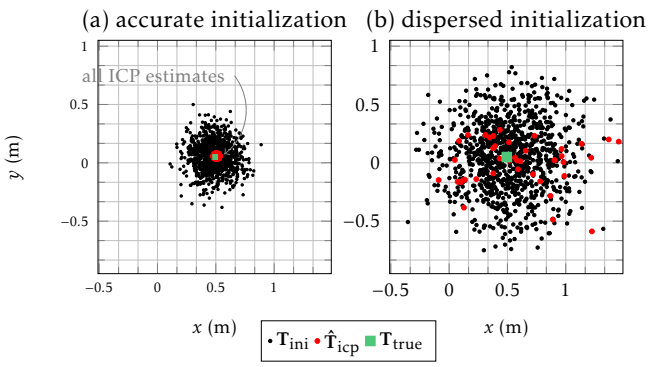


Figure 9.1: Horizontal translation estimates according to ICP ($\hat{\mathbf{T}}_{\text{icp}}$, red dots) for various initial estimates ($\mathbf{T}_{\text{ini}}$, black dots) and ground-truth ($\mathbf{T}_{\text{true}}$, square) for registering two scans of the sequence *Stairs* of [176], where we sample 1000 initial estimates from two distributions reflecting accurate (a) and dispersed (b) ICP initialization and that respectively correspond to the easy and medium scenarios of [177]. We see the uncertainty on the ICP estimate, that is, dispersion of red points, wholly depends on the accuracy of initialization. There is no "uncertainty of ICP" *per se*.

Sensor Bias Noise: the observed points share common errors that stem from: temperature drift effect, i.e. stability of the laser [182]; observed material [174]; incidence and beam angles resulting in large bias [183]; or wrong calibration, e.g. [135] found a distortion of 0.22 deg of the scan point clouds due to intrinsic calibration process. This *correlated* noise, a.k.a. bias, strictly limits the confidence we may have in the ICP estimate. To our best knowledge this is often omitted with a few exceptions: e.g., [183] removes bias on point measurements due to sensor beam angle, and preliminary ideas may be found in [178,179].

Randomness Inherent to the ICP Algorithm: ICP is generally configured with random filtering processes [170], e.g. sub-sampling, such that two solutions with exactly the same inputs would differ.

In the following we address uncertainty coming from 1), 2) and 3) and do not consider 4), which should be marginal.

1.2 1.2 Brief Literature Review

Various approaches exist for estimating the covariance of the ICP algorithm, each of which being a trade-off between accuracy and execution time. Monte-Carlo algorithms, e.g. [180,184], sample noisy scans (from a reference scan) and ICP initializations to compute a large number of ICP registration results, define the covariance of the sampled results as the covariance estimation, and use the estimated covariance for all future registration with the reference scan, thus getting a covariance function of the reference scan only. Another category of covariance estimation methods relies on closed-form expressions [169,185–187], whose underlying assumption consists in linearizing the objective function used in ICP around the convergence point, ruling out the possibility for wrong convergence and the uncertainty that stems from it. Albeit still used in practice, Censi's pioneering formula [169] is widely considered as overoptimistic, see e.g. [188]. Recently, [175] leveraged learning based approaches to estimate ICP uncertainty stemming from inaccurate ICP initialization.

1.3 1.3 Contributions and Chapter's Organization

Our approach introduced in Section 2 extends existing works in three ways: 1) we consider ICP uncertainty coming both from sensor errors and ICP initialization. 2) we raise an important point which is that ICP uncertainty in itself is meaningless as it is inherently related to uncertainty in the initialization pose (unless there is a global minimum). This is supported by experiments displayed in Figure 9.1. We address this problem by outputting a covariance matrix of larger dimension that also reflects the correlation between ICP final and initial estimates. And 3) we estimate in Section 3 the ICP uncertainty combining a closed-form expression using [169,186] accounting for sensor biases, and derivative-free methods using the unscented transform of [60,189], which comes at a lower computational cost than Monte-Carlo runs.

Besides, we evaluate, compare and discuss our approach on the dataset of [176] in Sections 4 and 5, where our approach obtains consistent estimates and achieves better results than existing methods. The code to reproduce the results of the chapter is made publicly available at: `https://github.com/CAOR-MINES-ParisTech/3d-icp-cov`.

Throughout the chapter, we configurate the ICP as suggested in [177] with a point-to-plane error metric.

2 2 Proposed Approach

2.1 2.1 Pose Representation and Pose Uncertainty Representation

The true transformation between two point clouds and its ICP-based estimate both live in the set of 3D rigid transformations $SE(3)$, see Chapter 2. Note that, it is consistent with matrix multiplication: if $\mathbf{T}_1$ transforms a first point cloud into a second one, and then $\mathbf{T}_2$ transforms the latter into a third cloud, then the matrix $\mathbf{T}_2\mathbf{T}_1 \in SE(3)$ encodes the transformation between the first and the third clouds.

We use the uncertainty representation for poses as

$$\hat{\mathbf{T}} = \mathbf{T}\exp(\xi), \text{ where } \xi \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}), \xi \in \mathbb{R}^6, \tag{9.1}$$

with ξ a zero-mean Gaussian variable of covariance $\mathbf{Q}$ and where $\exp(\cdot)$ denotes the exponential map of $SE(3)$, which maps elements ξ to poses. For uncertainty representation (9.1), we adopt the notation $\hat{\mathbf{T}} \sim \mathcal{N}_L(\mathbf{T}, \mathbf{Q})$. Note that, in (9.1), the vector $\xi \in \mathbb{R}^6$ may be viewed as the *error* between $\mathbf{T}$ and $\hat{\mathbf{T}}$. Indeed the relative transformation between poses $\mathbf{T}$ and $\hat{\mathbf{T}}$ is encoded in ξ as $\mathbf{T}^{-1}\hat{\mathbf{T}} = \exp(\xi)$.

2.2 2.2 The Role of ICP Initialization

The ICP procedure seeks to estimate the transformation $\mathbf{T}_{\text{true}} \in SE(3)$ that maps a first cloud of points $\mathcal{P}$ to a second cloud (or a model) $\mathcal{Q}$ as follows [170,171]:

i) we have a first "guess" for the transformation we call $\mathbf{T}_{\text{ini}}$, a.k.a. initial or coarse alignment [171];

ii) then we initialize the ICP algorithm by applying a transformation $\mathbf{T}_{\text{ini}}^{-1}$ to the cloud $\mathcal{Q}$. This way the transformation the ICP seeks to estimate become the relative pose $\mathbf{T}_{\text{rel}} := \mathbf{T}_{\text{ini}}^{-1}\mathbf{T}_{\text{true}}$. We thus get an estimate $\hat{\mathbf{T}}_{\text{rel}} = \text{icp}\left(\mathcal{P}, \mathbf{T}_{\text{ini}}^{-1}\mathcal{Q}\right)$ for $\mathbf{T}_{\text{rel}}$;

iii) finally the estimate of $\mathbf{T}_{\text{true}}$ that the algorithm outputs is $\hat{\mathbf{T}}_{\text{icp}} := \mathbf{T}_{\text{ini}}\hat{\mathbf{T}}_{\text{rel}} = \mathbf{T}_{\text{ini}}\text{icp}\left(\mathcal{P}, \mathbf{T}_{\text{ini}}^{-1}\mathcal{Q}\right)$.

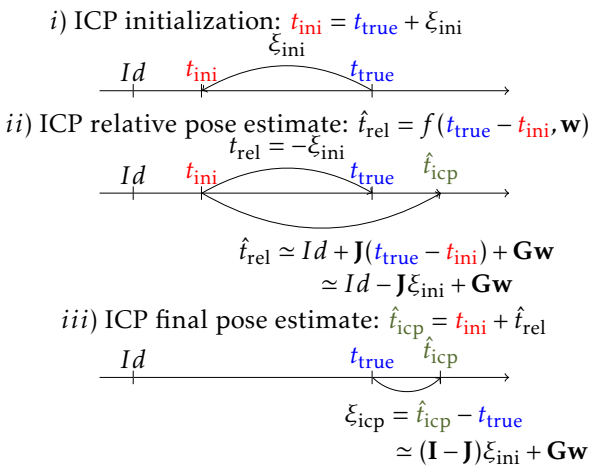


Figure 9.2: Schematic illustration of the ICP procedure, error definitions and linearizations in the case of 1D translation $t \in \mathbb{R}$ ($Id = 0$).

Note that *if* $\mathbf{T}_{\text{rel}}$ is perfectly estimated we recover $\mathbf{T}_{\text{true}}$ as then $\hat{\mathbf{T}}_{\text{icp}} = \mathbf{T}_{\text{ini}}\mathbf{T}_{\text{rel}} = \mathbf{T}_{\text{ini}}\mathbf{T}_{\text{ini}}^{-1}\mathbf{T}_{\text{true}} = \mathbf{T}_{\text{true}}$ no matter how far the initial guess $\mathbf{T}_{\text{ini}}$ is from $\mathbf{T}_{\text{true}}$.

Let us introduce the various errors at play in ICP. In robotics, the initial guess in i) is typically provided through inertial sensors or wheeled odometry [172,173]. We have thus an initialization error that stems from sensor imperfections, encoded by a vector ξ_{ini}, and one may write

$$\mathbf{T}_{\text{ini}} = \mathbf{T}_{\text{true}} \exp(\xi_{\text{ini}}),\ \xi_{\text{ini}} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_{\text{ini}}), \tag{9.2}$$

which is advocated in [90,190] to suit particularly well represent odometry errors in terms of pose. Then, ICP estimates the relative transformation between $\mathbf{T}_{\text{true}}$ and $\mathbf{T}_{\text{ini}}$, that is, outputs an estimate $\hat{\mathbf{T}}_{\text{rel}}$ of the actual initial error $\mathbf{T}_{\text{rel}}$ which writes

$$\begin{aligned}\mathbf{T}_{\text{rel}} &= \mathbf{T}_{\text{ini}}^{-1}\mathbf{T}_{\text{true}} = \exp(-\xi_{\text{ini}})\mathbf{T}_{\text{true}}^{-1}\mathbf{T}_{\text{true}} \\ &= \exp(-\xi_{\text{ini}}) \simeq \mathbf{I}_4 - \xi_{\text{ini}}^{\wedge}.\end{aligned} \tag{9.3}$$

2.3 2.3 ICP Estimate $\hat{\mathbf{T}}_{\text{rel}}$ of Relative Pose $\mathbf{T}_{\text{rel}}$

At step ii) above, that is, once initialization is done, see [170], ICP provides an estimate $\hat{\mathbf{T}}_{\text{rel}}$ of the relative transformation $\mathbf{T}_{\text{rel}}$ of (9.3) as a function

$$\hat{\mathbf{T}}_{\text{rel}} := \text{icp}\left(\mathcal{P}, \mathbf{T}_{\text{ini}}^{-1}\mathcal{Q}\right) \tag{9.4}$$

of the point clouds $\mathcal{P}$ and $\mathcal{Q}$. Thus $\hat{\mathbf{T}}_{\text{rel}}$ appears as a function $f(\mathbf{T}_{\text{rel}}) \in SE(3)$ of the true relative transformation $\mathbf{T}_{\text{rel}}$, typically affected by the phenomena of wrong convergence. Moreover, sensor (scanner) noise induces small fluctuations in the point clouds that affects this estimation. This yields:

$$\hat{\mathbf{T}}_{\text{rel}} = f(\mathbf{T}_{\text{rel}})\exp(\mathbf{G}\mathbf{w}) = f\left(\exp(-\xi_{\text{ini}})\right)\exp(\mathbf{G}\mathbf{w}), \tag{9.5}$$

where $\mathbf{w} \in \mathbb{R}^{6K}$ encodes errors due to sensor noise on each of the K pairs of points in the clouds, and $\mathbf{G}\mathbf{w} \in \mathbb{R}^6$ the resulting 6 degrees of freedom error made on $\mathbf{T}_{\text{rel}}$. Sensor

noise stems from unknown parameters that depend upon the calibration process and drift with temperature [182]. If the ICP is initialized on the true pose $\mathbf{T}_{\text{true}}$ then there is no wrong convergence and the only error stems from noise, i.e., $f(\exp(\mathbf{0}) = f(\mathbf{I}_4) = \mathbf{I}_4$. Thus $f(\cdot) \in SE(3)$ is close to $\mathbf{I}_4$, and the model may be linearized around $\xi_{\text{ini}} = \mathbf{0}$, $\mathbf{w} = \mathbf{0}$ as

$$\begin{aligned} f\Big(\exp(-\xi_{\text{ini}})\Big)\exp(\mathbf{G}\mathbf{w}) &\simeq f(\mathbf{I}_4 - \xi_{\text{ini}}^{\wedge})\exp(\mathbf{G}\mathbf{w}) \\ &\simeq \mathbf{I}_4 + (-\mathbf{J}\xi_{\text{ini}} + \mathbf{G}\mathbf{w})^{\wedge}, \end{aligned} \tag{9.6}$$

where matrix $\mathbf{J}$ encodes the linear approximation of $f(\cdot)$.

2.4 2.4 ICP Final Pose Error

Let us now consider step iii) of the ICP algorithm, i.e., the final estimate

$$\hat{\mathbf{T}}_{\text{icp}} = \mathbf{T}_{\text{ini}}\hat{\mathbf{T}}_{\text{rel}} \tag{9.7}$$

$$= \mathbf{T}_{\text{true}}\exp(\xi_{\text{ini}})f\Big(\exp(-\xi_{\text{ini}})\Big)\exp(\mathbf{G}\mathbf{w}). \tag{9.8}$$

(9.8) was obtained substituting (9.2) and (9.5) in (9.7). Linearizing (9.8) by recalling (9.6) and keeping only the first order in the small errors ξ_{ini} and $\mathbf{w}$ yields $\hat{\mathbf{T}}_{\text{icp}} \simeq \mathbf{T}_{\text{true}}\left[\mathbf{I}_4 + (\xi_{\text{ini}})^{\wedge}\right]\Big[\mathbf{I}_4 + (-\mathbf{J}\xi_{\text{ini}} + \mathbf{G}\mathbf{w}))^{\wedge}\Big] \simeq \mathbf{T}_{\text{true}}\Big(\mathbf{I}_4 + (\xi_{\text{ini}} - \mathbf{J}\xi_{\text{ini}} + \mathbf{G}\mathbf{w})^{\wedge}\Big)$ and thus in terms of uncertainty representation (9.1) we approximately find:

$$\boxed{\hat{\mathbf{T}}_{\text{icp}} \simeq \mathbf{T}_{\text{true}}\exp\left((\mathbf{I}_6 - \mathbf{J})\xi_{\text{ini}} + \mathbf{G}\mathbf{w}\right).} \tag{9.9}$$

Figure 9.2 recaps the computations for 1D translations. There are a couple of situations of interest. Let us momentarily assume sensor noise to be turned off, $\mathbf{w} = \mathbf{0}$, for simplicity.

- If there is one global minimum, then the ICP systematically recovers the relative transformation (9.3) at step ii) of the algorithm, i.e. $\hat{\mathbf{T}}_{\text{rel}} = \mathbf{T}_{\text{rel}}$ and thus $f(\mathbf{T}_{\text{rel}}) = \mathbf{T}_{\text{rel}}$. So $f(\exp(-\xi_{\text{ini}})) \simeq \mathbf{I}_4 - \xi_{\text{ini}}^{\wedge}$ and we identify $\mathbf{J} = \mathbf{I}_6$ in this case. As a result the final estimate (9.9) is $\mathbf{T}_{\text{true}}\exp(\mathbf{0}) = \mathbf{T}_{\text{true}}$ indeed.

- On the other hand, in the directions where we have no information, e.g. along hallways or in underconstrained environment [169], the relative transformation will not be affected in the corresponding directions meaning that (along those directions) $\mathbf{J} = \mathbf{0}$ and the final error then has the form $\mathbf{T}_{\text{true}}^{-1}\hat{\mathbf{T}}_{\text{icp}} = \mathbf{T}_{\text{true}}^{-1}\mathbf{T}_{\text{true}}\exp(\xi_{\text{ini}}) = \exp(\xi_{\text{ini}})$, that is, the initialization error fully remains.

In intermediate cases (when there are local minima) the remaining error is a fraction $\mathbf{J}\xi_{\text{ini}}$ of the initialization error.

2.5 2.5 Corresponding ICP Error Covariance

If we represent ICP uncertainties resorting to concentrated Gaussian (9.1) as $\hat{\mathbf{T}}_{\text{icp}} \sim \mathcal{N}_L\Big(\mathbf{T}_{\text{true}}, \mathbf{Q}_{\text{icp}}\Big)$, i.e., we posit $\mathbf{T}_{\text{true}}^{-1}\hat{\mathbf{T}}_{\text{icp}} = \exp(\xi_{\text{icp}})$, then the covariance matrix $\mathbf{Q}_{\text{icp}}$ of ξ_{icp} describes dispersion (hence uncertainty) of the ICP error. Plugging the latter representation into (9.9), we have $\xi_{\text{icp}} = (\mathbf{I}_6 - \mathbf{J})\xi_{\text{ini}} + \mathbf{G}\mathbf{w}$. As the initialization error is assumed to have covariance matrix $\mathbf{Q}_{\text{ini}}$ typically inferred though an odometry error model [90,190] and by denoting $\mathbf{Q}_{\text{sensor}}$ the covariance of scan sensor noise

$\mathbf{w}$, the covariances add up owing to independence of sensor noises, and by squaring $\xi_{\text{icp}} = (\mathbf{I}_6 - \mathbf{J})\xi_{\text{ini}} + \mathbf{G}\mathbf{w}$ we find

$$\boxed{\mathbf{Q}_{\text{icp}} = (\mathbf{I}_6 - \mathbf{J})\mathbf{Q}_{\text{ini}}(\mathbf{I}_6 - \mathbf{J})^T + \mathbf{G}\mathbf{Q}_{\text{sensor}}\mathbf{G}^T.} \tag{9.10}$$

This is our first result about ICP covariance. The first term related to the initialization uncertainty and that accounts for wrong convergence, lack of constraints in the clouds and unobservable directions, and the second one related to scan noise and that may be computed through "Censi-like" [169] formulas as we will show in section 3.

2.6 2.6 Discussion

Albeit not obvious, $\mathbf{J}$ actually heavily depends on $\mathbf{Q}_{\text{ini}}$. This is an insight of the present chapter: *uncertainty of ICP does not exist in itself.* Assume indeed there are various local minima. If $\mathbf{Q}_{\text{ini}}$ is very small, then all initializations $\mathbf{T}_{\text{ini}}$ fall within the attraction basin of $\mathbf{T}_{\text{true}}$ and thus $f(\mathbf{T}_{\text{rel}}) = \mathbf{T}_{\text{rel}}$ and we identify $\mathbf{J} = \mathbf{I}_6$. But if $\mathbf{Q}_{\text{ini}}$ is large enough only a fraction of initializations $\mathbf{T}_{\text{ini}}$ lead to $f(\mathbf{T}_{\text{rel}}) = \mathbf{T}_{\text{rel}}$, the ones that get trapped in other local minimas do not lead to correct estimate of $\mathbf{T}_{\text{rel}}$ and $\mathbf{J} \neq \mathbf{I}_6$. Thus $\mathbf{J}$ is not the analytical Jacobian of function $f(\cdot)$, and may be viewed as its "statistical linearization" [191]. This prompts the use of an unscented transform [189] to compute it, see Section 3.2.

2.7 2.7 Maximum Likelihood Fusion of Initial and ICP Estimates

$\mathbf{T}_{\text{ini}}$ and $\hat{\mathbf{T}}_{\text{icp}}$ may be viewed as two estimates of $\mathbf{T}_{\text{true}}$ associated with uncertainty respectively $\mathbf{Q}_{\text{ini}} = \text{cov}(\xi_{\text{ini}})$ and $\mathbf{Q}_{\text{icp}} = \text{cov}(\xi_{\text{icp}})$ where $\xi_{\text{icp}} = (\mathbf{I} - \mathbf{J})\xi_{\text{ini}} + \mathbf{G}\mathbf{w}$. The corresponding pose fusion problem of finding the Maximum Likelihood (ML) of a pose $\mathbf{T}_{\text{true}}$ given two uncertain pose estimates was considered in [90], with the important difference that herein ξ_{ini} and ξ_{icp} are not independent, they are *correlated*, with joint matrix of initialization and ICP errors

$$\mathbf{Q} := \text{cov}\left(\begin{bmatrix}\xi_{\text{ini}} \\ \xi_{\text{icp}}\end{bmatrix}\right) = \begin{bmatrix}\mathbf{Q}_{\text{ini}} & \mathbf{Q}_{\text{ini}}(\mathbf{I} - \mathbf{J})^T \\ (\mathbf{I} - \mathbf{J})\mathbf{Q}_{\text{ini}} & \mathbf{Q}_{\text{icp}}\end{bmatrix}. \tag{9.11}$$

Using linearization as previously and following first-order computations in [90,192], the maximum likelihood estimate of $\mathbf{T}_{\text{true}}$ may be approximated as $\hat{\mathbf{T}}_{\text{ML}} = \mathbf{T}_{\text{true}} \exp(\xi_{\text{ML}})$ with $\text{cov}(\xi_{\text{ML}}) = \mathbf{Q}_{\text{ML}}$, with $\mathbf{Q}_{\text{ML}}$ defined through its inverse:

$$\mathbf{Q}_{\text{ML}}^{-1} = \begin{bmatrix}\mathbf{I} \\ \mathbf{I}\end{bmatrix}^T \begin{bmatrix}\mathbf{Q}_{\text{ini}} & \mathbf{Q}_{\text{ini}}(\mathbf{I} - \mathbf{J})^T \\ (\mathbf{I} - \mathbf{J})\mathbf{Q}_{\text{ini}} & \mathbf{Q}_{\text{icp}}\end{bmatrix}^{-1} \begin{bmatrix}\mathbf{I} \\ \mathbf{I}\end{bmatrix}. \tag{9.12}$$

The latter stems from classical linear estimation theory and may be proved using the Kalman information filter: to the first order $\hat{\mathbf{T}}_{\text{ini}}$ and $\hat{\mathbf{T}}_{\text{icp}}$ are considered as two noisy measurements of $\mathbf{T}_{\text{true}}$ with joint covariance (9.11), the measurement matrix is thus $\mathbf{H} := \begin{bmatrix}\mathbf{I} & \mathbf{I}\end{bmatrix}^T$ and as $\mathbf{T}_{\text{true}}$ is initially totally unknown the prior covariance satisfies $\mathbf{P}^{-1} = \mathbf{0}$. The covariance of the Kalman estimate in the light of measurements is thus updated in information form as $\mathbf{P}^{-1} \leftarrow \mathbf{0} + \mathbf{H}^T\mathbf{Q}^{-1}\mathbf{H}$.

3 3 Practical Covariance Computation

This section describes our algorithm for estimating the 3D ICP uncertainty covariance (9.10) leveraging findings of Section 2. We propose to first compute the rightmost term of (9.10) which is due to sensor noise.

3.1 3.1 Computation of Dispersion Owing to Sensor Noise

We now focus on the computation of $\mathbf{G}\mathbf{Q}_{\text{sensor}}\mathbf{G}^T$. The cost function of point-to-plane ICP after initialization writes $J_{\hat{\mathbf{T}}_{\text{rel}}}\left(\mathcal{P}, \mathbf{T}_{\text{ini}}^{-1}\mathcal{Q}\right) = \sum_{k=1}^{K} \|\left(\hat{\mathbf{T}}_{\text{rel}} p_k - \tilde{q}_k\right) \cdot n_k\|^2$, where the $\tilde{q}_k$'s denote the points of $\mathbf{T}_{\text{ini}}^{-1}\mathcal{Q}$ and K is the number of pairs of matched points. Linearizing on $SE(3)$, we may linearize the cost $J_{\hat{\mathbf{T}}_{\text{rel}}\exp(\xi)}\left(\mathcal{P}, \mathbf{T}_{\text{ini}}^{-1}\mathcal{Q}\right) = \sum_{k=1}^{K} \|\left(\hat{\mathbf{T}}_{\text{rel}} \exp(\xi)\, p_k - \tilde{q}_k\right) \cdot n_k\|^2$ w.r.t. estimate $\hat{\mathbf{T}}_{\text{rel}}$ at $\hat{\mathbf{T}}_{\text{rel}} = \mathbf{T}_{\text{rel}}$ as

$$J_{\hat{\mathbf{T}}_{\text{rel}}\exp(\xi)}\left(\mathcal{P}, \mathbf{T}_{\text{ini}}^{-1}\mathcal{Q}\right) \simeq \sum_{k=1}^{K} \|\mathbf{B}_k \xi - d_k\|^2, \tag{9.13}$$

with d_k a scalar being function of differences between pairs of points and point normals. Least squares formulas yield an optimal value $\xi^* = \mathbf{A}^{-1} \sum_{k=1}^{K} \mathbf{B}_k^T d_k$, where we let $\mathbf{A} = \sum_{k=1}^{K} \mathbf{B}_k^T \mathbf{B}_k$. Each d_k is affected by k-th component w_i of previously introduced sensor noise $\mathbf{w}$, and this induces fluctuations in ξ^* over various experiments. Let's postulate $w_i = \mathbf{b} + \nu_i$ with ν_i a white noise of variance σ^2, and $\mathbf{b}$ and *unknown* calibration bias that is identical for all points but varies from one experiment to the next. Following least squares covariance, see [169,186], we end up with:

$$\mathbf{G}\mathbf{Q}_{\text{sensor}}\mathbf{G}^T = \sigma^2 \mathbf{A}^{-1} + \mathbf{A}^{-1}\mathbf{B}\,\text{cov}(\mathbf{b})\,\mathbf{B}^T\mathbf{A}^{-1}, \tag{9.14}$$

where $\mathbf{A} = \sum_{k=1}^{K} \mathbf{B}_k^T \mathbf{B}_k$, and $\mathbf{B} = \sum_{k=1}^{K} \mathbf{B}_k^T$. We recover the covariance $\sigma^2\mathbf{A}^{-1}$ of [169, 186] w.r.t. sensor white noise, and a new term, $\mathbf{A}^{-1}\mathbf{B}\,\text{cov}(\mathbf{b})\,\mathbf{B}^T\mathbf{A}^{-1}$, that represents the covariance w.r.t. the unknown bias $\mathbf{b}$, that is, correlated noise. This new additional term is paramount as $\mathbf{A}$ has magnitude proportional to K, hence $\mathbf{A}^{-1}$ is very small, explaining that Censi's formula (based on $\mathbf{A}^{-1}$ only) seems overoptimistic [188]. For example $\mathbf{A}^{-1}$ has trace 0.2 cm for the registration displayed in Figure 9.1 whereas the covariance w.r.t. the unknown bias has trace 2.6 cm. In practice $\mathbf{b}$ arises from sensor calibration, laser stability [182], observed material [174], and incidence of beams [183]. In the remainder we assume bias standard deviation to be approximately 5 cm as in [174].

Note that (9.14) captures the effect of underconstrained situations like hallways. Indeed in unobservable directions the cost $J_{\hat{\mathbf{T}}_{\text{rel}}}(\cdot)$ is constant, yielding small eigenvalues for $\mathbf{A}$ and hence large covariance (9.14). Derivation of $\mathbf{B}_k$ and extraction of $\mathbf{G}$ in (9.14) are available with chapter code.

3.2 3.2 Computation of Dispersion owing to ICP Initialization

Computation of $(\mathbf{I}_6 - \mathbf{J})\mathbf{Q}_{\text{ini}}(\mathbf{I}_6 - \mathbf{J})^T$ in (9.10) is of greater importance as in practice it largely dominates $\mathbf{G}\mathbf{Q}_{\text{sensor}}\mathbf{G}^T$. We propose to compute it in a deterministic derivative-free method, in which we adapt the unscented transform [189] (see e.g. Algorithm 1) pose by following [60,90]. The advantages of using our unscented based method rather than Monte-Carlo sampling are fourfold: 1) it is deterministic; 2) it remains computationally reasonable by adding only 12 ICP registrations; 3) it explicitly computes the cross-covariance matrix between $\hat{\mathbf{T}}_{\text{icp}}$ and $\mathbf{T}_{\text{ini}}$ as a by-product without extra computational operations; and 4) it scales with $\mathbf{Q}_{\text{ini}}$, i.e. our approach naturally self-adapts to the confidence we have in initialization without extra parameter tuning.

We compute the covariance as follows, see Algorithm 10:

- we consider the prior distribution $\mathbf{T}_{\text{prior}} \sim \mathcal{N}_L(\mathbf{T}_{\text{ini}}, \mathbf{Q}_{\text{ini}})$, which is approximated by a set of so-called sigma-points ξ_{ini}^j, see step 1);

Algorithm 10: Computation of matrix $\mathbf{J}$ in (9.10)

Input: $\mathcal{P}, \mathcal{Q}, \mathbf{T}_{\text{ini}}, \mathbf{Q}_{\text{ini}}, \hat{\mathbf{T}}_{\text{icp}} = \mathbf{T}_{\text{ini}}\text{icp}\left(\mathcal{P}, \mathbf{T}_{\text{ini}}^{-1}\mathcal{Q}\right)$;

// set sigma points

1 $\xi_{\text{ini}}^{j} = \text{col}\left(\sqrt{6\mathbf{Q}_{\text{ini}}}\right)_{j},\ j = 1,\ldots,6,$

$\xi_{\text{ini}}^{j} = -\text{col}\left(\sqrt{6\mathbf{Q}_{\text{ini}}}\right)_{j-6},\ j = 7,\ldots,12;$

// propagate sigma points through (7)

2 $\mathbf{T}_{\text{ini}}^{j} = \mathbf{T}_{\text{ini}}\exp\left(\xi_{\text{ini}}^{j}\right),\ j = 1,\ldots,12;$

$\hat{\mathbf{T}}_{\text{icp}}^{j} = \mathbf{T}_{\text{ini}}^{j}\text{icp}\left(\mathcal{P}, (\mathbf{T}_{\text{ini}}^{j})^{-1}\mathcal{Q}\right),\ j = 1,\ldots,12;$

3 $\hat{\xi}_{\text{icp}}^{j} = \exp^{-1}\left(\hat{\mathbf{T}}_{\text{icp}}^{-1}\hat{\mathbf{T}}_{\text{icp}}^{j}\right),\ j = 1,\ldots,12;$

// compute covariance and infer $\mathbf{J}$

4 $(\mathbf{I}_6 - \mathbf{J})\mathbf{Q}_{\text{ini}}(\mathbf{I}_6 - \mathbf{J})^T = \sum_{j=1}^{12}\frac{1}{12}\hat{\xi}_{\text{icp}}^{j}\hat{\xi}_{\text{icp}}^{jT};$

5 $\hat{\xi}_{\text{icp}} = \sum_{j=1}^{12}\frac{1}{12}\hat{\xi}_{\text{icp}}^{j};$

6 $\mathbf{J} = -\left(\sum_{j=1}^{12}\frac{1}{12}\left(\hat{\xi}_{\text{icp}}^{j} - \hat{\xi}_{\text{icp}}\right)\xi_{\text{ini}}^{jT}\right)\mathbf{Q}_{\text{ini}}^{-1} + \mathbf{I}_6;$

Output: $\mathbf{J}$, $(\mathbf{I}_6 - \mathbf{J})\mathbf{Q}_{\text{ini}}(\mathbf{I}_6 - \mathbf{J})^T$;

metric	NNE		KL div.		NNE*		KL div.*	
	trans.	rot.	trans.	rot.	trans.	rot.	trans.	rot.
$\hat{\mathbf{Q}}_{\text{censi}}$	10^3	10^3	10^4	10^5	38	10^2	10^3	10^5
$\hat{\mathbf{Q}}_{\text{carlo}}^{\text{monte}}$	10^3	10^2	10^4	10^4	22	20	10^3	10^3
proposed	**4.2**	**34**	$\mathbf{10^2}$	$\mathbf{10^2}$	**0.8**	**3.8**	**31**	**98**

Table 9.1: Results of ICP uncertainty estimation in term of Normalized Norm Error (NNE) and Kullbach-Leibler divergence (KL div.) divided into translation and rotation parts. As the ICP error distributions are not Gaussian [177], we provide robust statistics (starred, *) by removing both the more and less accurate quantiles of each registration. The proposed method outperforms the two others.

- we approximate the propagated distribution $\mathbf{T}_{\text{prop}} = \mathbf{T}_{\text{prior}}\text{icp}(\mathcal{P}, \mathbf{T}_{\text{prior}}^{-1}\mathcal{Q})$ as

$$\begin{aligned}\mathbf{T}_{\text{prop}} &= \mathcal{N}_L(\mathbf{T}_{\text{ini}}, \mathbf{Q}_{\text{ini}})\,\text{icp}\left(\mathcal{P}, \mathcal{N}_L(\mathbf{T}_{\text{ini}}, \mathbf{Q}_{\text{ini}})^{-1}\mathcal{Q}\right)\\ &\sim \mathcal{N}_L\left(\hat{\mathbf{T}}_{\text{icp}}, (\mathbf{I}_6 - \mathbf{J})\mathbf{Q}_{\text{ini}}(\mathbf{I}_6 - \mathbf{J})^T\right),\end{aligned} \tag{9.15}$$

after propagating each sigma-point in steps 2) and 3), where $\hat{\mathbf{T}}_{\text{icp}}$ is the given ICP pose estimate. We compute the covariance and infer the matrix $\mathbf{J}$ as a by-product in respectively steps 4) and 6).

We derive the algorithm by following [60] for pose measurement, zero-mean prior distribution, and where we set $\alpha = 1$.

4 4 Experimental Results

4.1 4.1 Dataset Description & ICP Algorithm Setting

This section evaluates the ability of the approach to estimate ICP uncertainty on the *Challenging data sets for point cloud registration algorithms* [176]. It comprises eight sequences where point clouds are taken in environments ranging from structured to unstructured, and indoor to outdoor. Each sequence contains between 31 and 45 point cloud scans along with ground-truth pose for each scan, that provides a total of 268 scans and 1020 different registrations as we align each scan with the three scans the following.

We configure the ICP as in [177] with 95% random sub-sampling, kd-tree for data association, and point-to-plane error metric where we keep the 70% closest point associations for rejecting outliers.

4.2 4.2 Compared Methods and Evaluation Metrics

This section evaluates the following methods:

$\hat{\mathbf{Q}}_{\text{censi}}$: the close-form method of [169] adapted for the ICP setting defined above;

$\hat{\mathbf{Q}}^{\text{monte}}_{\text{carlo}}$: the covariance computed after sampling of 65 Monte-Carlo ICP estimates;

$\hat{\mathbf{Q}}_{\text{icp}}$: our proposed approach detailed in Section 3.

Each method assumes depth sensor white noise and bias with 5 cm standard deviation, which is the mean value found in [174] for the Hokuyo sensor used for these experiments, and all methods know the initial uncertainty $\mathbf{Q}_{\text{ini}}$, whose magnitude 0.1 m and 10 deg corresponds to the easy scenario of [177].

We compare the above methods using two metrics:

Normalized Norm Error (NNE): that evaluates the historically challenging [169,184] prediction of the covariance scale, and is computed as

$$\text{NNE} = \Big(\frac{1}{N}\sum_{n=1}^{N} \|\xi_n\|_2^2 / \text{trace}(\hat{\mathbf{Q}}_n)\Big)^{1/2}, \tag{9.16}$$

where $\xi_n = \exp^{-1}(\mathbf{T}_{\text{true}}^{-1}\hat{\mathbf{T}}_n)$ with is the transformation error and $\hat{\mathbf{Q}}_n$ the estimated uncertainty covariance matrix, and averaged over N samples. This metric characterizes the uncertainty as only the true registration is known (the exact distribution of the point cloud is unknown). The target value is one, below one the estimation is pessimistic, whereas a value over one indicates an overoptimistic estimation.

Kullback-Leibler Divergence (KL div.): which is computed between a pseudo-true distribution and the estimated distribution. The pseudo-true distribution is computed after sampling 1000 ICP estimates of the evaluated registration over the initial position. As sensor noise is fixed in the point clouds, this distribution represents the uncertainty stemming from initialization errors.

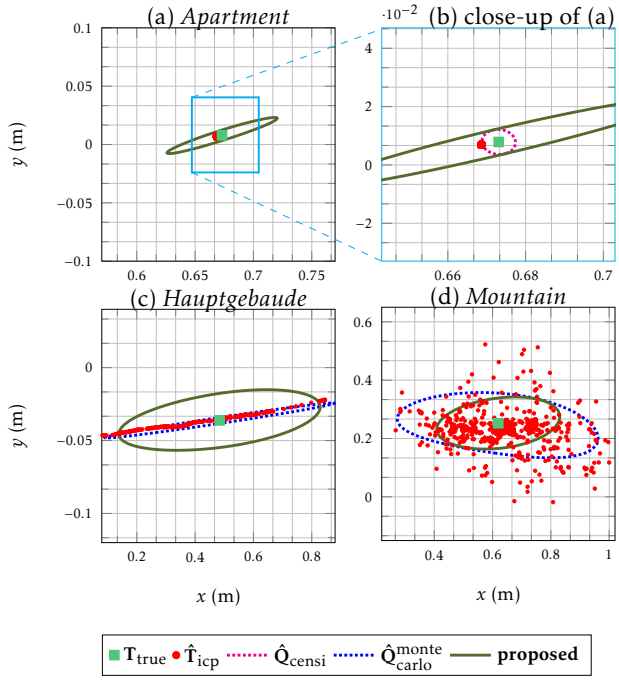


Figure 9.3: Results on real data of [176] projected onto the ground plane for visualization. Ellipses represent the 95% (3σ) confidence sets for each uncertainty estimation method. (a): "true convergence situation", the errors are mainly caused by sensor noises and Censi's formula should apply. (b): however we see the Censi ellipse seems optimistic as ground truth is almost outside it whereas it falls well within our ellipse (green). (c, d): "wrong convergence", the large errors are due ICP initialization. Only our approach is consistent with ICP uncertainty in each environment and does not suffer from overoptimism.

sequence	*Apartment*		*Hauptgebaude*		*Stairs*		*Mountain*		*Gazebo summer*	
Mah. dist.	trans.	rot.	trans.	rot.	trans.	rot.	trans.	rot.	trans.	rot. .
CELLO-3D	0.2	0.1	0.3	0.2	0.1	0.2	-	-	0.2	0.2
ini.+ICP	3.5	15	1.9	3.2	**1.1**	**4.2**	1.5	**1.2**	1.1	**2.1**
proposed	**2.3**	**9.8**	**1.8**	**2.9**	**1.1**	**4.2**	**1.2**	**1.2**	**1.0**	2.3

sequence	*Gazebo winter*		*Wood summer*		*Wood winter*	
Mah. dist.	trans.	rot.	trans.	rot.	trans.	rot.
CELLO-3D	0.1	0.2	0.1	**0.3**	0.1	**0.3**
ini.+ICP	1.9	**3.7**	**1.5**	4.6	**1.2**	4.8
proposed	**1.8**	**3.7**	**1.5**	4.7	**1.2**	4.2

Table 9.2: Trajectory consistency results in term of Mahalanobis distance (bold indicates best performance) split into translation and rotation parts for the sequences of [176], where *Mountain* is not considered in [175]. Our method obtains on average the best uncertainty assessment, albeit slightly optimistic.

4.3 4.3 Results

Results are averaged over 1000 initializations for each of the 1020 considered pairs of point clouds, representing a total of more than one million registrations, where the ICP is initialized with a different estimate $\mathbf{T}_{\text{ini}}$ sampled from $\mathcal{N}_L(\mathbf{T}_{\text{true}}, \mathbf{Q}_{\text{ini}})$. Table 9.1 provides average results over the eight sequences, and Figure 9.3 illustrates typical registrations from structured to unstructured environments. We observe:

- $\hat{\mathbf{Q}}_{\text{censi}}$ is far too optimistic and unreliable for sensor-fusion, as noted in [188]. Its centimetric confidence interval makes sense only when ICP is very accurate;
- $\hat{\mathbf{Q}}^{\text{monte}}_{\text{carlo}}$ is overoptimistic when the discrepancy arising from ICP initialization remains negligible, see Figure 9.3 (b), for which the method predicts a confidence interval with millimetric size. This is naturally explained as the method assumes no error caused by sensor noises;
- the proposed method obtains the best results for both metrics as displayed in Table 9.1. It notably outperforms $\hat{\mathbf{Q}}^{\text{monte}}_{\text{carlo}}$ while deterministic hence more reliable, and computationally much cheaper. The dominant term is generally due to initial uncertainty. However in “global minimum" cases the sensor bias used for computing $\hat{\mathbf{Q}}_{\text{icp}}$ slightly inflates the covariance of [169], and more closely captures actual uncertainty, see Figure 9.3 (a,b).

Besides outperforming the other methods, our method provides simple parameter tuning: we set the bias noise standard deviation as having same magnitude as sensor white noise, and the error stemming from ICP initialization does not need to be tuned when $\mathbf{Q}_{\text{ini}}$ is an output of inertial, visual, or wheeled odometry system [172,173].

Regarding computational complexity and execution time, step 2) of the algorithm requires 12 registrations which take 6 s when registrations are computed parallely, whereas the remaining part of the algorithm takes less than 0.1 s. The 65 Monte-Carlo runs are more than five times more demanding than the the proposed method.

5 5 Complementary Experimental Results

This section provides an in-depth analysis of the method in term of trajectory consistency, robustness to high and misknown initial uncertainty, and discusses the advantages and the validity of the approach.

5.1 5.1 Application to Trajectory Consistency

We asses the quality of the covariance estimation in Section 3 over trajectories as follows. For each sequence of [176], we compute the global pose estimate at scan l by compounding transformations such that $\hat{\mathbf{T}}^l = \hat{\mathbf{T}}^{0,1} \dots \hat{\mathbf{T}}^{l-1,l}$, whose covariance is computed with the closed-form expressions of [90] which are valid up to 4-th order approximation. We compare three methods defined as:

CELLO-3D : reproduced results of [175], that proposes a learning based method for estimating the ICP covariance, which is trained on environments similar to the tested sequence. The results are indicative as the ICP setting of [175] slightly differs from the setting of [177] we use;

ini.+ICP : combines initialization and ICP measurements with the covariance estimate (9.12) without considering cross-covariance terms, i.e., applying formulas of [90] ;

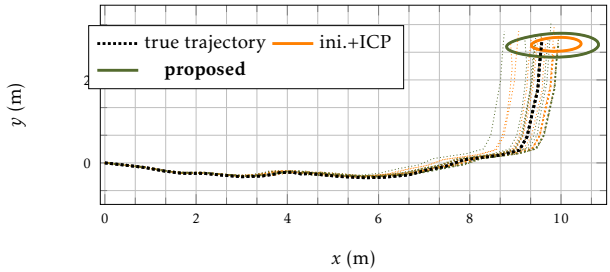


Figure 9.4: Results projected onto the ground plane for visualization in the *Stairs* sequence of [176], where the "ellipses" (lines) represent the 95% (3σ) final confidence sets.

proposed : based on the full proposed covariance of the maximum-likelihood estimate (9.12).

We set initial errors as in Section 4 and evaluate the above methods using the *Mahalanobis Distance* proposed in [175] between final trajectory estimates and ground truth

$$\text{Mah. dist.} = \left(\sum_{n=0}^{N} \frac{\xi_n^T \hat{\mathbf{Q}}_n^{-1} \xi_n}{\dim(\xi_n) N} \right)^{1/2}, \tag{9.17}$$

where $\xi_n = \exp^{-1}(\mathbf{T}_{\text{true}}^{-1} \hat{\mathbf{T}}_n)$ is the transformation error and $\hat{\mathbf{Q}}_n$ the estimated covariance matrix, averaged over N samples. The target value is one, below one the estimation is pessimistic, and above one the estimates are optimistic.

We average results over 40 different initial trajectories for each sequence, which are numerically displayed in Table 9.2 and illustrated in Figure 9.4. We observe:

- CELLO-3D is the only pessimistic method, which estimates uncertainty ranging from 3 to 10 times higher than actual uncertainty. It evidences how difficult it is to asses ICP uncertainty in practice;
- the proposed approach obtains on average the best results. It obtains similar estimates than ini.+ICP when the ICP algorithm is accurate (*Gazebo* and *Wood*). In more difficult environments, e.g. *Stairs*, it better incorporates initialization than ini.+ICP thanks to it accounting for measurement correlation encoded in (9.11), see Figure 9.4.

These results confirm the ability of the method to compute covariance estimates over trajectories also and the relevance of correlation terms between ICP and initial estimates.

5.2 5.2 Role of Initial Uncertainties in Covariance Estimation

We evaluate the influence of $\mathbf{Q}_{\text{ini}}$ on the covariance estimation in challenging situations where $\mathbf{Q}_{\text{ini}}$ is high, inaccurately known (the estimation of $\mathbf{Q}_{\text{ini}}$ is in itself challenging), and sensor noise is inflated. For each environment of [176], we evaluate the method in 9 situations where initial uncertainty is easy, medium and difficult with respectively 0.1, 0.5, 1 m and 10, 20, 50 deg standard deviation, see [177]. In each situation we evaluate the algorithm with different magnitudes for $\mathbf{Q}_{\text{ini}}$, hence assessing its robustness to pessimistic and optimisitic parametrization. We finally add white and depth bias

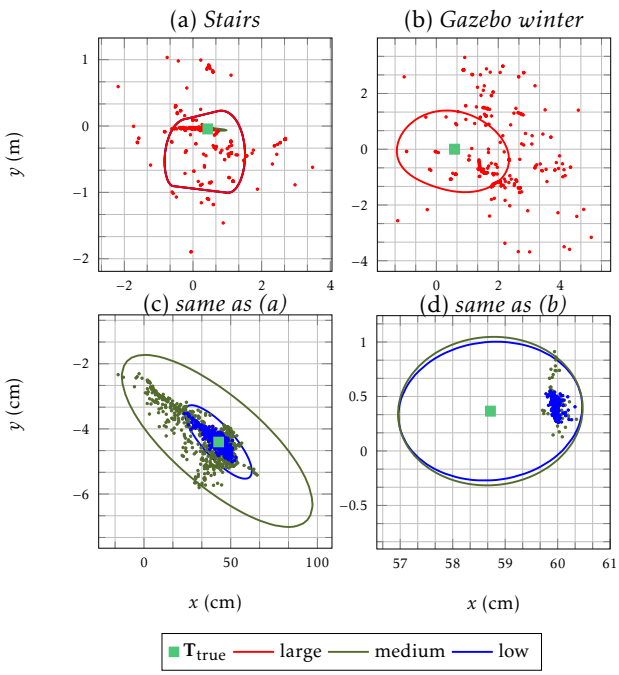


Figure 9.5: ICP results (dots) and 95% (3σ) confidence sets (lines) following our method for three levels of initial uncertainty. We see the latter highly influences the ICP registrations and ICP covariance estimations accordingly.

noises on already noisy point clouds with 5 cm standard deviation. Results are given in Table 9.3, and illustrated in Figure 9.5.

- The ICP algorithm obtains unreliable results for large initial uncertainty, see Figure 9.5, whereas it obtains centimetric errors for low levels of initial uncertainty. As anticipated in Section 2.2, significant ICP errors are caused by inaccurate initialization;

- ICP final outputs are agnostic to initialization when a global minimum exists, see e.g. the *Gazebo* and *Wood* environments in Table 9.3 for levels of $\mathbf{Q}_{\text{ini}}$ corresponding to easy and medium scenario. The method obtains correct estimates where the sensor noise terms numerically dominate the estimated covariances, and $\mathbf{J} \approx \mathbf{I}$, see Section 2.4;

- another environments contain local minima, e.g. *Hauptgebaude*. Then the algorithm outputs reflect the pessimistic or optimistic belief about the initial uncertainty. We recommend in these situations to set $\mathbf{Q}_{\text{ini}}$ sufficiently high to favour conservatism;

- our method is able to detect inaccurate ICP registrations by providing very high covariance estimates, although it cannot accurately describe non Gaussian distributions, see Figure 9.5 (a,b).

$\mathbf{Q}_{\text{ini}}$(true)	easy			medium			difficult		
$\hat{\mathbf{Q}}_{\text{ini}}$(algo)	easy	med.	diff.	easy	med.	diff.	easy	med.	diff.
Appart.	1.6	1.5	10^{-6}	1.8	1.7	10^{-6}	10^{5}	10^{5}	0.6
Haupt.	1.8	0.3	10^{-3}	3.6	0.7	10^{-3}	10^{3}	10^{3}	0.3
Stairs	0.7	0.2	10^{-3}	1.9	0.8	10^{-3}	10^{3}	10^{3}	0.7
Montain	2.1	1.1	10^{-3}	3.3	1.8	10^{-3}	10^{3}	10^{3}	4.0
Gazebo	1.1	1.0	10^{-5}	1.3	1.2	10^{-5}	10^{4}	10^{4}	0.8
Wood	2.1	2.1	10^{-3}	2.2	2.2	10^{-3}	10^{3}	10^{3}	0.3

Table 9.3: NNE, see (9.16), for different levels of true and supposed initial uncertainty. Difficult initial uncertainty leads to highly erroneous ICP outputs that the proposed method detects if correctly parametrized.

5.3 5.3 Discussion about the Proposed Approach

We finally examine the pros, cons, and fundamental assumptions of the method. The main advantages are: it being anchored in a mathematical theory, its efficiency to assess uncertainty for acceptable levels of initial uncertainties, its simplicity, while being computationally reasonable, see Algorithm 1. The cross-covariance term in (9.12) may be fruitful for increasing robustness of back-end systems, e.g. pose-graph [188], as it correlates two previously supposed independent measurements. Comparisons between diagonal terms in (9.12) finally provides a way for trading-off between initial odometry guesses and ICP estimates, see [193].

The Gaussian error assumption of the ICP estimates is the core hypothesis of the method. We required this assumption to obtain a tractable method able to provide a covariance for a state estimator, e.g. a Kalman filter. However, if one pursues a more accurate estimation of the ICP distribution, we suggest massive sampling methods as an expansive alternative, although our method largely proves sufficient to detect problematic situations.

The method finally requires the covariance of the initial uncertainty as an input. If the provided initialization confidence is inexact, the method outputs may reflect the initial optimism or pessimism in general situations. Nonetheless an insight of the present chapter, see Section 2 and e.g. Figure 9.5, is that ICP errors intrinsically depend on the initial accuracy so that a coarse idea of initial uncertainty is essential, whatever the method one desires to use.

Finally, [18,90,190] show that concentrated Gaussian distribution (9.1) faithfully describe robot odometry models, such that methods like the Kalman filter are able to provide an accurate (concentrated) Gaussian approximation to the true initial uncertainty for which our approach has been designed.

6 6 Conclusion

This chapter presents a novel method for real time estimation of 3D uncertainty covariance matrix of the ICP algorithm. The method relies on a careful study of the influence of both sensor noises and algorithm initialization on the ICP estimates, that we leverage in a deterministic scheme which remains very simple in terms of parameter tuning. The core of our approach is versatile as one can apply it to various choices of error metrics. However with point-to-point ICP the closed form part of the covariance is not valid, see [186]. The approach is successfully validated on individual pairs

of point clouds and over trajectories on challenging real datasets, where it obtains consistent results.

Part III

Using Deep Learning to Extract Information from an IMU

CHAPTER 10

Introduction to Part III

Deep neural networks allows building data-driven models that learn features for realizing a given task, e.g. image classification. Especially since 2014 and its success in computer vision, deep learning [161] spread among all the subjects of computer sciences, robotics, natural language programming, and others. This fast development comes from 3 enablers:

1. *New theories* for training and applying deep learning in a wide range of topics;
2. *Powerful hardware* like graphic card with mass parallelization;
3. *Lots of data*, with the development of big data and publicly available datasets.

This part of the thesis leverages deep learning for inertial navigation and enhancing of more "traditional" state-estimation methods such as the Kalman filter.

1 1 Relations between Neural Networks and the Kalman Filter

Both neural networks and Kalman filters may estimate parameters from a stream of data/measurements, see Figure 10.1, and have non-linear transformation that are local linearization of the measurements and propagation error models in the EKF, while neural networks use linear operations between consecutive layers.

In a Kalman filter, the researcher applies a specific domain knowledge, e.g. physics equations, and noise models to estimate the parameters in a way which is optimal as it minimizes the mean squared error under certain assumptions (zero-mean Gaussian noise). Indeed, the Kalman filter computes the posterior of the state given observations through the prediction step and update step. So the propagation $f(\cdot)$ and the observation $h(\cdot)$ functions should be explicit and tractable, which requires great efforts on designing an approximation.

In a neural network the model is not explicitly written, but learned from the measurements, by looking at many examples and by using an appropriate optimization scheme. Indeed, vanilla neural networks ignore whatever happens in the middle and directly compute the mapping between input and output, with a massive parameterized differentiable function. The *a priori* absence of expert knowledge makes neural networks appears as a dream tool for complex cross-domain tasks.

However, in practice, applying neural networks requires some kind of expertise, see e.g. the success of [194] for point cloud classification and segmentation. Regarding

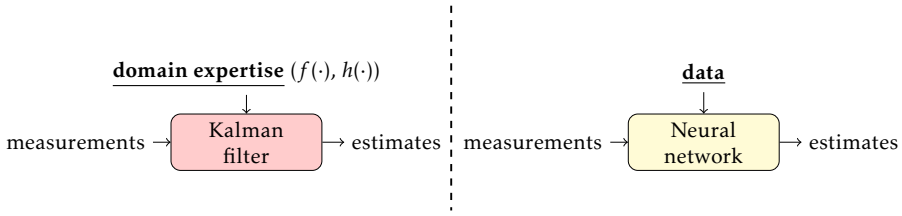


Figure 10.1: Overview of Kalman filter (left) and neural networks for state estimation (right). Both approaches share same inputs and outputs but differ in their design. A Kalman filter is generally built with domain expertise while neural networks are trained with data.

state estimation, many works combine neural networks and Kalman filters which can be classified in two manners. The first is to use neural networks to learn the model of the Kalman filters, where the neural network optimizes *directly* the filter estimates, see [150,153,195] and Chapter 8. The other method is to train a neural network for an auxillary and *decoupled* task. The network outputs are feed, once training is over, to the Kalman filter.

2 2 Tightly Coupled Approach: Designed "Deep" Kalman Filter with Neural Networks

Recent works such as [150,153] propose two methods to learn Kalman filters using neural networks, where the authors search to learn respectively a generative temporal model $f(\cdot)$ and an observation model $h(\cdot)$ from high-dimensional data, using deep neural networks as a building block, see also [196] that performs preliminary researches to design model for UKFs. One main advantage is that the Kalman filter pipeline is yet given, and only the models (or parts of these model such as noise covariance matrices) have to be learn. Thus the neural network is trained for *directly* improving state estimates.

Albeit the authors get promising results and that training is that way seem optimal, see also [152] for pedestrian navigation, this type of methods is still limited by accuracy, generalization, robustness, data, and computational requirement. This holds for Chapter 8, where adding a new sensor would require to completely relearn the neural networks.

3 3 Loosely Coupled Approach: Adding Information to Kalman Filter with Neural Networks

Let us consider the safety features of an autonomous car which is an example to observe how neural networks and Kalman filters are related in practical scenario.

For the autonomous pilot feature to function seamlessly, the vehicle needs to know the type of objects on the road (car, truck or motor cycle) and also it needs to know the distance to the object, its velocity and heading, in order to predict the trajectory of the object and make decisions on how the vehicle should function.

Consider now a simple case of how does an autonomous vehicle detect a car in the surroundings and estimate its position, velocity and heading. The front camera captures the images and sends it to an embedded platform that runs the software to detect

what kind of an object are present in the image. This image classifier is a trained neural network that has been trained with thousands of images to classify various objects. i.e. the input to the neural network is an image and output is a classified object such as a car or a truck or a pedestrian. Furthermore, from a series of images it is possible to estimate the distance of the car and its velocity as well, but the estimated distance and velocity are not accurate. In order to get a better accuracy of the estimations, the information from the front looking radar is used, that can estimate the distance to the car, its velocity and its heading as well.

Now here comes the Kalman filter, where the filter fuses the information (such as the type of the object, its distance, velocity and heading) from both the camera and the radar in a prediction and update step, to get an accurate estimation of the state of the object, which will then be fed into the decision making unit of the autonomous pilot to make maneuvers accordingly.

In these paradigm, neural networks are trained for another purpose, e.g. modelling $f(\cdot)$ or $h(\cdot)$, but independently from the filter, and then used in the filter. We illustrate this point here with two examples:

- Learning relative displacement from IMU and use them in a MSCKF like filter based on a IMU only [197];
- Learning depth from convolutional neural networks that provides initial depth estimates of a semi-direct visual odometry system [198].

Both examples successfully train the neural networks, and require manual tweaking to adapt the network output to the filter, e.g. by inflating network output uncertainty. We choose to perform research in this direction, that is, leveraging our expert knowledge in inertial navigation problem to design deep neural network for specific tasks thar are measurement detection in Chapter 11, and measurement denoising in Chapter 12. Chapter 13 presents a few additional results as the combination of the methods of Chapter 8 and Chapter 11. Finally, we provide in Appendix few tips to practitioners regarding Kalman filtering enhanced by deep learning.

CHAPTER 11

RINS-W: Robust Inertial Navigation System on Wheels

The present chapter has been published in the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).

Résumé 🇫🇷

Ce chapitre propose une approche en temps réel pour la navigation inertielle à long terme basée *uniquement* sur une centrale inertielle pour les robots à roues se localisant de manière autonome. L'approche repose sur deux composantes: 1) un détecteur robuste qui utilise des réseaux de neurones profonds récurrents pour détecter dynamiquement une variété de situations d'intérêt, telles qu'une vitesse nulle ou aucun glissement latéral; et 2) un filtre de Kalman qui incorpore ces connaissances en tant que pseudo-mesures pour la localisation. Les évaluations faites sur un jeu de données provenant d'une voiture et publiquement accessible démontrent que le schéma proposé peut atteindre une précision finale de 20 m pour une trajectoire de 21 km de long d'un véhicule roulant pendant plus d'une heure, équipé d'une centrale inertielle de précision modérée (le taux de dérive du gyroscope est de 10 degrés par heure). Ce chapitre combine des techniques d'apprentissage approfondies sophistiquées avec des méthodes de filtrage de pointe pour la navigation inertielle pure sur véhicules à roues et, à ce titre, ouvre la voie à de nouvelles techniques de navigation inertielle basées sur les approaches d'apprentissage.

Chapter abstract

This chapter proposes a real-time approach for odometry inertial navigation based *only* on an IMU for self-localizing wheeled robots. The approach builds upon two components: 1) a robust detector that uses recurrent deep neural networks to dynamically detect a variety of situations of interest, such as zero velocity or no lateral slip; and 2) a state-of-the-art Kalman filter which incorporates this knowledge as pseudo-measurements for localization. Evaluations on a publicly available car dataset demonstrates that the proposed scheme may achieve a final precision of 20 m for a 21 km long trajectory of a vehicle driving for over an hour, equipped with an IMU of moderate precision (the gyro drift rate is 10 deg/h). This chapter combines sophisticated deep learning techniques with state-of-the-art filtering methods for pure inertial navigation on wheeled vehicles and as such opens up for novel data-driven inertial navigation techniques.

1 1 Introduction

In this chapter, we provide a method for odometry localization for wheeled robots only using an IMU. Our contributions, and the chapter's organization, are as follows:

- we introduce specific motion profiles frequently encountered by a wheeled vehicle which bring information about the motion when correctly identified, along with their mathematical description in Section 3;
- we design an algorithm which automatically detects in real time if any of those assumptions about the motion holds in Section 4.1, based only on the IMU measurements. The detector builds upon recurrent deep neural networks [161] and is trained using IMU and ground truth data;
- we implement a state-of-the-art invariant extended Kalman filter [42,43] that exploits the detector's outputs as pseudo-measurements to combine them with the IMU outputs in a statistical way, in Section 4.2. It yields accurate estimates of pose, velocity and sensor biases, along with associated uncertainty (covariance);
- we demonstrate the performances of the approach on a publicly available car dataset [199] in Section 5. Our approach solely based on the IMU produces accurate estimates with a final distance w.r.t. ground truth of 20 m on the 73 minutes test sequence urban16, see Figure 11.1. In particular, our approach outperforms differential wheel odometry, as well as wheel odometry aided by an expensive fiber optics gyro whose drift is 200 times smaller than the one of the IMU we use;
- this evidences that accurately detecting some specific patterns in the IMU data and incorporating this information in a filter may prove very fruitful for localization;
- the method is not restricted to IMU only based navigation. It is possible to feed the Kalman filter with other sensors' measurements. Besides, once trained the detector may be used as a building block in any localization algorithm.

2 2 Inertial Navigation System & Sensor Model

Denoting the IMU orientation by $\mathbf{R}_n \in SO(3)$, i.e. the rotation matrix that maps the body frame to the world frame w, its velocity in w by $\mathbf{v}_n^{\text{IMU}} \in \mathbb{R}^3$ and its position in w by $\mathbf{p}_n^{\text{IMU}} \in \mathbb{R}^3$, the dynamics are as follows

$$\mathbf{R}_{n+1}^{\text{IMU}} = \mathbf{R}_n^{\text{IMU}} \exp_{SO(3)}(\boldsymbol{\omega}_n dt), \tag{11.1}$$

$$\mathbf{v}_{n+1}^{\text{IMU}} = \mathbf{v}_n^{\text{IMU}} + (\mathbf{R}_n \mathbf{a}_n + \mathbf{g})\, dt, \tag{11.2}$$

$$\mathbf{p}_{n+1}^{\text{IMU}} = \mathbf{p}_n^{\text{IMU}} + \mathbf{v}_n^{\text{IMU}} dt, \tag{11.3}$$

between two discrete times, with sampling time dt, and where $\left(\mathbf{R}_0^{\text{IMU}}, \mathbf{v}_0^{\text{IMU}}, \mathbf{p}_0^{\text{IMU}}\right)$ is the initial configuration. The true angular velocity $\boldsymbol{\omega}_n \in \mathbb{R}^3$ and the true specific acceleration $\mathbf{a}_n \in \mathbb{R}^3$ are the inputs of the system (11.1)-(11.3). In our application scenarios, the effects of earth rotation and Coriolis acceleration are ignored, and earth is considered flat.

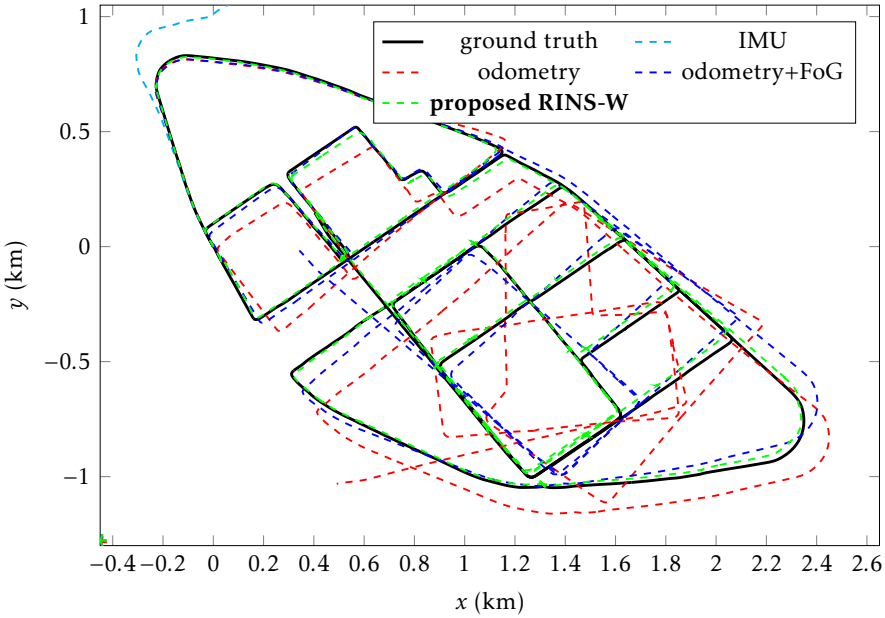


Figure 11.1: Trajectory ground truth and estimates obtained by various methods: integration of IMU signals; odometry based on a differential wheel encoder system; odometry combined with an highly accurate and expensive Fiber optics Gyro (FoG) that provides orientation estimates; and the proposed RINS-W approach which considers only the IMU sensor embarked in the vehicle, and which outperforms the other schemes. The final distance error for this long-term sequence urban16 (73 minutes) of the car dataset [199] is 20 m for the RINS-W solution. The deep learning based detector (see Section 4.1) has of course *not* been trained or cross-validated on this sequence.

2.1 2.1 Inertial Measurement Unit (IMU) Model

The IMU provides noisy and biased measurements of $\boldsymbol{\omega}_n \in \mathbb{R}^3$ and $\mathbf{a}_n \in \mathbb{R}^3$ as follows

$$\boldsymbol{\omega}_n^{\text{IMU}} = \boldsymbol{\omega}_n + \mathbf{b}_n^{\omega} + \mathbf{w}_n^{\omega}, \tag{11.4}$$

$$\mathbf{a}_n^{\text{IMU}} = \mathbf{a}_n + \mathbf{b}_n^{\mathbf{a}} + \mathbf{w}_n^{\mathbf{a}}, \tag{11.5}$$

where $\mathbf{b}_n^{\omega}$, $\mathbf{b}_n^{\mathbf{a}}$ are quasi-constant biases and $\mathbf{w}_n^{\omega}$, $\mathbf{w}_n^{\mathbf{a}}$ are zero-mean Gaussian noises. The biases follow a random-walk

$$\mathbf{b}_{n+1}^{\omega} = \mathbf{b}_n^{\omega} + \mathbf{w}_n^{\mathbf{b}_\omega}, \tag{11.6}$$

$$\mathbf{b}_{n+1}^{\mathbf{a}} = \mathbf{b}_n^{\mathbf{a}} + \mathbf{w}_n^{\mathbf{b}_\mathbf{a}}, \tag{11.7}$$

where $\mathbf{w}_n^{\mathbf{b}_\omega}$, $\mathbf{w}_n^{\mathbf{b}_\mathbf{a}}$ are zero-mean Gaussian noises.

All sources of error - particularly biases - are yet undesirable since a simple implementation of (11.1)-(11.3) leads to a triple integration of raw data, which is much more harmful that the unique integration of differential wheel speeds. Even a small error may thus cause the position estimate to be way off the true position, within seconds.

3 3 Specific Motion Profiles for Wheeled Systems

We describe in this section characteristic motions frequently encountered by a wheeled robot, that provide useful complementary information to the IMU when correctly detected.

3.1 3.1 Considered Motion Profiles

We consider 4 distinct specific motion profiles, whose validity is encoded in the following binary vector:

$$\mathbf{z}_n = (z_n^{\text{VEL}},\ z_n^{\text{ANG}},\ z_n^{\text{LAT}},\ z_n^{\text{UP}}) \in \{0,1\}^4. \tag{11.8}$$

- **Zero velocity**: the velocity of the platform is null. As a consequence so is the linear acceleration, yielding:

$$z_n^{\text{VEL}} = 1 \Rightarrow \begin{cases} \mathbf{v}_n^{\text{IMU}} = \mathbf{0} \\ \mathbf{R}_n^{\text{IMU}}\mathbf{a}_n + \mathbf{g} = \mathbf{0} \end{cases}. \tag{11.9}$$

  Such profiles frequently occur for vehicles moving in urban environments, and are leveraged in the well known Zero velocity UPdaTe (ZUPT), see e.g. [145,200].

- **Zero angular velocity**: the heading is constant:

$$z_n^{\text{ANG}} = 1 \Rightarrow \boldsymbol{\omega}_n = \mathbf{0}. \tag{11.10}$$

- **Zero lateral velocity**[1]: the lateral velocity is considered roughly null (which is not always true)

$$z_n^{\text{LAT}} = 1 \Rightarrow v_n^{\text{LAT}} \simeq 0, \tag{11.11}$$

  where we obtain the lateral velocity v_n^{LAT} after expressing the velocity in the body frame B as

$$\mathbf{v}_n^{\text{B}} = (\mathbf{R}_n^{\text{IMU}})^T \mathbf{v}_n^{\text{IMU}} = \begin{bmatrix} v_n^{\text{FOR}} \\ v_n^{\text{LAT}} \\ v_n^{\text{UP}} \end{bmatrix}. \tag{11.12}$$

[1] Without loss of generality, we assume that the body frame is aligned with the IMU frame.

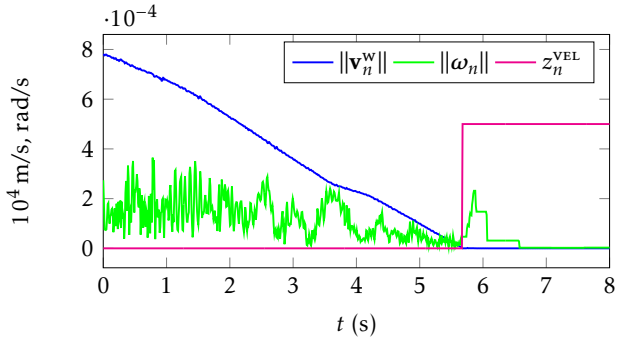


Figure 11.2: Real IMU data of a car stopping from sequence urban06 of [199]. We see (11.9) holds and thus $z_n^{\mathrm{VEL}} = 1$ at $t = 5.8s$ while (11.10) does not hold yet.

- **Zero vertical velocity**: the vertical velocity is considered roughly null

$$z_n^{\mathrm{UP}} = 1 \Rightarrow v_n^{\mathrm{UP}} \simeq 0. \tag{11.13}$$

The two latter are common assumptions for wheeled robots or cars moving forward on human made roads.

3.2 3.2 Discussion on the Choice of Profiles

The motion profiles were carefully chosen in order to match the specificity of wheeled robots equipped with shock absorbers. Indeed, as illustrated on Figure 11.2, when the wheels of a car actually stop, the car undergoes a rotational motion forward and then backward. As a result, null velocity (11.9) does not imply null angular velocity (11.10). For the level of precision we pursue in the present chapter, distinguishing between (11.9) and (11.10) is pivotal since it allows us to: *i*) properly label motion profiles before training (see Section 5.2, where we have different thresholds on position and on angular velocity); and: *ii*) improve detection accuracy since only one motion pattern can be identified as valid.

(11.11) and (11.13) generally hold for robots moving indoors or cars on roads. Note that (11.13) is expressed in the body frame, and thus generally holds for a car moving on a road even if the road is not level. As such (11.13) is more refined that just assuming the car is moving in a 2D horizontal plane. And it is actually quite a challenge for an IMU-based detector to identify when (11.11) and (11.13) are *not* valid.

3.3 3.3 Expected Impacts on Robot Localization

The motion profiles fall into two categories in terms of the information they bring:

1. **Zero velocity constraints**: the profiles (11.9)-(11.10), when correctly detected may allow to correct the IMU biases, the pitch and the roll.

2. **Vehicle motion constraints**: the profiles (11.11) and (11.13) are useful constraints for the estimates accuracy over the long term. In particular, Section 5.5 will experimentally demonstrate the benefits of accounting for (11.11) and (11.13).

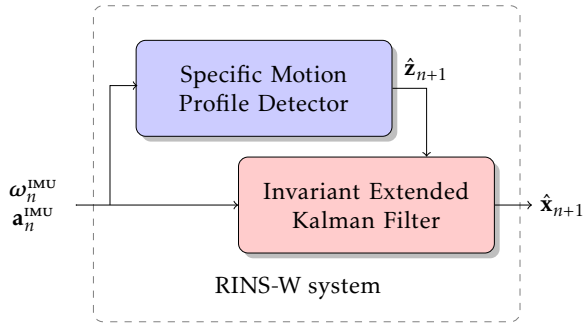


Figure 11.3: Structure of the proposed RINS-W system for inertial navigation. The detector identifies specific motion profiles (11.8) from raw IMU signals only.

4 4 Proposed RINS-W Algorithm

This section describes our system for recovering trajectory and sensor bias estimates from an IMU. Figure 11.3 illustrates the approach which consists of two main blocks summarized as follow:

- the detector estimates the binary vector $\mathbf{z}_n$ from raw IMU signals, and consists of recurrent neural networks;
- the filter integrates the IMU measurements in its dynamic model and exploits the detected motion profiles as pseudo-measurements to refine its estimates, as customary in inertial navigation, see e.g. [145].

The detector does not use the filter's output and is based only on IMU measurements. Thus the detector operates autonomously and is trained independently, see Section 5.2. Note that our approach is different from more tightly coupled approaches such as [153]. We now describe each block.

4.1 4.1 The Specific Motion Profile Detector

The detector determines at each instant n which ones of the specific motion profiles (11.8) are valid, see Figure 11.4. The base core of the detector is a recurrent neural network, namely a Long-Short Term Memory (LSTM) [161]. The LSTMs take as input the IMU measurements and compute:

$$\hat{\mathbf{u}}_{n+1},\ \mathbf{h}_{n+1} = \text{LSTM}\left(\left\{\omega_i^{\text{IMU}},\ \mathbf{a}_i^{\text{IMU}}\right\}_{i=0}^{n}\right) \tag{11.14}$$

$$= \text{LSTM}\left(\omega_n^{\text{IMU}},\ \mathbf{a}_n^{\text{IMU}},\ \mathbf{h}_n\right), \tag{11.15}$$

where $\hat{\mathbf{u}}_{n+1} \in \mathbb{R}^4$ contains probability scores for each motion profiles and $\mathbf{h}_n$ is the hidden state of the neural network. Probability scores are then converted to a binary vector $\hat{\mathbf{z}}_n = \text{Threshold}\left(\hat{\mathbf{u}}_{n+1}\right)$ with a threshold for each motion profile.

The thresholds must be set with care, and the procedure will be described in Section 5.1. Indeed, false alarms lead to degraded performance, since a zero velocity assumption is incompatible with an actual motion. On the other hand, a missed profile is not harmful since it results in standard IMU based dead reckoning using (11.1)-(11.3).

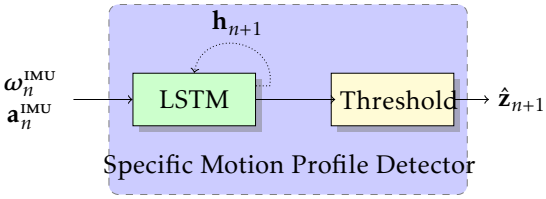


Figure 11.4: Structure of the detector, which consists for each motion pattern of a recurrent neural network (LSTM) followed by a threshold to obtain an output vector $\hat{\mathbf{z}}_n$ that consists of binary components. The hidden state $\mathbf{h}_n$ of the neural network allows the full structure to be recursive.

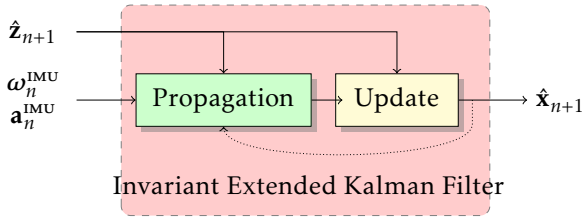


Figure 11.5: Structure of the IEKF. The filter leverages motion profile information $\hat{\mathbf{z}}_n$ both for the propagation and the update of the state $\hat{\chi}_{n+1}$.

4.2 4.2 The Invariant Extended Kalman Filter

We opt for an IEKF, see Chapter 2, to perform the fusion between the IMU measurements and the detected specific motion profiles. The IEKF outputs the state $\hat{\chi}_n$ that consists of pose and velocity of the IMU, the IMU biases, along with their covariance. We now describe the filter more in detail, whose architecture is displayed on Figure 11.5.

IMU state: we define the IMU state as

$$\chi_n = (\mathbf{R}_n^{\text{IMU}},\ \mathbf{v}_n^{\text{IMU}},\ \mathbf{p}_n^{\text{IMU}},\ \mathbf{b}_n^{\omega},\ \mathbf{b}_n^{\mathbf{a}}), \tag{11.16}$$

which contains robot pose, velocity, and the IMU biases. The state evolution is given by the dynamics (11.1)-(11.7), see Section 2. As (11.9)-(11.13) are all measurements expressed in the robot's frame, they lend themselves to the Right IEKF methodology, see [42]. Applying it, we define the linearized state error as

$$\mathbf{e}_n = \begin{bmatrix} \xi_n \\ \mathbf{e}_n^{\mathbf{b}} \end{bmatrix} \sim \mathcal{N}(\mathbf{0}, \mathbf{P}_n), \tag{11.17}$$

where uncertainty is based on the use of the Lie exponential as advocated in [90] in a wheel odometry context, and mapped to the state as

$$\chi_n = \exp_{SE_2(3)}(\xi_n)\,\hat{\chi}_n, \tag{11.18}$$

$$\mathbf{b}_n = \hat{\mathbf{b}}_n + \mathbf{e}_n^{\mathbf{b}}, \tag{11.19}$$

where $\chi_n \in SE_2(3)$ is a matrix that lives is the Lie group $SE_2(3)$ and represents the vehicle state $\mathbf{R}_n$, $\mathbf{v}_n^{\text{IMU}}$, $\mathbf{p}_n^{\text{IMU}}$, $\mathbf{P}_n \in \mathbb{R}^{15\times 15}$ is the error state covariance matrix, $\mathbf{b}_n = (\mathbf{b}_n^{\omega}, \mathbf{b}_n^{\mathbf{a}}) \in \mathbb{R}^6$, $\hat{\mathbf{b}}_n = \left(\hat{\mathbf{b}}_n^{\omega}, \hat{\mathbf{b}}_n^{\mathbf{a}}\right) \in \mathbb{R}^6$, and $\hat{(\cdot)}$ denote estimated variables.

Propagation Step: if no specific motion is detected, i.e. $\hat{z}^{\text{VEL}}_{n+1} = 0$, $\hat{z}^{\text{ANG}}_{n+1} = 0$, we apply (11.1)-(11.7) to propagate the state and obtain $\hat{\chi}_{n+1}$ and associated covariance through the Riccati equation

$$\mathbf{P}_{n+1} = \mathbf{F}_n \mathbf{P}_n \mathbf{F}_n^T + \mathbf{G}_n \mathbf{Q}_n \mathbf{G}_n^T, \tag{11.20}$$

where the Jacobians $\mathbf{F}_n$, $\mathbf{G}_n$ are given in Appendix 6, and where $\mathbf{Q}_n$ denotes the covariance matrix of the noise $\mathbf{w}_n = \left(\mathbf{w}_n^{\omega}, \mathbf{w}_n^{\mathbf{a}}, \mathbf{w}_n^{\mathbf{b}^{\omega}}, \mathbf{w}_n^{\mathbf{b}^{\mathbf{a}}}\right) \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_n)$. By contrast, if a specific motion profile is detected, we modify model (11.1)-(11.7) as follows:

$$\hat{z}^{\text{VEL}}_{n+1} = 1 \Rightarrow \begin{cases} \mathbf{v}^{\text{IMU}}_{n+1} = \mathbf{v}^{\text{IMU}}_{n} \\ \mathbf{p}^{\text{IMU}}_{n+1} = \mathbf{p}^{\text{IMU}}_{n} \end{cases}, \tag{11.21}$$

$$\hat{z}^{\text{ANG}}_{n+1} = 1 \Rightarrow \mathbf{R}^{\text{IMU}}_{n+1} = \mathbf{R}^{\text{IMU}}_{n}, \tag{11.22}$$

and the estimated state $\hat{\chi}_{n+1}$ and covariance $\mathbf{P}_{n+1}$ are modified accordingly.

Update: each motion profile yields one of the following pseudo-measurements:

$$\mathbf{y}^{\text{VEL}}_{n+1} = \begin{bmatrix} (\mathbf{R}^{\text{IMU}}_{n+1})^T \mathbf{v}^{\text{IMU}}_{n+1} \\ \mathbf{b}^{\mathbf{a}}_{n+1} - (\mathbf{R}^{\text{IMU}}_{n+1})^T \mathbf{g} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{a}^{\text{IMU}}_{n} \end{bmatrix}, \tag{11.23}$$

$$\mathbf{y}^{\text{ANG}}_{n+1} = \mathbf{b}^{\omega}_{n+1} = \omega^{\text{IMU}}_{n}, \tag{11.24}$$

$$\mathbf{y}^{\text{LAT}}_{n+1} = v^{\text{LAT}}_{n+1} = 0, \tag{11.25}$$

$$\mathbf{y}^{\text{UP}}_{n+1} = v^{\text{UP}}_{n+1} = 0. \tag{11.26}$$

A vector $\mathbf{y}_{n+1}$ is computed by stacking the pseudo-measurements of the detected motion profiles. Note that, if $\hat{z}^{\text{VEL}}_{n+1} = 1$ we do not consider (11.25)-(11.26) since (11.23) implies (11.25)-(11.26). If no specific motion is detected, the update step is skipped, otherwise we follow the IEKF methodology [42] and compute

$$\mathbf{K} = \mathbf{P}_{n+1} \mathbf{H}^T_{n+1} / \left(\mathbf{H}_{n+1} \mathbf{P}_{n+1} \mathbf{H}^T_{n+1} + \mathbf{N}_{n+1}\right), \tag{11.27}$$

$$\mathbf{e}^{+} = \mathbf{K}(\mathbf{y}_{n+1} - \hat{\mathbf{y}}_{n+1}) = \begin{bmatrix} \xi^{+} \\ \mathbf{e}^{\mathbf{b}+} \end{bmatrix}, \tag{11.28}$$

$$\hat{\chi}^{+}_{n+1} = \exp(\xi^{+})\, \hat{\chi}_{n+1}, \; \mathbf{b}^{+}_{n+1} = \mathbf{b}_{n+1} + \mathbf{e}^{\mathbf{b}+}, \tag{11.29}$$

$$\mathbf{P}^{+}_{n+1} = (\mathbf{I} - \mathbf{K}\mathbf{H}_{n+1})\, \mathbf{P}_{n+1}, \tag{11.30}$$

summarized as Kalman gain (11.27), state innovation (11.28), state update (11.29) and covariance update (11.30), where $\mathbf{H}_{n+1}$ is the measurement Jacobian matrix given in Appendix 6 and $\mathbf{N}_{n+1}$ the noise measurement covariance.

Initialization: launching the system when the platform is moving without estimating biases and orientation can induce drift at the beginning which then is impossible to compensate. As the filter is able to correctly self-initialize the biases, pitch, roll and its covariance matrix when the trajectory is first stationary, we enforce each sequence in Section 5 to start by a minimum of 1 s stop. Admittedly restrictive, the required stop is of extremely short duration, especially as compared to standard calibration techniques [201].

Figure 11.6: The considered dataset [199] contains data logs of a Xsens MTi-300[2] (right) recorded at 100 Hz along with the ground truth pose.

5 5 Results on Car Dataset

The following results are obtained on the *complex urban LiDAR dataset* [199], that consists of data recorded on a consumer car moving in complex urban environments, e.g. metropolitan areas, large building complexes and underground parking lots, see Figure 11.6. Our goal is to show that using an IMU of moderate cost, one manages to obtain surprisingly accurate dead reckoning by using state-of-the-art machine learning techniques to detect relevant assumptions that can be fed into a state-of-the-art Kalman filter. The detector is trained on a series of sequences and tested on another sequences, but all sequences involve the same car and inertial sensors.

5.1 5.1 Implementation Details

We provide in this section the detector and filter setting of the RINS-W system. The detector disposes of four LSTMs, one for each motion profile. Each LSTM consists of 2 layers of 250 hidden units and its hidden state is mapped to a score probability by a 2 layers multi-perceptron network with a ReLU activation function and is followed by a sigmoid function [161] that outputs a scalar value in the range [0, 1]. We implement the detector on PyTorch[3] and set the threshold values to 0.95 for $(z_n^{\text{VEL}}, z_n^{\text{ANG}})$, and 0.5 for $(z_n^{\text{LAT}}, z_n^{\text{UP}})$. The filter operates at the 100 Hz IMU rate ($dt = 10^{-2}$ s) and its noise covariance matrices are parameterized as

$$\mathbf{Q}_n = \text{diag}\left(\sigma_\omega^2 \mathbf{I},\ \sigma_{\mathbf{a}}^2 \mathbf{I},\ \sigma_{\mathbf{b}_\omega}^2 \mathbf{I},\ \sigma_{\mathbf{b}_\mathbf{a}}^2 \mathbf{I}\right), \tag{11.31}$$

$$\mathbf{N}_n = \text{diag}\left(\sigma_{\text{VEL},\mathbf{v}}^2 \mathbf{I},\ \sigma_{\text{VEL},\mathbf{a}}^2 \mathbf{I},\ \sigma_{\text{ANG}}^2 \mathbf{I},\ \sigma_{\text{LAT}}^2,\ \sigma_{\text{UP}}^2\right), \tag{11.32}$$

where we set $\sigma_\omega = 0.01$ rad/s, $\sigma_{\mathbf{a}} = 0.2\,\text{m/s}^2$, $\sigma_{\mathbf{b}_\omega} = 0.001$ rad/s, $\sigma_{\mathbf{b}_\mathbf{a}} = 0.02\,\text{m/s}^2$ for the noise propagation covariance matrix $\mathbf{Q}_n$, and $\sigma_{\text{VEL},\mathbf{v}} = 1$ m/s, $\sigma_{\text{VEL},\mathbf{a}} = 0.4\,\text{m/s}^2$, $\sigma_{\text{ANG}} = 0.04$ rad/s, $\sigma_{\text{LAT}} = 3$ m/s, and $\sigma_{\text{UP}} = 3$ m/s for the noise measurement covariance matrix $\mathbf{N}_n$.

5.2 5.2 Detector Training

The detector is trained with the sequences urban06 to urban14, that represents 100 km of training data (sequences urban00 to urban05 does not have acceleration data). For each sequence, we compute ground truth position velocity $\mathbf{v}_n^{\text{IMU}}$ and angular velocity $\boldsymbol{\omega}_n$ after differentiating the ground pose and applying smoothing. We then compute the ground-truth $\mathbf{z}_n$ by applying a small threshold on the ground truth velocities, e.g. we consider $z_n^{\text{VEL}} = 1$ if $\|\mathbf{v}_n^{\text{IMU}}\| < 0.01$ m/s. We set similarly the other motion profiles and

[2] https://www.xsens.com/
[3] https://pytorch.org/

test seq.	wheels odo.	odo. + FoG	**RINS-W**
15: 16 min	19 / 5 / 36	**7** / **2** / **7**	**7** / 5 / 12
16: 73 min	140 / 127 / 1166	34 / 20 / 164	**27** / **11** / **20**
17: 19 min	96 / 64 / 427	58 / 51 / 166	**13** / **11** / **13**
15-17: 108 min	114 / 98 / 677	34 / 30 / 152	**22** / **10** / **18**

Table 11.1: Results obtained by the 3 methods on urban test sequences 15, 16, 17 in terms of: m-ATE /aligned m-ATE / final distance error to ground truth, in m. Last line is the concatenation of the three sequences. Direct IMU integration always diverges. The proposed RINS-W outperforms differential wheel speeds based odometry and outperforms on average (see last line) the expensive combination of odometry + FoG. Indeed, RINS-W uses an IMU with gyro stability of 10 deg/h, whereas FoG stability is 0.05 deg/h.

use a threshold of 0.005 rad/s for the angular velocity, and a threshold of 0.1 m/s for the lateral and upward velocities.

The detector is trained during 500 epochs with the ADAM optimizer [162], whose learning rate is initializing at 10^{-3} and managed by a learning rate scheduler. Regularization is enforced with dropout layer, where $p = 0.4$ is the probability of any element to be zero. We use the binary cross entropy loss since we have four binary classification problems. For each epoch, we organize data as a batch of 2 min sequences, where we randomly set the start instant of each sequence, and constraints each starting sequence to be a stop of at minimum 1 s. Training the full detector takes less than one day with a GTX 1080 GPU.

5.3 5.3 Evaluation Metrics

To assess performances we consider three error metrics:

Mean Absolute Trajectory Error (m-ATE): which averages the planar translation error of estimated poses with respect to a ground truth trajectory and is less sensitive to single poor estimates than root mean square error;

Mean Absolute Aligned Trajectory Error (aligned m-ATE): that first aligns the estimated trajectory with the ground truth and then computes the mean absolute trajectory error. This metric evaluates the consistency of the trajectory estimates;

Final distance error: which is the final distance between the un-aligned estimates and the ground truth.

5.4 5.4 Trajectory Results

After training the detector on sequences urban06 to urban14, we evaluate the approach on test sequences urban15 to urban17, that represent 40 km of evaluation data. We compare 4 methods:

- **IMU**: the direct integration of the IMU measurements based on (11.1)-(11.7), that is, pure inertial navigation;

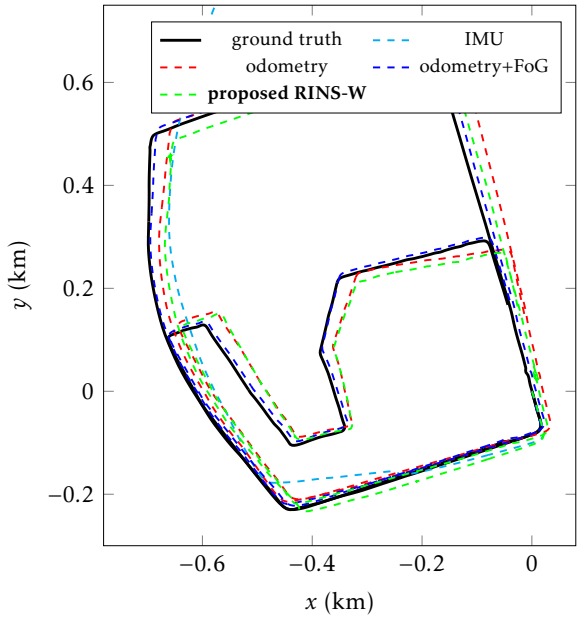


Figure 11.7: Ground truth and trajectory estimates for the test sequence urban15 of the car dataset [199]. RINS-W obtains results comparable with FoG-based odometry.

- **Odometry**: the integration of a differential wheel encoder which computes linear and angular velocities;
- **RINS-W (ours)**: the proposed approach, that uses only the IMU signals and involves no other sensor.
- **Odometry + FoG**: the integration of a differential wheel encoder which computes only linear velocity. The angular velocity is obtained after integrating the increments of an highly accurate and costly KVH DSP-1760[4] Fiber optics Gyro (FoG). The FoG gyro bias stability (0.05 deg/h) is 200 times smaller than the gyro stability of the IMU used by RINS-W;

We delay each sequence such that the trajectory starts with a 1 s stop to initialize the orientation and the biases, see Section 4.2. Bias initialization is *also* performed for IMU pure integration and the FoG. Optimized parameters for wheel speeds sensors calibration are provided by the dataset.

Experimental results in terms of error with respect to ground truth are displayed in Table 11.1, and illustrated on Figures 11.1, 11.7, and 11.8. Results demonstrate that:

- directly integrating IMU leads to divergence at the first turn, even after a careful calibration procedure;
- wheel-based differential odometry accurately estimates the linear velocity but has troubles estimating the yaw during sharp bends, even if the dataset has been obtained in an urban environment and the odometry parameters are calibrated. This drawback may be remedied at the expense of using an additional high-cost gyroscope;

[4] https://www.kvh.com

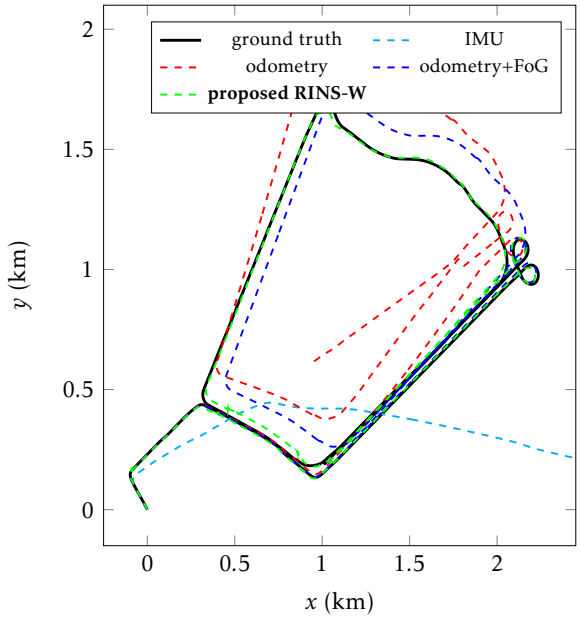


Figure 11.8: Ground truth and trajectory estimates for the test sequence urban17 of the car dataset [199]. RINS-W clearly outperforms the odometry and even the odometry + FoG solution. We note that RINS-W accurately follows the interchange road located at $(x = 2, y = 1)$.

- the proposed scheme completely outperforms wheel encoders, albeit in urban environment. More surprisingly, our approach competes with the combination of wheel speed sensors + (200 hundred times more accurate) Fyber optics Gyro, and even outperforms it on average.

Furthermore, although comparisons were performed in 2D environments our method yields the full 3D pose of the robot, and as such is compatible with non planar environments.

5.5 5.5 Discussion

The performances of RINS-W can be explained by: *i*) a false-alarm free detector; *ii*) the fact incorporating side information into IMU integration obviously yields better results; and *iii*) the use of IEKF that has been proved to be well suited for localization tasks.

We also emphasize the importance of (11.11) and (11.13) in the procedure, i.e. applying (11.25)-(11.26). For illustration, we consider sequence urban07 of [199], where the vehicle moves during 7 minutes without stop so that ZUPT may not be used. We implement the detector trained on the first 6 sequences, and compare the proposed RINS-W to a RINS-W which does *not* use updates (11.25)-(11.26) when detected, see Figure 11.9. Clearly, the reduced RINS-W diverges at the first turn whereas the full RINS-W is accurate along all the trajectory and obtains a final distance w.r.t. ground truth of 5 m. In contrast, odometry + FoG achieves 16 m.

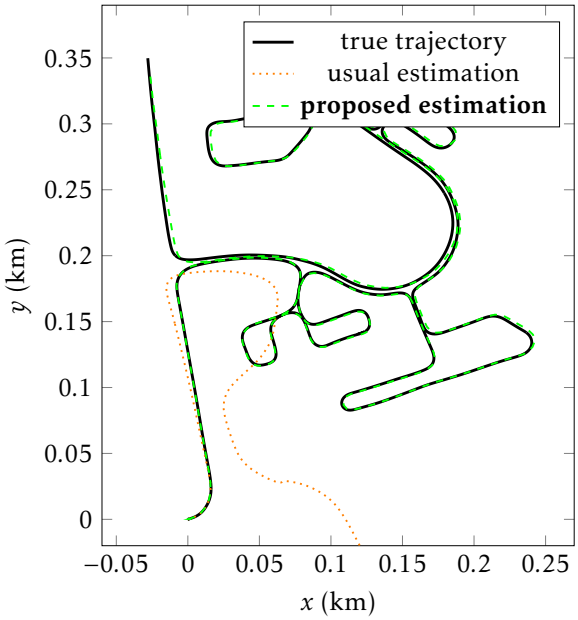


Figure 11.9: Comparison on the sequence urban07 between proposed RINS-W, usual integration that only integrates the IMU measurements without applying pseudo-measurement update. The final distance between ground truth and RINS-W estimates is as small as 5 m, whereas ignoring (11.25)-(11.26) yields divergence.

5.6 5.6 Detector Evaluation

In Section 5.5 we mentioned three possible reasons explaining the performances of RINS-W, but could not assess what is owed to each. To assess the detector's performance, and to demonstrate the interest of our powerful deep neural network based approach (see Section 4.1) we focus on the zero velocity detection (11.9), and compare the detector with the Acceleration-Moving Variance Detector (AMVD) [200] on the test sequences urban15-17, which represent 64.10^4 measurements. The AMVD computes the accelerometer variance over a sample window $W = 10^2$ and assumes the vehicle is stationary if the variance falls below a threshold $\gamma = 10^{-3}$. To make sure AMVD performs at its best, the parameters W and γ are optimized by grid search *on the test sequences*. Results are shown in Table 11.2 and demonstrate the detector is more accurate than this "ideal" AMVD.

6 6 Conclusion

This chapter proposes a novel approach for robust inertial navigation for wheeled robots. Our approach exploits deep neural networks to identify specific patterns in wheeled vehicle motions and incorporates this knowledge in IMU integration for localization. The entire algorithm is fed with IMU signals only, and requires no other sensor. The method leads to surprisingly accurate results, and opens new perspectives for combination of data-driven methods with well established methods for autonomous systems.

z_n^{vel} detection	ideal AMVD	our detector
true positive / false pos.	47.10^4 / 4.10^3	$\mathbf{48.10^4}$ / $\mathbf{7.10^2}$
true negative / false neg.	$\mathbf{16.10^4}$ / 1.10^4	$\mathbf{16.10^4}$ / $\mathbf{9.10^3}$
precision / recall	0.974 / **0.940**	**0.996** / **0.940**

Table 11.2: Results on zero velocity (11.9) detection obtained by an *ideal* AMVD [200] and the proposed detector on test sequences urban15-17. The proposed detector systematically obtains better results and precision. This is remarkable because the detector is not trained on those sequences, whereas AMVD parameters were optimized on the considered test sequences.

Appendix

Following the Right IEKF of [42], the Jacobians required for the computation of the filter propagation (11.20) are given as

$$\mathbf{F}_n = \mathbf{I} + \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathbf{R}_n & \mathbf{0} \\ \mathbf{g}_\times & \mathbf{0} & \mathbf{0} & -(\mathbf{v}_n^{\text{IMU}})_\times \mathbf{R}_n^{\text{IMU}} & -\mathbf{R}_n^{\text{IMU}} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} & -(\mathbf{p}_n^{\text{IMU}})_\times \mathbf{R}_n^{\text{IMU}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} dt, \tag{11.33}$$

$$\mathbf{G}_n = \begin{bmatrix} \mathbf{R}_n^{\text{IMU}} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ (\mathbf{v}_n^{\text{IMU}})_\times \mathbf{R}_n^{\text{IMU}} & \mathbf{R}_n^{\text{IMU}} & \mathbf{0} & \mathbf{0} \\ (\mathbf{p}_n^{\text{IMU}})_\times \mathbf{R}_n^{\text{IMU}} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} dt, \tag{11.34}$$

when $\hat{z}_n^{\text{VEL}} = 0$ and $\hat{z}_n^{\text{ANG}} = 0$. Otherwise, we set the appropriate rows to zero in $\mathbf{F}_n$ and $\mathbf{G}_n$, i.e.:

- if $\hat{z}_n^{\text{VEL}} = 1$ we set the 4 to 9 rows of the right part of $\mathbf{F}_n$ in (11.33) and of $\mathbf{G}_n$ to zero.
- if $\hat{z}_n^{\text{ANG}} = 1$ we set the 3 first rows of the right part of $\mathbf{F}_n$ in (11.33) and of $\mathbf{G}_n$ to zero.

Once again following [42], the measurement Jacobians used in the filter update (11.27)-(11.30) are given as

$$\mathbf{H}_n^{\text{VEL}} = \begin{bmatrix} \mathbf{0} & (\mathbf{R}_n^{\text{IMU}})^T & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ (\mathbf{R}_n^{\text{IMU}})^T \mathbf{g}_\times & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathbf{I} \end{bmatrix}, \tag{11.35}$$

$$\mathbf{H}_n^{\text{ANG}} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathbf{I} & \mathbf{0} \end{bmatrix}, \tag{11.36}$$

and we obtains $\mathbf{H}_n^{\text{LAT}}$ and $\mathbf{H}_n^{\text{UP}}$ as respectively the second and third row of $\mathbf{H}_n^{\text{VEL}}$.

CHAPTER 12

Denoising IMU Gyroscopes with Deep Learning for Open-Loop Attitude Estimation

The present chapter has been accepted in IEEE Robotics and Automation Letters.

Résumé 🇫🇷

Ce chapitre propose une méthode d'apprentissage pour le débruitage des gyroscopes des centrales inertielles pour estimer en temps réel l'orientation d'un robot à l'estime. L'algorithme obtenu surpasse l'état de l'art sur les séquences de test. Les performances sont obtenues grâce à un modèle bien choisi, une fonction de coût appropriée pour des incréments d'orientation, et grâce à l'identification de points clés lors de la formulation du problème avec des données inertielles à haute fréquence. Notre approche s'appuie sur un réseau de neurones basé sur des convolutions dilatées, sans nécessiter de réseau récurrent. Nous démontrons l'efficacité de notre stratégie pour l' estimation d'orientation 3D sur les jeux de données EuRoC et TUM-VI. Fait intéressant, nous observons que notre algorithme réussit à rivaliser avec le meilleurs systèmes d'odométrie visuel-inertiel en termes d'estimation d'attitude bien qu'il n'utilise pas de capteurs de vision. Notre implémentation open-source est disponible à l':

`https://github.com/mbrossar/denoise-imu-gyro`.

Chapter abstract

This chapter proposes a learning method for denoising gyroscopes of IMUs using ground truth data, to estimate in real time the orientation (attitude) of a robot in dead reckoning. The obtained algorithm outperforms the state-of-the-art on the (unseen) test sequences. The obtained performances are achieved thanks to a well chosen model, a proper loss function for orientation increments, and through the identification of key points when training with high-frequency inertial data. Our approach builds upon a neural network based on dilated convolutions, without requiring any recurrent neural network. We demonstrate how efficient our strategy is for 3D attitude estimation on the EuRoC and TUM-VI datasets. Interestingly, we observe our dead reckoning algorithm manages to beat top-ranked visual-inertial odometry systems in terms of attitude estimation although it does not use vision sensors. Our open-source implementation is available at:

`https://github.com/mbrossar/denoise-imu-gyro`.

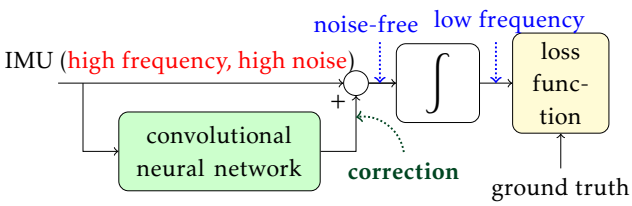


Figure 12.1: Schematic illustration of the proposed method. The convolutional neural network computes gyro corrections (based on past IMU measurements) that filters undesirable errors in the raw IMU signals. We then perform *open-loop* time integration on the noise-free measurements for regressing low frequency errors between ground truth and estimated orientation increments.

1 1 Introduction

IMUs allow estimating a robot's trajectory relative to its starting position, a task called odometry [202]. Small and cheap IMUs are ubiquitous in smartphones, industrial and robotics applications, but suffer from difficulties to estimate sources of error, such as axis-misalignment, scale factors and time-varying offsets [203,204]. Hence, IMU signals are not only noisy, but they are biased. In the present chapter, we propose to leverage deep learning for denoising the gyroscopes (gyros) of an IMU, that is, reduce noise and biases. As a byproduct, we obtain accurate attitude (i.e. orientation) estimates simply by open-loop integration of the obtained noise-free gyro measurements.

1.1 1.1 Links and Differences with Existing Literature

IMUs are generally coupled with complementary sensors to obtain robust pose estimates in sensor-fusion systems. To obtain accurate pose estimates, a proper IMU calibration is required, see e.g. the widely used *Kalibr* library [203,205], which computes offline the underlying IMU intrinsic and extrinsic calibration parameters. Our approach, which is recapped in Figure 12.1, is applicable to any system equipped with an IMU. It estimates offline the IMU calibration parameters and extends methods such as [203,205] to time-varying and difficult to model signal corrections.

Machine learning (more specifically deep learning) has been recently leveraged to perform LiDAR, visual-inertial, and purely inertial localization, where methods are divided into supervised [148,149,206,207] and unsupervised [208]. Most works extract relevant features in the sensors' signals which are propagated in time through recurrent neural networks, whereas [209] proposes convolutional neural networks for pedestrian inertial navigation. A related approach [210] applies reinforcement learning for guiding the user to properly calibrate visual-inertial rigs. Our method is supervised (we require ground truth poses for training), leverages convolutions rather than recurrent architectures, and outperforms the latter approach. We obtain significantly better results while requiring considerably less data and less time. Finally, the reference [207] estimates orientation with an IMU and recurrent neural networks, but our approach proves simpler.

1.2 1.2 Contributions

Our main contributions are as follows:

- detailed modelling of the problem of learning orientation increments from low-cost IMUs;
- the convolutional neural network which regresses gyro corrections and whose features are carefully selected;
- the training procedure which involves a trackable loss function for estimating relative orientation increments;
- the approach evaluation on datasets acquired by a drone and a hand-held device [106,211], where our method outperforms [207] and competes with VIO methods [112,212] although it does not use vision;
- perspectives towards more efficient VIO and IMU based learning methods;
- a publicly available open-sourced code, where training is done in 5 minutes per dataset.

2 2 Kinematic & Low-Cost IMU Models

We detail in this section our model.

2.1 2.1 Kinematic Model based on Orientation Increments

The 3D orientation of a rigid platform is obtained by integrating orientation increments, that is, gyro outputs of an IMU, through

$$\mathbf{R}_n = \mathbf{R}_{n-1} \exp(\boldsymbol{\omega}_n dt), \tag{12.1}$$

where the rotation matrix $\mathbf{R}_n \in SO(3)$ at timestamp n maps the IMU frame to the global frame, the angular velocity $\boldsymbol{\omega}_n \in \mathbb{R}^3$ is averaged during dt, and with $\exp(\cdot)$ the $SO(3)$ exponential map. The model (12.1) successively integrates in open-loop $\boldsymbol{\omega}_n$ and *propagates* estimation errors. Indeed, let $\hat{\mathbf{R}}_n$ denotes an estimate of $\mathbf{R}_n$. The error present in $\hat{\mathbf{R}}_{n-1}$ is propagated in $\hat{\mathbf{R}}_n$ through (12.1).

2.2 2.2 Low-Cost Inertial Measurement Unit (IMU) Sensor Model

The IMU provides noisy and biased measurements of angular rate $\boldsymbol{\omega}_n$ and specific acceleration $\mathbf{a}_n$ at high frequency (200 Hz in our experiments) as, see [203,204],

$$\mathbf{u}_n^{\text{IMU}} = \begin{bmatrix} \boldsymbol{\omega}_n^{\text{IMU}} \\ \mathbf{a}_n^{\text{IMU}} \end{bmatrix} = \mathbf{C} \begin{bmatrix} \boldsymbol{\omega}_n \\ \mathbf{a}_n \end{bmatrix} + \mathbf{b}_n + \mathbf{w}_n, \tag{12.2}$$

where $\mathbf{b}_n \in \mathbb{R}^6$ are quasi-constant biases, $\mathbf{w}_n \in \mathbb{R}^6$ are commonly assumed zero-mean white Gaussian noises, and

$$\mathbf{a}_n = \mathbf{R}_{n-1}^T \left((\mathbf{v}_n - \mathbf{v}_{n-1})/dt - \mathbf{g} \right) \in \mathbb{R}^3 \tag{12.3}$$

is the acceleration in the IMU frame without the effects of gravity $\mathbf{g}$, with $\mathbf{v}_n \in \mathbb{R}^3$ the IMU velocity expressed in the global frame. The intrinsic *calibration* matrix

$$\mathbf{C} = \begin{bmatrix} \mathbf{S}_\omega \mathbf{M}_\omega & \mathbf{A} \\ \mathbf{0}_{3\times 3} & \mathbf{S}_\mathbf{a} \mathbf{M}_\mathbf{a} \end{bmatrix} \approx \mathbf{I}_6 \tag{12.4}$$

contains the information for correcting signals: axis misalignments (matrices $\mathbf{M}_\omega \approx \mathbf{I}_3$, $\mathbf{M}_\mathbf{a} \approx \mathbf{I}_3$); scale factors (diagonal matrices $\mathbf{S}_\omega \approx \mathbf{I}_3$, $\mathbf{S}_\mathbf{a} \approx \mathbf{I}_3$); and linear accelerations on gyro measurements, a.k.a. g-sensitivity (matrix $\mathbf{A} \approx \mathbf{0}_{3\times3}$). Remaining intrinsic parameters, e.g. level-arm between gyro and accelerometer, can be found in [203,204].

We now make three remarks regarding (12.1)-(12.4):

1. equations (12.2)-(12.4) represent a model that *approximates* reality. Indeed, calibration parameters $\mathbf{C}$ and biases $\mathbf{b}_n$ should both depend on time as they vary with temperature and stress [144,204], but are difficult to estimate in real-time. Then, vibrations and platform excitations due to, e.g., rotors make Gaussian noise $\mathbf{w}_n$ colored in practice [213], albeit commonly assumed white;

2. substituting actual measurements ω_n^{IMU} in place of true value ω_n in (12.1) leads generally to quick drift (in a few seconds) and poor orientation estimates;

3. in terms of method evaluation, one should always compare the learning method with respect to results obtained with a properly calibrated IMU as a proper estimation of the parameters $\mathbf{C}$ and $\mathbf{b}_n$ in (12.2) actually leads to surprisingly precise results, see Section 4.

3 3 Learning Method for Denoising the IMU

We describe in this section our approach for regression of noise-free gyro increments $\hat{\omega}_n$ in (12.2) in order to obtain accurate orientation estimates by integration of $\hat{\omega}_n$ in (12.1). Our goal thus boils down to estimating $\mathbf{b}_n$, $\mathbf{w}_n$, and correcting the misknown $\mathbf{C}$.

3.1 3.1 Proposed Gyro Correction Model

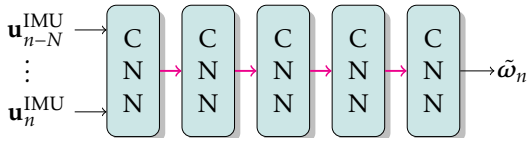
Leveraging the analysis of Section 2, we compute the noise-free increments as

$$\hat{\omega}_n = \hat{\mathbf{C}}_\omega \omega_n^{\text{IMU}} + \tilde{\omega}_n, \tag{12.5}$$

with $\hat{\mathbf{C}}_\omega = \hat{\mathbf{S}}_\omega \hat{\mathbf{M}}_\omega \in \mathbb{R}^{3\times3}$ the intrinsic parameters that account for gyro axis-misalignment and scale factors, and where the gyro bias is included in the gyro correction $\tilde{\omega}_n$. Explicitly considering the small accelerometer influence $\mathbf{A}$, see (12.2)-(12.4), does not affect the results so it is ignored.

We now search to compute $\tilde{\omega}_n$ and $\hat{\mathbf{C}}_\omega$. The neural network described in Section 3.2 computes $\tilde{\omega}_n$ by leveraging information present in a past local window of size N around ω_n^{IMU}. In contrast, we let $\hat{\mathbf{C}}_\omega$ be static parameters initialized at $\mathbf{I}_3$ and optimized during training since each considered dataset uses *one* IMU. The learning problem involving a time varying $\hat{\mathbf{C}}_\omega$ and/or multiple IMUs is let for future works.

The consequences of opting for the simple model (12.5) and the proposed network structure are as follows. First, the corrected gyro may be initialized on the original gyro, i.e. $\hat{\omega}_n \approx \omega_n^{\text{IMU}}$ with $\hat{\mathbf{C}}_\omega = \mathbf{I}_3$ and $\tilde{\omega}_n \approx \mathbf{0}_3$ before training. This way, the method improves the estimates as soon as the first training epoch. Then, our method is intrinsically robust to overfitting as measurements outside the local windows, i.e. whose timestamps are less than $n-N$ or greater than n, see Figure 12.2, do not participate in infering $\tilde{\omega}_n$. This allows us to train the method with 8 or less minutes of data, see Section 4.1.



CNN layer #	1	2	3	4	5
kernel dim.	7	7	7	7	1
dilatation gap	1	4	16	64	1
channel dim.	16	32	64	128	1

Figure 12.2: Proposed neural network structure which computes gyro correction $\tilde{\omega}_n$ in (12.5) from the N past IMU measurements. We set the Convolutional Neural Networks (CNNs) as indicated in the table, and define between two convolutional layers a batchnorm layer [214] and a smooth GELU activation function [215] (magenta arrows). The adopted structure defines the window size $N = \max(\text{kernel dim.} \times \text{dilation gap}) = 7 \times 64 = 448$, corresponding to 2.24 s of past information, and has been found after trial-and-error on datasets [106,211].

3.2 3.2 Dilated Convolutional Neural Network Structure

We define here the neural network structure which infers the gyro correction as

$$\tilde{\omega}_n = f(\mathbf{u}_{n-N}^{\text{IMU}}, \ldots, \mathbf{u}_n^{\text{IMU}}), \tag{12.6}$$

where $f(\cdot)$ is the function defined by the neural network. The network should extract information at temporal multi-scales and compute smooth corrections. Note that, the input of the network consists of IMU data, that is, gyros naturally, but also accelerometers signals. Indeed, from (12.3), if the velocity varies slowly between successive increments we have

$$\begin{aligned} \mathbf{a}_{n+1} - \mathbf{a}_n &\approx -(\mathbf{R}_n - \mathbf{R}_{n-1})^T \mathbf{g} \\ &\approx -(\exp(-\omega_n dt) - \mathbf{I}_3)\mathbf{R}_{n-1}^T \mathbf{g}, \end{aligned} \tag{12.7}$$

which also provides information about angular velocity.

We leverage dilated convolutions that infer a correction based on a local window of $N = 448$ previous measurements, which represents 2.24 s of information before timestamp n in our experiments. Dilated convolutions are a method based on convolutions applied to input with defined dilatation gap, see [216], which: i) supports exponential expansion of the receptive field, i.e., N, without loss of resolution or coverage; ii) is computationally efficient with few memory consumption; and iii) maintains the temporal ordering of data. We thus expect the network to detect and correct various features such as rotor vibrations that are not modeled in (12.2). Our configuration given in Figure 12.2 requires learning 77 052 parameters, which is *extremely* low and contrasts with recent (visual-)inertial learning methods, see e.g. [208] Figure 2, where IMU processing only requires more than 2 600 000 parameters.

3.3 3.3 Loss Function based on Integrated Gyro Increments

Defining a loss function directly based on errors $\omega_n - \hat{\omega}_n$ requires having a ground truth ω_n at IMU frequency (200 Hz), which is not feasible in practice as the best tracking

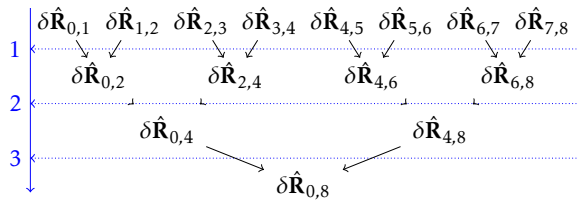


Figure 12.3: Time efficient computation of the loss (12.9) by viewing the orientation integration (12.8) as a tree of matrix multiplications. Computation for length j requires $\log_2(j)$ "batch" operations as denoted by the blue vertical arrow on the left. We see we need 3 batches of parallel operations for $j = 8$ on the chart above. In the same way, we only need 5 operations for $j = 32$.

systems are accurate at 20-120 Hz. In place, we suggest defining a loss based on the following integrated increments

$$\delta\mathbf{R}_{i,i+j} = \mathbf{R}_i^T \mathbf{R}_{i+j} = \prod_{k=i}^{i+j-1} \exp(\omega_k), \tag{12.8}$$

i.e., where the IMU frequency is reduced by a factor j. We then compute the loss for a given j as

$$\mathcal{L}_j = \sum_i \rho\left(\log\left(\delta\mathbf{R}_{i,i+j}\delta\hat{\mathbf{R}}_{i,i+j}^T\right)\right), \tag{12.9}$$

where $\log(\cdot)$ is the $SO(3)$ logarithm map, and $\rho(\cdot)$ is the Huber loss function. We set in our experiments the Huber loss parameter to 0.005, and define our loss function as

$$\mathcal{L} = \mathcal{L}_{16} + \mathcal{L}_{32}. \tag{12.10}$$

The motivations for (12.9)-(12.10) are as follows:

- the choice of Huber loss $\rho(\cdot)$ yields robustness to ground truth outliers;
- (12.8) is invariant to rotations which suits well IMUs, whose gyro and accelerometer measurements are respectively invariant to rotations and yaw changes [202, 217], i.e., left shifting $\mathbf{R}_n \leftarrow \delta\mathbf{R}\mathbf{R}_n$ and $\hat{\mathbf{R}}_n \leftarrow \delta\mathbf{R}\hat{\mathbf{R}}_n$ with $\delta\mathbf{R}_n \in SO(3)$ leaves (12.9) unchanged;
- the choice of (12.10) corresponds to error increments at $200/16 \approx 12$ Hz and $200/32 \approx 6$ Hz, which is barely slower than ground truth. Setting too high a j, or in the extreme case using a loss based on the overall orientation error $\mathbf{R}_n^T\hat{\mathbf{R}}_n$, would make the algorithm prone to overfitting, and hence makes the method too sensitive to specific trajectory patterns of training data.

3.4 3.4 Efficient Computation of (12.8)-(12.10)

First, note that thanks to parallelization applying e.g., $\exp(\cdot)$, to one or parallelly to many $\hat{\omega}_n$ takes similar execution time on a GPU (we found experimentally that one operation takes 5 ms whereas 10 million operations in parallel take 70 ms, that is, the time

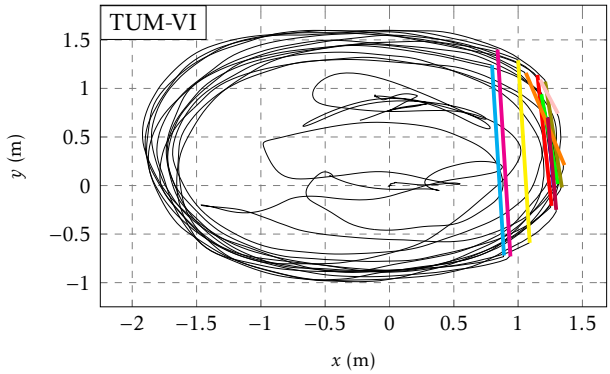


Figure 12.4: Horizontal ground truth trajectory for the sequence `room 1` of [211]. Ground truth is periodically absent due to occlusions of the hand-held device from the motion capture system, see the color lines on the right of the figure.

per operation drops to 7 ns). We call an operation that is parallelly applied to many instances a *batch operation*. That said, an apparent drawback of (12.8) is to require j matrix multiplications, i.e. j operations. However, first, we may compute ground truth $\delta\mathbf{R}_{i,i+j}$ only once, store it, and then we only need to compute $\delta\hat{\mathbf{R}}_{i,i+j}$ multiple times. Second, by viewing (12.8) as a tree of matrix multiplications, see Figure 12.3, we reduce the computation to $\log_2(j)$ batch GPU operations only. We finally apply subsampling and take one i every j timestamps to avoid counting multiple times the same increment. Training speed is thus increased by a factor $32/\log_2(32) \approx 6$.

3.5 3.5 Training with Data Augmentation

Data augmentation is a way to significantly increase the diversity of data available for training without actually collecting new data, to avoid overfitting. This may be done for the IMU model of Section 2 by adding Gaussian noise $\mathbf{w}_n$, adding static bias $\mathbf{b}_n$, uncalibrating the IMU, and shifting the orientation of the IMU in the accelerometer measurement. The two first points were noted in [207], whereas the two latter are to the best of our knowledge novel.

Although each point may increase the diversity of data, we found they do not necessarily improve the results. We opted for addition of a Gaussian noise (only), during each training epoch, whose standard deviation is the half the standard deviation that the dataset provides (0.01 deg/s).

4 4 Experiments

We evaluate the method in term of 3D orientation and yaw estimates, as the latter are more critical regarding long-term odometry estimation [202,218].

4.1 4.1 Dataset Descriptions

We divide data into training, validation, and test sets, defined as follows, see Chapter I.5.3 of [161]. We optimize the neural network and calibration parameters on the training set. Validation set intervenes when training is over, and provides a *biased* evaluation, as the validation set is used for training (data are seen, although never used for

dataset	sequence	VINS-Mono [112]	VINS-Mono (loop-closure)	Open-VINS [212]	Open-VINS (**proposed**)
EuRoC [106]	MH 02 easy	1.34/1.32	**0.57**/**0.50**	1.11/1.05	1.21/1.12
	MH 04 difficult	1.44/1.40	**1.06**/1.00	1.60/1.16	1.40/0.89
	V1 01 easy	0.97/0.90	**0.57**/**0.44**	0.80/0.67	0.80/0.67
	V1 03 difficult	4.72/4.68	4.06/4.00	2.32/2.27	**2.25**/2.20
	V2 02 medium	2.58/2.41	1.83/1.61	1.85/1.61	**1.81**/**1.57**
	average	2.21/2.14	1.62/1.52	1.55/1.37	**1.50**/1.30
TUM-VI [211]	room 2	**0.60**/**0.45**	0.69/0.50	2.47/2.36	1.95/1.84
	room 4	0.76/0.63	**0.66**/**0.51**	0.97/0.88	0.93/0.83
	room 6	0.58/0.38	**0.54**/**0.33**	0.63/0.51	0.60/0.51
	average	0.66/0.49	**0.63**/**0.45**	1.33/1.25	1.12/1.05

dataset	sequence	zero motion	raw IMU	OriNet* [207]	calibrated IMU (**proposed**)	**proposed IMU**
EuRoC [106]	MH 02	44.4/43.7	146/130	5.12/–	7.09/1.49	1.39/0.85
	MH 04	42.3/41.9	130/77.9	7.77/–	5.64/2.53	1.40/**0.25**
	V1 01	114/76	71.3/71.2	5.01/–	6.65/3.95	1.13/0.49
	V1 03	119/84.9	120/74.5	13.2/–	3.56/2.04	2.70/**0.96**
	V2 02	93.9/93.5	117/86	9.59/–	4.63/2.30	3.85/2.25
	average	125/89.0		7.70/–	5.51/2.46	2.10/**0.96**
TUM-VI [211]	room 2	91.8/90.4	118/88.1	–/–	10.6/10.5	1.31/1.18
	room 4	107/103	74.1/48.2	–/–	2.43/2.30	1.48/0.85
	room 6	138/131	94.0/76.1	–/–	4.39/4.31	1.04/0.57
	average	112/108	95.7/70.8	–/–	5.82/5.72	1.28/0.82

Table 12.1: Absolute Orientation Error (AOE) in terms of **3D orientation/yaw**, in degree, on the *test* sequences. We see our approach competes with VIO (while entirely based on IMU signals) and outperforms other inertial methods. (*) Note that, results from OriNet correspond instead to the mean orientation error $\sum_{n=1}^{M} \|\log(\mathbf{R}_n^T \hat{\mathbf{R}}_n)\|_2 / M$ which is by definition always less than AOE criterion, and are unavailable on the TUM-VI dataset.

"learning"). The test set is the gold standard to provide an *unbiased* evaluation. It is only used once training (using the training and validation sets) is terminated. The datasets we use are as follows.

EuRoC: the dataset [106] contains image and inertial data at 200 Hz from a *micro aerial vehicle* divided into 11 flight trajectories of 2-3 minutes in two environments. The ADIS16448 IMU is *uncalibrated* and we note ground truth from laser tracker and motion capture system is accurately time-synchronized with the IMU, although dynamic motions deteriorate the measurement accuracy. As yet noticed in [212], ground truth for the sequence `V1 01 easy` needs to be recomputed.

We define the train set as the first 50 s of the six sequences `MH{01,03,05}`, `V1{02}`, `V2{01,03}`, the validation set as the remaining ending parts of these sequences, and we constitute the test set as the five remaining sequences. We show in Section 4.5 that using only 8 minutes of accurate data for training - the beginning and end of each trajectory are the most accurately measured - is sufficient to obtain relevant results.

TUM-VI: the recent dataset [211] consists of visual-inertial sequences in different scenes from an *hand-held* device. The cheap BMI160 IMU logs data at 200 Hz and was properly *calibrated*. Ground truth is accurately time-synchronized with the IMU, although each sequence contains periodic instants of duration 0.2 s where ground truth is *unavailable* as the acquisition platform was hidden from the motion capture system, see Figure 12.4. We take the 6 room sequences, which are the sequences having longest ground truth (2-3 minutes each).

We define the train set as the first 50 s of the sequences `room 1`, `room 3`, and `room 5`, the validation set as the remaining ending parts of these sequences, and we set the test set as the 3 other room sequences. This slipt corresponds to 45 000 training data points (90 000 for EuRoC) which is in the same order as the number of optimized parameters, 77 052, and requires regularization techniques such as weight decay and dropout during training.

4.2 4.2 Method Implementation & Training

Our open-source method is implemented on PyTorch 1.5, where we configure the training hyperparameters as follows. We set weight decay with parameter 0.1, and dropout with 0.1 the probability of an element to be set equal to zero. Both techniques reduce overfitting.

We choose the ADAM optimizer [162] with cosines warning restart scheduler [219] where learning rate is initialized at 0.01. We train for 1800 epochs, which is is very fast as it takes less than 5 minutes for each dataset with a GTX 1080 GPU device.

4.3 4.3 Compared Methods

We compare a set of methods based on camera and/or IMU.

Methods Based on the IMU Only: we compare the following approaches:

- **raw IMU**, that is an uncalibrated IMU. It refers also to the proposed method once initialized but not trained;
- **OriNet** [207], which is based on *recurrent neural networks*, and whose validation set corresponds to the test set (our training setting is thus more challenging);

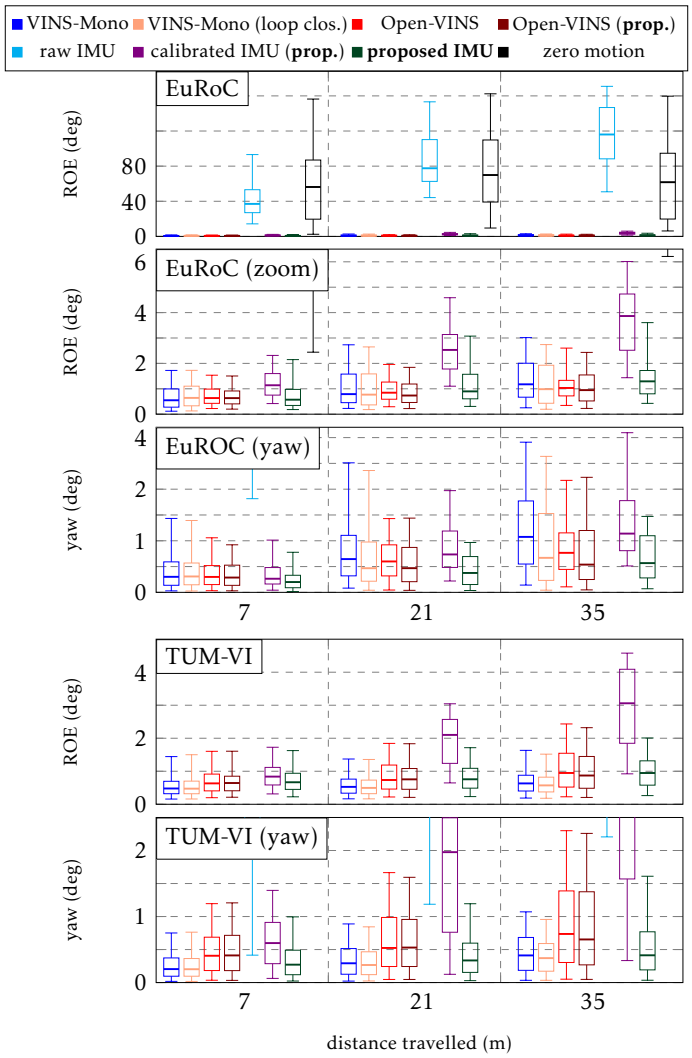


Figure 12.5: Relative Orientation Error (ROE) in terms of 3D orientation and yaw errors on the test sequences. Our method outperforms calibrated IMU and competes with VIO methods albeit based only on IMU signals. Raw IMU and zero motion are way off. Results from OriNet are unavailable.

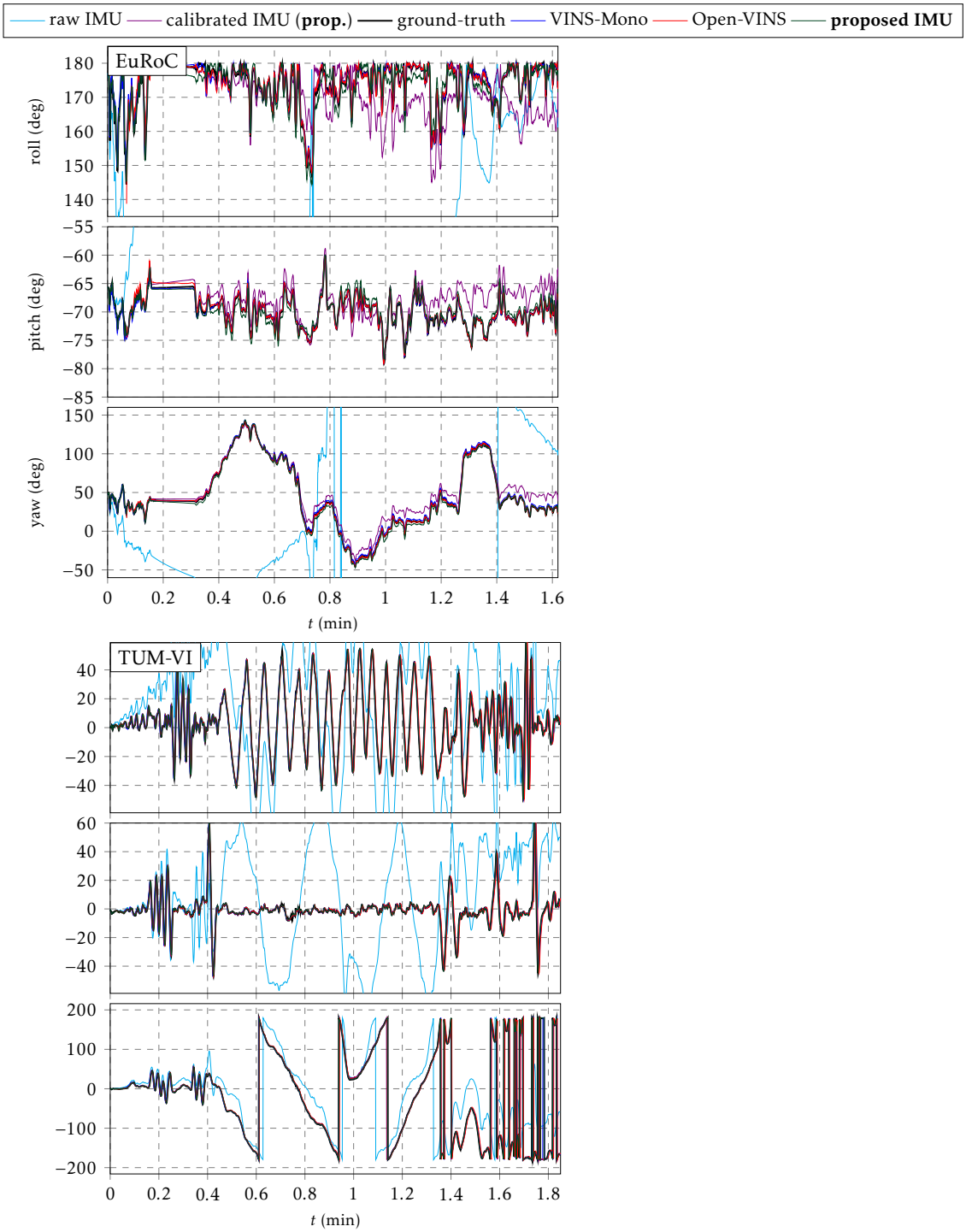


Figure 12.6: Orientation estimates on the test sequence `MH 04 difficult` of [106] (left), and `room 4` of [211] (right). Our method removes errors of the calibrated IMU and competes with VIO algorithms.

- **calibrated IMU**, that is, our method where the 12 parameters $\hat{\mathbf{C}}_\omega$ and $\tilde{\omega}_n$ are *constant*, nonzero, and optimized;
- **proposed IMU**, which is our learning based method described in Section 3.

Methods Based on Camera and the IMU: we run each of the following method with the same setting, ten times to then average results, and on a Dell Precision Tower 7910 workstation desktop, i.e., without deterioration due to computational limitations [218]. We compare:

- **VINS-Mono** [112], a monocular VIO framework with notable performances on the EuRoC benchmark [218];
- **VINS-Mono (loop closure)**, which is the original VINS-Mono [112] reinforced with loop-closure ability;
- **Open-VINS** [212], a versatile filter-based visual-inertial estimator for which we choose the stereo configuration, and that is top-ranked on the drone dataset of [220];
- **Open-VINS (proposed)**, which is Open-VINS of [212] but where gyro inputs are the proposed corrected gyro measurements (12.5) output by our method (trained on sequences that are of course different from those used for evaluation).

Remaining Methods: we finally add a basic **zero motion**, that is $\omega_n = \mathbf{0}_3$ considered as the standard prior in visual odometry when IMU is not available.

4.4 4.4 Evaluation Metrics

We evaluate the above methods using the following metrics that we compute with the toolbox of [212].

Absolute Orientation Error (AOE): which computes the mean square error between the ground truth and estimates for a given sequence as

$$\text{AOE} = \sqrt{\sum_{n=1}^{M} \frac{1}{M} \|\log\left(\mathbf{R}_n^T \hat{\mathbf{R}}_n\right)\|_2^2}, \tag{12.11}$$

with M the sequence length, $\log(\cdot)$ the $SO(3)$ logarithm map, and where the estimated trajectory has been aligned on the ground truth at the first instant $n = 0$.

Relative Orientation Error (ROE): which is computed as [217]

$$\text{ROE} = \|\log\left(\delta\mathbf{R}_{n,g(n)}^T \delta\hat{\mathbf{R}}_{n,g(n)}\right)\|_2, \tag{12.12}$$

for each pair of timestamps $(n, g(n))$ representing an IMU displacement of 7, 21 or 35 meters. Collecting the error (12.12) for all the pairs of sub-trajectories generates a collection of errors where informative statistics such as the median and percentiles are calculated. As [212,217,218], we strongly recommend ROE for comparing odometry estimation methods since AOE is highly sensitive to the time when the estimation error occurs. We finally consider slight variants of (12.11)-(12.12) when considering *yaw* (only) errors, and note that errors of visual methods generally scale with distance travelled whereas errors of inertial only methods scales with time. We provide in the present chapter errors w.r.t. distance travelled to favor comparison with benchmarks such as [218], and same conclusions hold when computing ROE as function of different times.

4.5 4.5 Results

Results are given in term of AOE and ROE respectively in Table 12.1 and Figure 12.5. Figure 12.6 illustrates roll, pitch and yaw estimates for a test sequence of each dataset, and Figure 12.7 shows orientation *errors*. We note that:

Uncalibrated IMU is Unreliable: raw IMU estimates deviate from ground truth in as low as 10 s, see Figure 12.6, and are barely more reliable than null rotation assumption.

Calibrated IMU Outperforms OriNet: only calibrating an IMU (via our optimization method) leads to surprisingly accurate results, see e.g., Figure 12.6 (right) where it is difficult to distinguish it from ground truth. This evidences cheap sensors can provide very accurate information once they are correctly calibrated.

The Proposed Method Outperforms Inertial Methods: OriNet [207] is outperformed. Moreover, our method improves accurate calibrated IMU by a factor 2 to 4. Our approach notably obtains as low as a median error of 1.34 deg/min and 0.68 deg/min on respectively EuRoC and TUM-VI datasets.

The Proposed Method Competes with VIO: our IMU only method is accurate even on the high motion dynamics present in both datasets, see Figure 12.6, and competes with VINS-Mono and Open-VINS, although trained with only a few minutes of data.

Finally, as the performance of each method depends on the dataset and the algorithm setting, see Figure 12.5, it is difficult to conclude which VIO algorithm is the best.

4.6 4.6 Further Results and Comments

We provide a few more comments, supported by further experimental results.

Small Corrections Might Lead to Large Improvement: the calibrated and corrected gyro signals are visually undistinguishable: differences between them rely in corrections $\tilde{\omega}_n$ of few deg/s, as shown in Figure 12.8. However, they bring drastic improvement in the estimates. This confirms the interest of leveraging neural networks for model correction (12.2)-(12.4).

The Proposed Method is Well Suited to Yaw Estimation: according to Table 12.1 and Figure 12.5, we see yaw estimates are particularly accurate. Indeed, VIO methods are able to recover at any time roll and pitch thanks to accelerometers, but the yaw estimates drift with time. In contrast our dead-reckoning method never has access to information allowing to recover roll and pitch during testing, and nor does it use "future" information such as VINS-Mono with loop-closure ability. We finally note that accurate yaw estimates could be fruitful for yaw-independent VIO methods such as [82].

Correcting Gyro Slightly Improves Open-VINS [212]: both methods based on Open-VINS perform similarly, which is not surprising as camera alone already provides accurate orientation estimates and the gyro assists stereo cameras.

Figure 12.7: Orientation *errors* on the sequence `room` 4 of [211]. Our method removes errors of the calibrated IMU and competes with VIO algorithms.

Our Method Requires few Computational Ressources: each VIO method performs here at its best while resorting to high computational requirements, and we expect our method - once trained - is very attractive when running onboard with limited resources. Note that, the proposed method performs e.g. 3 times better in terms of yaw estimates than a slightly restricted VINS-Mono, see Figure 3 of [218].

5 5 Discussion

We now provide the community with feedback regarding the method and its implementation. Notably, we emphasize a few points that seem key to a successful implementation when working with a low-cost high frequency IMU.

5.1 5.1 Key Points Regarding the Dataset

One should be careful regarding the quality of data, especially when IMU is sampled at high-frequency. This concerns:

IMU Signal: the IMU signal acquisition should be correct with constant sampling time.

Ground Truth Pose Accuracy: we note that the EuRoC ground truth accuracy is better at the beginning of the trajectory. As such, training with only this part of data (the first 50 s of the training sequences) is sufficient (and best) to succeed.

Ground Truth Time-Alignement: the time alignment between ground truth and IMU is significant for success, otherwise the method is prone to learn a time delay.

We admit that our approach requires a proper dataset, which is what constitutes its main *limitation*.

5.2 5.2 Key Points Regarding the Neural Network

Our conclusions about the neural network are as follows.

Activation Function: the GELU and other smooth activation functions [215], such as ELU, perform well, whereas ReLU based network is more prone to overfit. We believe ReLU activation function favors sharp corrections which does not make sense when dealing with physical signals.

Neural Network Hyperparameters: increasing the depth, channel and/or kernel sizes of the network, see Figure 12.2, does not systematically lead to better results. We tuned these hyperparameters with random search, although more sophisticated methods such as [221] exist.

Normalization Layer: batchnorm layer improves both training speed and accuracy [214], and is highly recommended.

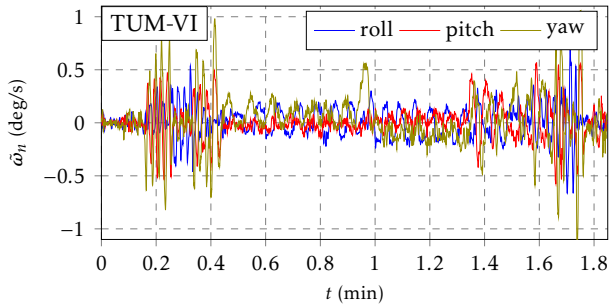


Figure 12.8: Gyro correction $\tilde{\omega}_n$ on the sequence `room 4` of [211]. We see we manage to divide orientation errors by a factor at least 2 w.r.t. calibrated IMU applying corrections whose amplitude is as low as 1 deg/s (max).

5.3 5.3 Key Points Regarding Training

As in any machine learning application, the neural network architecture is a key component among others [161]. Our comments regarding training are as follows:

Optimizer: the ADAM optimizer [162] performs well.

Learning Rate Scheduler: adopting a learning rate policy with cosinus warning restart [219] leads to substantial improvement and helps to find a correct learning rate.

Regularization: dropout and weight decay hyperparameters are crucial to avoid overfitting. Each has a range of ideal values which is quickly tuned with basic grid-search.

5.4 5.4 Remaining Key Points

We finally outline two points that we consider useful to the practitioner:

Orientation Implementation: we did not find any difference between rotation matrix or quaternion loss function implementation once numerical issues are solved, e.g., by enforcing quaternion unit norm. Both implementations result in similar accuracy performance and execution time.

Generalization and Transfert Learning: it may prove useful to assess to what extent a learning method is generalizable. The extension of the method, trained on one dataset, to another device or to the same device on another platform is considered as challenging, though, and left for future work.

6 6 Conclusion

This chapter proposes a deep-learning method for denoising IMU gyroscopes and obtains remarkable accurate attitude estimates with only a low-cost IMU, that outperforms state-of-the-art [207]. The core of the approach is based on a careful design and feature selection of a dilated convolutional network, and an appropriate loss function leveraged for training on orientation increment at the ground truth frequency. This

leads to a method robust to overfitting, efficient and fast to train, which serves as offline IMU calibration and may enhance it. As a remarkable byproduct, the method competes with state-of-the-art visual-inertial methods in term of attitude estimates on a drone and hand-held device datasets, where we simply integrate noise-free gyro measurements.

CHAPTER 13

Additional Results for Inertial Navigation of Cars

In this Chapter, we show unpublished additional results for inertial navigation of car. The approach is a direct extension of the Chapter 11 where the noise covariance matrix of the filter is adapted by the method described in Chapter 8.

1 1 Introduction

We show here a modular method for odometry localization of ground vehicles, whose overview is depicted in Figure 13.1, that requires only an IMU. The approach identifies instants where the vehicle is stopping from raw sensor signals and leverages them through an invariant Kalman filter for robust state estimation of vehicle pose, vehicle velocity and IMU biases.

We perform substantial evaluations on three vehicle datasets, which represent a total amount of 366 km of data acquired during 15 hours in diverse environments such as cities, highway and countries, where the vehicle moves up to 25 m/s during three sequences of minimum 70 minutes long on our own specifically dedicated experiments.

The method obtain impressively accurate results that exhibit how sufficient a medium-cost IMU contains information for odometry localization.

2 2 Experimental Results

We evaluate the performances of the proposed approach on three datasets. Our primary goal in this section is to show that using an IMU of moderate cost, one manages to obtain surprisingly accurate dead-reckoning by combining machine learning techniques with a state-of-the-art Kalman filter.

2.1 2.1 Dataset Descriptions

The three datasets have been acquired on distinct vehicles, each of them being equipped with its proper sensor suite, see Figure 13.2. We dispose of a total amount of 366 km high-quality experimental data acquired during more than 15 h with a great diversity in terms of vehicle dynamics, sensors, and external environments, see Table 13.1. We gives in the following a brief description of each dataset.

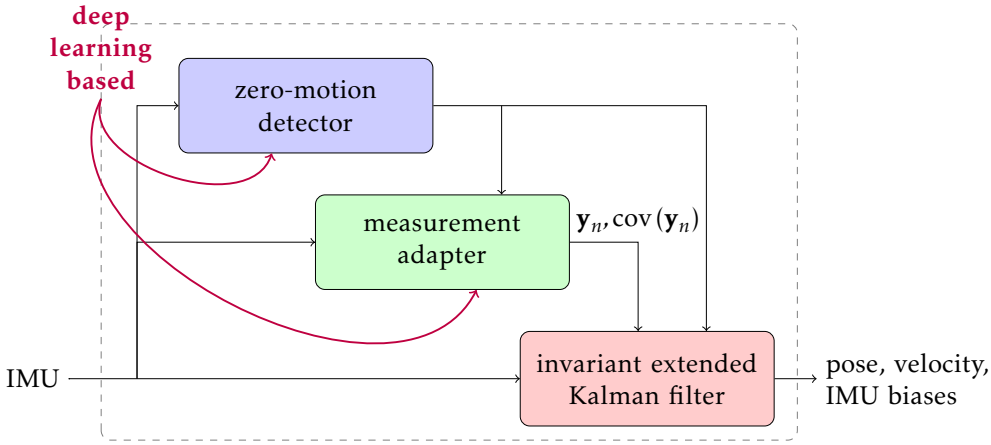


Figure 13.1: Structure of the proposed approach for inertial based navigation, which requires only an IMU. The detector (blue) feeds zero-motion information to the measurement adapter and the filter from raw sensor data. The adapter (green) outputs a vector of observations $\mathbf{y}_n$ along with its uncertainty covariance matrix $\mathrm{cov}(\mathbf{y}_n)$, both are inputs of the filter (red) that is in charge of state estimation.

Figure 13.2: The approach is evaluated on three distinct datasets: from left to right, images of the vehicle in the publicly available *KAIST* dataset [199,222], the *Lille* dataset which is publicly available for point cloud classification, see [223,224], and our *Magny-Les-Hameaux* (MLH) dataset. Each dataset disposes among other of IMU data, ground-truth poses and refers to diverse vehicle dynamics and environments.

dataset	length (km)	duration (min)	environment
KAIST	235	518	urban, highway
Lille	28	132	urban
MLH (ours)	103	242	country
total	366	892 (~15 h)	

Table 13.1: Description of the considered datasets. They dispose all of IMU, centimeter position ground-truth, represent various vehicle dynamics, sensors, and consist of a tremendous amount of data targeting diverse environments.

dataset	odo. ATE (m)	odo. *a*-ATE (m)	odo. AVE (m/s)	**IMU** ATE (m)	**IMU** *a*-ATE (m)	**IMU** AVE (m/s
KAIST	363	145	1.98	61	25	0.55
Lille	-	-	-	**7.1**	**4.2**	**0.12**
MLH	3.10^3	831	20	117	45	0.33

Table 13.5: Results on different datasets in terms of mean-Absolute Trajectory Error (ATE), aligned mean-Absolute Trajectory Error (*a*-ATE) and mean-Absolute Velocity Error (AVE), where we discard INS results since purely inertial integration often diverges. **IMU** performs particularly well in cities on the KAIST and Lille datasets, keep robust on our challenging MLH dataset, and never diverge. They completely outperform standard wheel odometry (odo.).

KAIST Dataset [199,222]: The *KAIST Urban Data Set* is a publicly available dataset that provides IMU, fiber optic gyro, vision, radar, encoder and altimeter data targeting the highly complex urban environment around Daejong, Korea. The Xsens MTi-300 IMU logs data at 100 Hz and has a 10 deg/h gyro bias stability. We download 32 sequences `urban06` to `urban37` and exclude the remaining sequences that are provided without accelerometer measurements.

Lille Dataset: The *Paris-Lille-3D* dataset [223,224] is a publicly available urban point cloud dataset for automatic segmentation from which we recover IMU data along with centimeter accurate ground-truth pose for trajectories located in the city of Lille, France. The vehicle is a Citroën Jumper van equipped with a precise Novatel FlexPak 6 dual-phase RTK-GPS and an Ixsea LANDINS IMU whose signal is sampled at 100 Hz. The dataset contains 28 km of experiments divided into 8 sequences up to 20 min long.

Magny-Les-Hameaux Dataset (MLH): We obtain experiments from the Safran[1] company, its R&T center "SafranTech" located at Magny-Les-Hameaux, France, and its laboratory specifically dedicated for autonomous vehicles. We collect IMU, GPS, and wheel encoder data from a Citroën Picasso in the countryside environment around the R&T center. The Epsilon10 IMU[2] is a medium-cost device specifically dedicated for hybrid land navigation and orientation systems, and the Geoflex RTK-GPS provides ground-truth poses. Three sequences of minimum 70 min long have been recorded, representing a total of 105 km data with relatively high vehicle velocity (25 m/s), speed bumps and speed reducers.

2.2 2.2 Evaluation

This section evaluates the dead-reckoning method on the KAIST, Lille and MLH datasets. We benchmark our method in this section considering only IMU signals.

Evaluation Metrics and Compared Methods: To assess performances we consider three error metrics:

[1] `https://www.safran-group.com`
[2] `https://www.safran-electronics-defense.com/security/navigation-systems`

test seq.	length (km)	duration (min)	odo. ATE (m)	odo. *a*-ATE (m)	odo. AVE (m/s)	**IMU** ATE (m)	**IMU** *a*-ATE (m)	**IMU** AVE (m/s)
06	19	30	3.10^3	794	11.8	126	38	0.40
07	2	9	10	9	0.64	**3**	**1**	**0.06**
09	15	56	274	107	2.18	35	12	0.35
10	14	40	77	76	1.40	43	35	0.43
11	7	7	158	62	1.46	**42**	12	0.52
12	8	34	214	144	0.96	123	52	0.38
13	2	23	10	6	0.25	**2**	**2**	0.06
14	7	28	219	137	1.41	27	15	0.17
15	5	16	27	12	0.56	10	10	0.17
16	21	73	140	83	1.71	**13**	7	0.15
17	10	19	78	30	0.82	**10**	8	0.23
26	4	9	58	25	0.72	9	7	0.20
27	5	19	73	59	1.02	110	59	0.38
28	11	32	445	231	4.42	33	20	0.33
29	3	7	123	15	0.69	9	7	0.20
30	5	19	112	126	0.94	31	30	0.49
31	11	15	512	275	3.01	50	**26**	0.77
32	6	17	66	32	0.82	7	7	0.19
33	7	21	71	57	0.92	17	13	0.17
34	5	4	50	13	0.86	13	8	0.25
average	185	478	345	138	1.52	40	20	0.29

Table 13.6: Results on the KAIST dataset, where we discard INS results since purely inertial integration often diverges. **IMU** outperforms odometry based on wheels. The vehicle starts by a minimum of 5 s stop in all these sequences that is sufficient for the proposed method to self-initialize orientation and sensor biases.

test seq.	length (km)	tion (min)	odo. ATE (m)	odo. *a*-ATE (m)	odo. AVE (m/s)	**IMU** ATE (m)	**IMU** *a*-ATE (m)	**IMU** AVE (m/s)
08	2	5	16	7	0.69	4	3	0.35
18	4	3	212	12	3.20	50	30	1.80
19	3	2	127	22	3.11	88	36	1.72
20	3	2	299	94	9.32	99	58	2.52
21	4	2	10^3	322	31	177	85	3.95
22	3	2	10^3	175	26	207	126	5.80
23	3	2	310	90	10	85	30	3.30
24	4	2	10^3	211	20	206	93	3.94
25	2	2	145	9	3.41	90	41	3.56
35	3	3	57	4	0.81	139	38	2.63
36	9	6	943	678	12.6	148	40	1.97
37	11	9	10^3	344	8.05	10^3	242	8.07
average	50	40	583	227	7.5	309	89	3.7

Table 13.7: Results on the KAIST dataset. We discard INS results since purely inertial integration often diverges. The proposed **IMU** outperforms the odometry. The vehicle is always moving in these sequences and the proposed method manage to early initialize sensor biases and state error covariance which constitutes itself a notable feature.

test seq.	length (km)	duration (min)	INS ATE (m)	INS *a*-ATE (m)	INS AVE (m/s)	**IMU** ATE (m)	**IMU** *a*-ATE (m)	**IMU** AVE (m/s)
00	6	23	10.10^3	7.10^3	22	**8**	**4**	**0.12**
01	2	10	6.10^3	5.10^3	33	**4**	**1**	**0.07**
02	1	6	10^3	10^3	6	**2**	**2**	**0.13**
03	4	21	18.10^3	13.10^3	66	**14**	**12**	**0.14**
04	4	23	25.10^3	22.10^3	74	**42**	**25**	**0.09**
05	4	22	23.10^3	23.10^3	105	**8**	**5**	**0.10**
06	1	6	10^3	10^3	8	**2**	**1**	**0.07**
07	6	21	32.10^3	24.10^3	103	**16**	**12**	**0.16**
average	28	132	18.10^3	15.10^3	64	**7.1**	**4.2**	**0.12**

Table 13.8: Results on the Lille dataset. Purely inertial integration (INS) tends to diverge, whereas the proposed **IMU** keep accurate for the longest 20 min long sequences. Both methods uses only the IMU. The proposed method particularly well estimates curved bends and roundabouts, see Figure 13.18.

test seq.	length (km)	duration (min)	odo. ATE (m)	odo. *a*-ATE (m)	odo. AVE (m/s)	**IMU** ATE (m)	**IMU** *a*-ATE (m)	**IMU** AVE (m/s)
00	38	70	5700	300	10	151	49	0.48
01	27	71	2100	1400	50	106	42	0.34
02	38	101	2000	800	7	102	44	0.21
average	103	242	3100	831	20	117	45	0.33

Table 13.9: Results on our specifically dedicated Magny-Les-Hameaux (MLH) dataset. We discard INS results since purely inertial integration often diverges. The proposed **IMU** outperforms the odometry. Such results are impression on this countryside dataset which contains many speed bump and wheel slip in long sequences at relatively high speed (25 m/s).

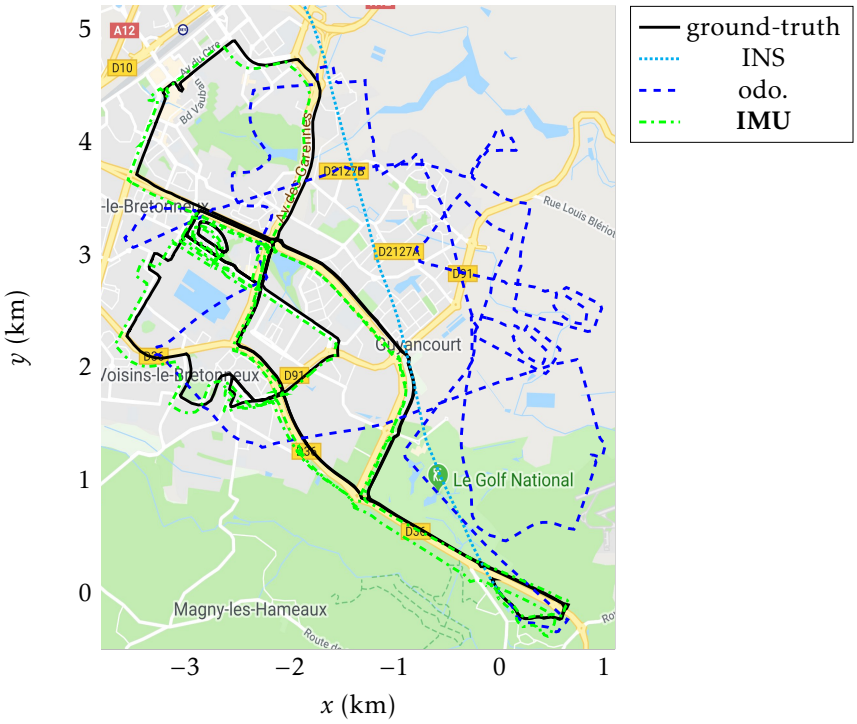


Figure 13.15: Trajectory estimates on the sequence 01 on the MLH dataset (27 km, 71 min). **IMU** follows ground-truth for this particularly long experiment.

ATE: mean-Absolute Trajectory Error, which averages the translation error of estimated poses w.r.t. a ground-truth trajectory,

$$\text{ATE} = \frac{1}{N}\sum_{n=0}^{N}\left\|\mathbf{p}_n^{\text{IMU}} - \hat{\mathbf{p}}_n^{\text{IMU}}\right\|_1; \tag{13.1}$$

a-**ATE:** aligned mean-Absolute Trajectory Error, that first aligns the estimated trajectory with the ground-truth and then computes the mean-absolute trajectory error,

$$a\text{-ATE} = \frac{1}{N}\sum_{n=0}^{N}\|\mathbf{p}_n^{\text{IMU}} - \mathbf{R}\hat{\mathbf{p}}_n^{\text{IMU}} - \mathbf{p}\|_1, \tag{13.2}$$

where $\mathbf{R}, \mathbf{p} = \arg\min_{\mathbf{R},\mathbf{p}}\left(\sum_{n=0}^{N}\|\mathbf{p}_n^{\text{IMU}} - \mathbf{R}\hat{\mathbf{p}}_n^{\text{IMU}} - \mathbf{p}\|_1\right)$ is computed by Umeyama alignment. This metric evaluates the full consistency of the trajectory estimates and is less affected than ATE by precocious yaw misalignment;

AVE: mean-Absolute Velocity Error, which averages the estimated speed w.r.t. a ground-truth velocity,

$$\text{AVE} = \frac{1}{N}\sum_{n=0}^{N}\|\mathbf{v}_n^{\text{IMU}} - \hat{\mathbf{v}}_n^{\text{IMU}}\|_1, \tag{13.3}$$

and indicates the ability of a method for regressing velocity in the world frame.

The three above l_1 metrics are less sensitive to poor estimates than root mean square error and allow correcting position when zero-motion is detected. We compare on the aforementioned metrics three methods:

INS: the direct integration of the inertial measurements;

odo.: the integration of a differential wheel encoder which computes linear and angular velocities, a.k.a. wheel-based odometry;

IMU (ours): the proposed approach, that uses only the IMU signals and involves no other sensor, that is our purely inertial navigation system.

2.3 2.3 Results

Results are averaged in Table 13.5 and detailed for each dataset in Tables 13.6, 13.7, 13.8 and 13.9. Figures 13.15, 13.16, 13.17, and 13.18 illustrate diverse trajectory results. From these results, we see that:

- directly integrating the IMU signals leads to rapid drift of the estimates;
- wheel-based differential odometry accurately estimates the linear velocity but encounters troubles estimating the yaw, even in Figure 13.17 whose data has been obtained in an urban environment and provided with calibrated odometry intrinsics;
- the proposed schemes completely outperforms wheel based odometry in urban, countryside and highway environments.

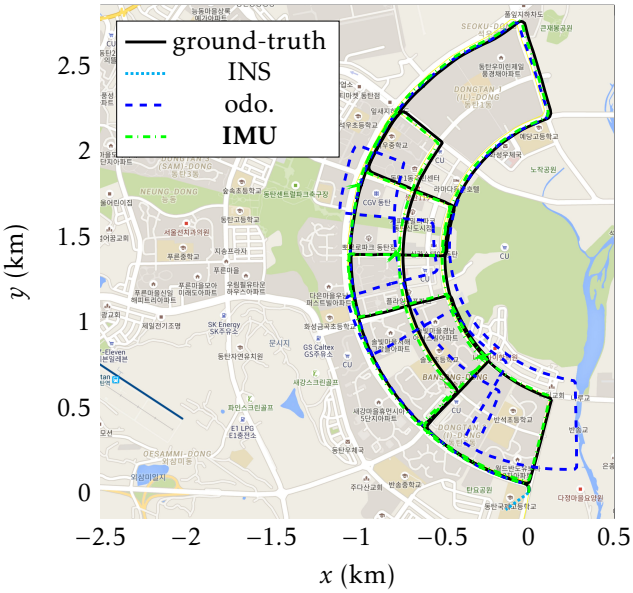


Figure 13.16: Trajectory estimates on the sequence urban10 of the KAIST dataset (14 km, 40 min). **IMU** keeps accurate methods and follows many corner bends.

The results are remarkable as we use for each dataset an IMU which has moderate precision and is targeted for hybridation with GPS. Beyond these unexpected long-term accurate position estimations, one may benefit from the method for promptly calibrating inherent and strong IMU biases whose magnitude is greater than 10^{-3} rad/s and 10^{-2} m/s^2 on the KAIST dataset. This hold when the car is first stopping and the first motions contain sufficiently turns and accelerations that render IMU bias observable. However, when both the car starts by moving and follows a straight highway, one can not identify IMU bias and the estimation drifts, which is similarly encountered in visual-inertial odometry when degenerate motions occur, see [225]. This explains the results obtained in Table 13.7. Finally, the AVE metrics confirms us that the method is a particularly precise speedometer.

3 3 Further Experimental Results

We provide in this section experimental results that completes the dead-reckoning evaluation performed in Section 2. We analyze the zero-motion detector, and inspects the computational execution times of the method. All the results are based on the three datasets described in Section 2.1.

3.1 3.1 Detector Results

To assess the performances of our learning based detection approach (see Section 4.1), we focus on the zero velocity detection (11.9), and compare the proposed detectors with two popular detectors [200,226]. To make sure such traditional detectors perform at their best, their parameters are optimized by grid search on each full dataset, i.e. with test sequences, whereas our detectors still learn only on training data. Since generalization to new type of data is a cumbersome task for deep learning based algo-

Figure 13.17: Trajectory estimates on the sequence urban07 of the KAIST dataset (2 km, 9 min). **IMU** is especially accurate on smooth trajectories like in this apartment complex area.

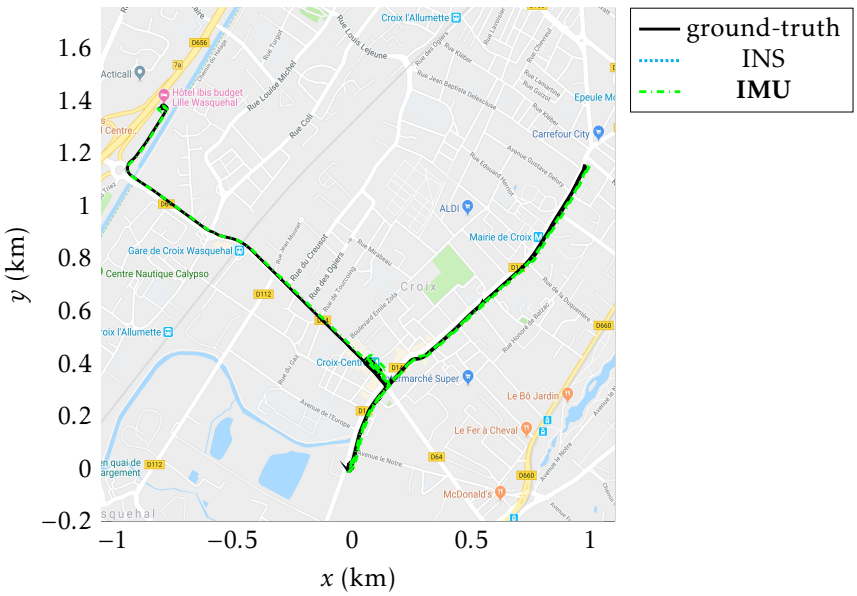


Figure 13.18: Trajectory estimates for the sequence 07 on the *Lille* dataset (6 km, 23 min). **IMU** keeps accurate after parking slot.

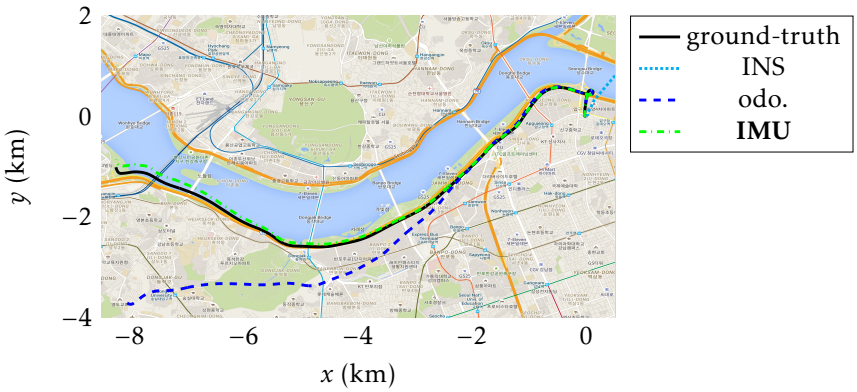


Figure 13.19: Trajectory results on the sequence urban31 of the KAIST dataset (11 km, 15 min). **IMU**' results are already impressive on this fast highway sequence, see the slight differences between the proposed methods and the ground-truth at the end of the trajectory.

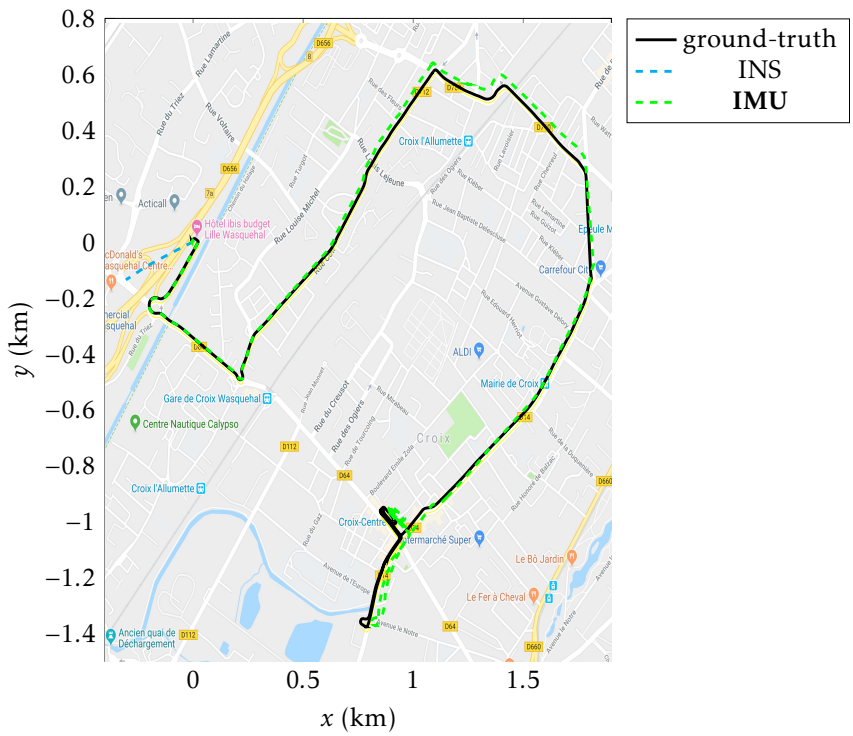


Figure 13.20: Trajectory estimates for the sequence 00 on the *Lille* dataset (6 km, 23 min). The proposed **IMU** follows ground-truth and keep accurate for the all trajectory whereas standard inertial integration (INS) diverges quickly.

dataset	ideal AMVD	ideal ARED	**IMU**
KAIST	0.943	0.986	0.993
Lille	0.715	0.950	**0.957**
Magny-Les-Hameaux	0.935	0.974	0.995
average score	0.907	0.977	0.988

Table 13.10: Results of various classifiers for detecting zero velocity z_n^{vel} in term of standardized partial Area Under Curve (AUC) with maximal false positive rate 0.1. More the AUC is, better is the detector. The proposed detectors outperforms their conventional counterparts. All methods excepted used only IMU signal.

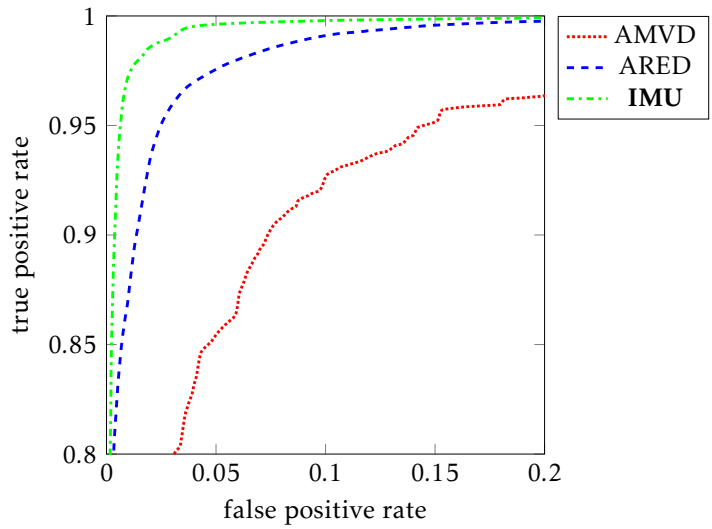


Figure 13.23: Detector results in term of ROC curve for classifying non zero-motion ($z_n^{\text{vel}} = 1$) from zero-motion ($z_n^{\text{vel}} = 0$). More the curve approaches the point $(0, 1)$, better is the classifier. The proposed detectors outperform the conventional classifiers and provide high true positive rate at very low false positive rate.

dataset	detector	adapter	Kalman filter		total
			propagation	update	
KAIST	11 (14%)	12 (15%)	14 (18%)	41 (53%)	78
Lille	11 (14%)	12 (15%)	13 (17%)	42 (54%)	78
MLH	11 (14%)	12 (16%)	15 (19%)	39 (52%)	77
average	11 (14%)	12 (16%)	12 (18%)	14 (52%)	77

Table 13.11: Execution times of the approach averaged for 1 s of data, in ms and percentage of the total execution time of the approach. Time execution slightly depends on the ratio of the number of non zero-motion detections over the number of zero-motion detections. The approach is more than 10 times faster than real-time execution with Python implementation.

rithms [161], we provide in this section preliminary results toward the generalization of the method to a new vehicle, by training the method on a platform and then validating it on another platform.

We compare three zero-motion detectors:

ideal ARED: the Angular Rate Energy Detector computes a measure of the gyro variance over a sample window. If the variance measure falls below a threshold, the vehicle is assumed to be stationary;

ideal AMVD: the Acceleration-Moving Variance Detector computes a measure of the accelerometer variance over a sample window. If the variance measure falls below a threshold, the vehicle is assumed to be stationary;

IMU-only (ours): the proposed detector with IMU only, and trained with training sequences of the evaluated dataset.

We plot the partial Receiver Operating Characteristic (ROC) curve in Figure 13.23, which illustrates the ability of a classifier as its threshold is varying [227,228]. The curve displays the true positive rate (i.e. where $\hat{z}_n^{\text{vel}} = z_n^{\text{vel}} = 1$ occurs) against the false positive rate for a total of 10^9 evaluated points. The best possible detector would yield a point in the upper left corner of coordinate $(0,1)$, representing 100% sensitivity (no false negatives) and 100% specificity (no false positives). Figure 13.23 zooms at a specific region of the whole curve since detectors get no false negatives once a sufficiently high false negative rate is crossed.

We numerically compare the classifiers in term of Area Under Curve (AUC) in Table 13.10, which is the area under the ROC curve, where we adjust the maximal false positive rate to 0.1 for highlighting discrepancies between methods. It measures the performance of a classifier as representing the probability that the classifier ranks a randomly drawn positive example higher than a randomly drawn negative example, and is independent from a manually defined threshold. From these results, we observe:

- the conventional albeit ideal AMVD and ARED can not detect zero-motion without avoiding false detection;
- the proposed IMU-only detector identifies zero-motion with a particularly low level of false detections, a.k.a. false alarms. We set the detector with a threshold of 0.95 and thus respect the filter constraint for almost never encountering false-alarm.

3.2 3.2 Computational Execution Times

The computational execution times of the method is provided in Table 13.11, where we divide time statistics into datasets. We run the method on a Dell Precision Tower 7910 machine which accelerates the detector and adapter computations through a GTX 1080 GPU, and we establish time statistic with the Python profiler. The filter operates on the CPU since computing on GPU is slower in our case and generally unefficient for computations effected in double precision with many back-and-forth transfers of data.

We observe in Table 13.11 small variations of execution times across the datasets, which is due to different ratios of the number of non zero-motion detections over the number of zero-motion detections. Indeed, detecting a zero-motion modifies the propagation step of the filter and the dimension of the observation vector $\mathbf{y}_n$. The computational requirements are reasonable as the methods keeps 10 times faster than real execution times, and we finally note our Python implementation can be drastically

accelerated both for the filter and deep neural networks blocks with the recent development of the PyTorch C++ API[3].

4 4 Conclusion

This chapter extends the approach for inertial based navigation of ground vehicles. Our approach exploits deep neural networks to identify zero-motion information from raw sensor signals which is then leveraged in an invariant extended Kalman filter that performs localization, velocity and sensor bias estimation. We additionally design deep learning based blocks that feed to the filter measurements along with time-varying measurement uncertainties which alleviate the practitioner for cumbersome parameter tuning. The entire pipeline requires only IMU data, and can incorporate GPS, wheel encoders and altimeter. The method leads to surprisingly accurate results on four datasets, and open new perspectives for combination of data-driven methods with well established methods for autonomous systems. Future works are an ablation study to precisely asses the advantages and limitations of each pseudo-measurement we use.

[3] https://pytorch.org/cppdocs/

CHAPTER 14

Conclusion of the Thesis

This thesis investigates various challenging aspects regarding Kalman filtering for localization and navigation for vehicles equipped with sensors such as cameras and inertial measurement units. It brings several contributions divided in two categories.

The first category of contributions is for enhancing and applying Kalman filter theory. It consists in building on the recent IEKF theory to address the inconsistency of EKF for SLAM, navigation with vision sensors, and the more general question of Kalman filtering on manifolds. The thesis starts by revisiting the theoretical problem of EKF inconsistency for SLAM, and shows how the use of the IEKF may resolve those issues. It is then studied and shown how the well established unscented Kalman filter may be adapted in the case where the state does not belong to a vector space, but to a more complicated space, namely a manifold. The proposed method applies to all Lie groups, and in particular allows to propose an unscented version of the IEKF. This allows fast prototyping, in the sense that it spares the sometimes difficult computation of Jacobians that must be performed in the extended Kalman filter methodology. The practical problem of SLAM and odometry in the presence of vision and inertial sensors is then addressed. The thesis builds on the state-of-the art multi-state constrained Kalman filter to attack the problem, and the contribution essentially consists in building an invariant version of this filter, which then comes with consistency properties, and to propose a computationally efficient unscented version.

The second category of contributions of the thesis explores how recent tools from the field of deep learning can be used to improve Kalman filters. The thesis notably focuses on the assessment of uncertainty of sensors or pseudo-measurements, and designs an approach to automatically relate sensors' measurements to the state, and to tune the filter, that is, using machine learning techniques to find a dynamical tuning strategy of the Kalman filter parameters that best matches the data. As a result, a machine learning algorithm may learn the extent of uncertainty that lies in the formulated assumptions of zero lateral and vertical velocities for inertial navigation of ground vehicles, where the proposed method based only on inertial sensors competes with state-of-the art methods that use inertial sensors plus vision. Another type of uncertainty may stem from the nature of the observation, as pose estimates from laser scanners and the ICP algorithm. As it is important for the filter that estimates the robot's trajectory to "know" the amount of uncertainty present in the ICP's estimated displacement, the thesis proposes a simple theoretical approach based on statistics and an algorithm for this problem which goes beyond existing solutions, and overcomes their main drawbacks. The thesis then wholly focuses on the use of deep learning as

a tool to extract pertinent information from inertial measurements, and to use this information to navigate. In all cases, the method is supervised, and some ground truth information is necessary for the algorithms to learn. The thesis builds upon the zero velocity update which is a way to correct the estimates when a wheeled vehicle stops, and leverages deep learning to make the estimator system detect stops based only on the inertial sensors, hence relaxing the need for additional sensors besides inertial sensors. Evaluations on a publicly available car dataset demonstrates indeed that the proposed scheme may achieve a final precision of 20 m for a 21 km long trajectory of a vehicle driving for over an hour, equipped with an IMU of moderate precision. The thesis then focuses on the estimation of the orientation of the vehicle only, namely its attitude, and leverage deep learning to refine the modelling of the measurements emanating from the inertial sensors, and to calibrate them.

Along the thesis, several questions about filtering on manifolds and deep learning for Kalman filtering were raised. First, theoretical questions arise regarding filtering on manifold. The thesis has followed in focusing on parallelizable manifolds where a global coordinate system of tangent spaces exists, and readily provides a transport operation over the manifold. However, there are multiple choices for the parallel transport operation, e.g. Levi-Civita connection for parallel transport, which depends on the chosen metric.

Second this thesis addresses filtering methods for SLAM and visual-inertial odometry. While current systems provide accurate motion tracking in small-scale "friendly" environments, they are not robust enough for long-term, large-scale and safety-critical deployments for e.g. autonomous driving. More than accurate, these methods should be robust. These algorithms are thus subject to both theoretical and practical challenges: how a VIO system equipped with biased IMU is able to estimate scale, and to correctly perform state initialization are unclear, especially for degenerated motions. Other sources of inconsistency may come from the visual feature tracking which is subject to outliers. The tuning of MSCKF measurement update is a computer vision and filtering challenging question. The strategy to choose which camera pose to marginalize has a large impact on performances, if well done, one could expect accuracy with low computational requirement, but this problem has hardly been addressed, see [229].

Regarding deep learning, the thesis opens new perspectives for combination of data-driven methods with well established methods for autonomous systems, and opens up for exciting avenues. As a MSCKF is complex, using deep learning to improve it in a coupled-manner is debatable and we suggest to first attack specific challenge as the marginalization strategy mentionned above. Secondly, we managed to train neural networks by using accurate datasets ([106,211] have less than centimeter accuracy). If one searches to enhance a yet accurate system or sensors, it seems natural to first dispose of a sufficiently accurate ground-truth. A typical solution would consist in first computing a time expansive solution based on smoothing or optimization algorithms with various sensors. An other way is to leverage unsupervised learning, see e.g. [208] for visual-inertial odometry. Finally, the question of transfer learning, i.e. how using a neural network trained on one device for another device, which is decisive for industrial application, is left for future research works. These questions, along with the crucial issue of dealing with safety, computational and economical trade-offs, are fundamental if one wants to provide autonomous systems with reliable information in an industrial context.

Bibliography

[1] S. Julier and J. Uhlmann, "A New Extension of the Kalman Filter to Nonlinear Systems," *AeroSense*, pp. 182–193, 1997.

[2] S. Julier and J. Uhlmann, "Unscented Filtering and Nonlinear Estimation," *Proceedings of the IEEE*, vol. 92, no. 3, pp. 401–422, 2004.

[3] R. Van Der Merwe, *Sigma-Point Kalman Filters for Probabilistic Inference in Dynamic State-Space Models*. PhD thesis, The faculty of the OGI School of Science & Engineering at Oregon Health & Science University, 2004.

[4] T. Lefebvre, H. Bruyninckx, and J. De Schuller, "Comment on "A New Method for the Nonlinear Transformation of Means and Covariances in Filters and Rstimators"," *IEEE Transactions on Automatic Control*, vol. 47, no. 8, pp. 1406–1409, 2002.

[5] G. P. Huang, A. I. Mourikis, and S. I. Roumeliotis, "A Quadratic-Complexity Observability-Constrained Unscented Kalman Filter for SLAM," *IEEE Transactions on Robotics*, vol. 29, no. 5, pp. 1226–1243, 2013.

[6] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization Algorithms on Matrix Manifolds*. Princeton Univ. Press, 2009.

[7] G. Chirikjian, *Stochastic Models, Information Theory, and Lie Groups, Volume 1*. Birkhäuser, 2009.

[8] P. Kotaru and K. Sreenath, "Variation Based Extended Kalman Filter on S2," in *European Control Conference*, pp. 875–882, IEEE, 2019.

[9] S. Bonnabel, "Symmetries in Observer Design: Review of Some Recent Results and Applications to EKF-based SLAM," in *Robot Motion and Control*, pp. 3–15, Springer, 2012.

[10] A. Barrau and S. Bonnabel, "An EKF-SLAM Algorithm with Consistency Properties," *arXiv*, 2015.

[11] T. Barfoot, *State Estimation for Robotics*. Cambridge: Cambridge University Press, 2017.

[12] G. Chirikjian, *Stochastic Models, Information Theory, and Lie Groups, Volume 2*. Birkhäuser, 2012.

[13] S. Bonnabel, *Observateurs asymptotiques invariants: théorie et exemples*. PhD thesis, Mines ParisTech, 2007.

[14] N. Aghannan and P. Rouchon, "On Invariant Asymptotic Observers," in *Conference on Decision and Control (CDC)*, vol. 2, pp. 1479–1484, IEEE, 2002.

[15] S. Bonnabel, P. Martin, and P. Rouchon, "Symmetry-Preserving Observers," *IEEE Transactions on Automatic Control*, vol. 53, no. 11, pp. 2514–2526, 2008.

[16] S. Bonnabel, P. Martin, and E. Salaün, "Invariant Extended Kalman Filter: Theory and Application to a Velocity-Aided Attitude Estimation Problem," in *Conference On Decision and Control*, pp. 1297–1304, IEEE, 2009.

[17] R. Mahony, T. Hamel, and J.-M. Pflimlin, "Nonlinear Complementary Filters on the Special Orthogonal Group," *IEEE Transactions on Automatic Control*, vol. 53, no. 5, pp. 1203–1218, 2008.

[18] G. Bourmaud, R. Mégret, M. Arnaudon, and A. Giremus, "Continuous-Discrete Extended Kalman Filter on Matrix Lie Groups Using Concentrated Gaussian Distributions," *Journal of Mathematical Imaging and Vision*, vol. 51, no. 1, pp. 209–228, 2015.

[19] J. Bohn and A. K. Sanyal, "Unscented State Estimation for Rigid Body Motion on SE(3)," in *Conference on Decision and Control (CDC)*, pp. 7498–7503, IEEE, 2012.

[20] P. Batista, C. Silvestre, and P. Oliveira, "A GES Attitude Observer with Single Vector Observations," *Automatica*, vol. 48, no. 2, pp. 388–395, 2012.

[21] A. Khosravian, J. Trumpf, R. Mahony, and C. Lageman, "Observers for Invariant Systems on Lie Groups with Biased Input Measurements and Homogeneous Outputs," *Automatica*, vol. 55, pp. 19–26, 2015.

[22] T. Lee, "Global Unscented Attitude Estimation via the Matrix Fisher Distributions on SO(3)," in *American Control Conference*, (1016), pp. 4942–4947, IEEE, 2016.

[23] D. E. Zlotnik and J. R. Forbes, "Nonlinear Estimator Design on the Special Orthogonal Group Using Vector Measurements Directly," *IEEE Transactions on Automatic Control*, vol. 62, no. 1, pp. 149–160, 2017.

[24] A. Walsh, J. Arsenault, and J. R. Forbes, "Invariant Sliding Window Filtering for Attitude and Bias Estimation," in *American Control Conference*, pp. 3161–3166, IEEE, 2019.

[25] G. G. Scandaroli, P. Morin, and G. Silveira, "A Nonlinear Observer Approach for Concurrent Estimation of Pose, IMU bias and Camera-to-IMU Rotation," in *International Conference on Intelligent Robots and Systems (IROS)*, pp. 3335–3341, IEEE/RSJ, 2011.

[26] E. Malis, T. Hamel, R. Mahony, and P. Morin, "Estimation of Homography Dynamics on the Special Linear Group," in *Visual Servoing via Advanced Numerical Methods* (M. Morari, M. Thoma, G. Chesi, and K. Hashimoto, eds.), vol. 401, pp. 133–150, Springer, 2010.

[27] D. E. Zlotnik and J. R. Forbes, "Higher Order Nonlinear Complementary Filtering on Lie Groups," *IEEE Transactions on Automatic Control*, vol. 64, no. 5, pp. 1772–1783, 2019.

[28] H. A. Hashim, L. J. Brown, and K. McIsaac, "Nonlinear Stochastic Attitude Filters on the Special Orthogonal Group 3: Ito and Stratonovich," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 9, pp. 1853–1865, 2019.

[29] S. Berkane, A. Abdessameud, and A. Tayebi, "Hybrid Attitude and Gyro-Bias Observer Design on SO(3)," *IEEE Transactions on Automatic Control*, vol. 62, no. 11, pp. 6044–6050, 2017.

[30] S. Bonnabel, P. Martin, and P. Rouchon, "A Non-Linear Symmetry-Preserving Observer for Velocity-Aided Inertial Navigation," in *American Control Conference*, pp. 2910–2914, IEEE, 2006.

[31] M. Barczyk, S. Bonnabel, J. Deschaud, and F. Goulette, "Invariant EKF Design for Scan Matching-Aided Localization," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 6, pp. 2440–2448, 2015.

[32] C. Lageman, J. Trumpf, and R. Mahony, "Gradient-Like Observers for Invariant Dynamics on a Lie Group," *IEEE Transactions on Automatic Control*, vol. 55, no. 2, pp. 367–377, 2010.

[33] S. Diemer and S. Bonnabel, "An invariant Linear Quadratic Gaussian Controller for a Simplified Car," in *International Conference on Robotics and Automation (ICRA)*, pp. 448–453, IEEE, 2015.

[34] S. Bonnabel, "Left-Invariant Extended Kalman Filter and Attitude Estimation," in *Conference on Decision and Control (CDC)*, pp. 1027–1032, IEEE, 2007.

[35] J.-P. Condomines, C. Seren, and G. Hattenberger, "Nonlinear State Estimation Using an Invariant Unscented Kalman Filter," in *AIAA Guidance, Navigation, and Control*, 2013.

[36] A. Barrau and S. Bonnabel, "Three examples of the stability properties of the invariant extended Kalman filter," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 431–437, 2017.

[37] A. Barrau, *Non-Linear State Error Based Extended Kalman Filters with Applications to Navigation*. PhD thesis, Mines ParisTech, 2015.

[38] M.-A. Lavoie, J. Arsenault, and J. R. Forbes, "An Invariant Extended H infini Filter," in *Conference on Decision and Control (CDC)*, pp. 7905–7910, IEEE, 2019.

[39] P. Chauchat, A. Barrau, and S. Bonnabel, "Invariant Smoothing on Lie Groups," in *International Conference on Intelligent Robots and Systems (IROS)*, IEEE/RSJ, 2018.

[40] P. Chauchat, A. Barrau, and S. Bonnabel, "Kalman Filtering with a Class of Geometric State Equality Constraints," in *Conference on Decision and Control (CDC)*, pp. 2581–2586, IEEE, 2017.

[41] J. L. Crassidis, "Unscented Filtering for Spacecraft Attitude Estimation," *Journal of guidance, control, and dynamics*, vol. 26, no. 4, pp. 536–542, 2003.

[42] A. Barrau and S. Bonnabel, "The Invariant Extended Kalman Filter as a Stable Observer," *IEEE Transactions on Automatic Control*, vol. 62, no. 4, pp. 1797–1812, 2017.

[43] A. Barrau and S. Bonnabel, "Invariant Kalman Filtering," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, no. 1, pp. 237–257, 2018.

[44] A. Barrau and S. Bonnabel, "Linear Observed Systems on Groups," *Systems & Control Letters*, vol. 129, pp. 36–42, 2019.

[45] G. P. Huang, A. I. Mourikis, and S. I. Roumeliotis, "Observability-Based Rules for Designing Consistent EKF SLAM Estimators," *The International Journal of Robotics Research*, vol. 29, no. 5, pp. 502–528, 2010.

[46] M. Li and A. Mourikis, "High-Precision, Consistent EKF-Based Visual-Inertial Odometry," *The International Journal of Robotics Research*, vol. 32, no. 6, pp. 690–711, 2013.

[47] J. A. Hesch, D. G. Kottas, S. L. Bowman, and S. I. Roumeliotis, "Camera-IMU-based Localization: Observability Analysis and Consistency Improvement," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 182–201, 2014.

[48] Y. Bar-Shalom, X.-R. Li, and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation*. John Wiley & Sons, Inc., 2001.

[49] S. Julier and J. Uhlmann, "A Counter Example to the Theory of Simultaneous Localization and Map Building," in *International Conference on Robotics and Automation (ICRA)*, vol. 4, pp. 4238–4243, IEEE, 2001.

[50] J. A. Castellanos, J. Neira, and J. Tardós, "Limits to the Consistency of EKF-based SLAM," *IFAC Proceedings Volumes*, vol. 37, no. 8, pp. 716–721, 2004.

[51] T. Bailey, J. Nieto, J. Guivant, M. Stevens, and E. Nebot, "Consistency of the EKF-SLAM Algorithm," in *International Conference on Intelligent Robots and Systems (IROS)*, pp. 3562–3568, IEEE/RSJ, 2006.

[52] G. Huang, "Towards Consistent Filtering for Discrete-Time Partially-Observable Nonlinear Systems," *Systems & Control Letters*, vol. 106, pp. 87–95, 2017.

[53] G. P. Huang, A. I. Mourikis, and S. I. Roumeliotis, "Analysis and Improvement of the Consistency of Extended Kalman Filter Based SLAM," in *International Conference on Robotics and Automation (ICRA)*, pp. 473–479, IEEE, 2008.

[54] G. Huang, M. Kaess, and J. J. Leonard, "Towards Consistent Visual-Inertial Navigation," in *International Conference on Robotics and Automation (ICRA)*, pp. 4926–4933, IEEE, 2014.

[55] G. Huang, K. Eckenhoff, and J. Leonard, "Optimal-State-Constraint EKF for Visual-Inertial Navigation," in *Robotics Research*, Springer, pp. 125–139, Springer, Cham, 2018.

[56] G. P. Huang, N. Trawny, A. I. Mourikis, and S. I. Roumeliotis, "Observability-Based Consistent EKF Estimators for Multi-Robot Cooperative Localization," *Autonomous Robots*, vol. 30, no. 1, pp. 99–122, 2011.

[57] P. Olver, *Classical Invariant Theory*, vol. 44. Cambridge University Press, 1999.

[58] T. Zhang, K. Wu, J. Song, S. Huang, and G. Dissanayake, "Convergence and Consistency Analysis for A 3D Invariant-EKF SLAM," *IEEE Robotics and Automation Letters (RA-L)*, vol. 2, no. 2, pp. 733–740, 2017.

[59] K. Wu, T. Zhang, D. Su, S. Huang, and G. Dissanayake, "An Invariant-EKF VINS Algorithm for Improving Consistency," in *International Conference on Intelligent Robots and Systems (IROS)*, pp. 1578–1585, IEEE/RSJ, 2017.

[60] M. Brossard, S. Bonnabel, and J.-P. Condomines, "Unscented Kalman filtering on Lie groups," in *International Conference on Intelligent Robots and Systems (IROS)*, pp. 2485–2491, IEEE/RSJ, 2017.

[61] D. Caruso, *Improving Visual-Inertial Navigation Using Stationary Environmental Magnetic Disturbances*. PhD thesis, Paris-Sacly, 2018.

[62] S. Heo and C. G. Park, "Consistent EKF-Based Visual-Inertial Odometry on Matrix Lie Group," *IEEE Sensors*, vol. 18, no. 9, pp. 3780–3788, 2018.

[63] R. Mahony and T. Hamel, "A Geometric Nonlinear Observer for Simultaneous Localisation and Mapping," in *Conference on Decision and Control (CDC)*, pp. 2408–2415, IEEE, 2017.

[64] J. A. Castellanos, R. Martinez-Cantin, J. Tardós, and J. Neira, "Robocentric Map Joining: Improving the Consistency of EKF-SLAM," *Robotics and Autonomous Systems*, vol. 55, no. 1, pp. 21–29, 2007.

[65] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, "iSAM2: Incremental Smoothing and Mapping using the Bayes Tree," *The International Journal of Robotics Research*, vol. 31, no. 2, pp. 216–235, 2012.

[66] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G2o: A General Framework for Graph Optimization," in *International Conference on Robotics and Automation (ICRA)*, pp. 3607–3613, IEEE, 2011.

[67] Z. Chen, K. Jiang, and J. Hung, "Local Observability Matrix and its Application to Observability Analyses," in *Annual Conference of IEEE Industrial Electronics Society*, pp. 100–103, IEEE, 1990.

[68] B. Guerreiro, P. Batista, C. Silvestre, and P. Oliveira, "Globally Asymptotically Stable Sensor-Based Simultaneous Localization and Mapping," *IEEE Transactions on Robotics*, vol. 29, no. 6, pp. 1380–1395, 2013.

[69] P. Lourenço, B. Guerreiro, P. Batista, P. Oliveira, and C. Silvestre, "Simultaneous Localization and Mapping for Aerial Vehicles: A 3-D Sensor-Based GAS Filter," *Autonomous Robots*, vol. 40, pp. 881–902, June 2016.

[70] K. Leung, T. Barfoot, and H. Liu, "Decentralized Cooperative SLAM for Sparsely-Communicating Robot Networks: A Centralized-Equivalent Approach," *Journal of Intelligent & Robotic Systems*, vol. 66, no. 3, pp. 321–342, 2012.

[71] S. Huang, Z. Wang, and G. Dissanayake, "Sparse Local Submap Joining Filter for Building Large-Scale Maps," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 1121–1130, 2008.

[72] E. Leffens, F. Markley, and M. Shuster, "Kalman Filtering for Spacecraft Attitude Estimation," *Journal of Guidance, Control, and Dynamics*, vol. 5, no. 5, pp. 417–429, 1982.

[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*, vol. 14, no. 1, pp. 57–77, 2013.

[74] J. R. Forbes, A. H. de Ruiter, and D. E. Zlotnik, "Continuous-Time Norm-Constrained Kalman Filtering," *Automatica*, vol. 50, no. 10, pp. 2546–2554, 2014.

[75] J. Solà, "Quaternion Kinematics for the Error-State Kalman Filter," *arXiv*, p. 94, 2012.

[76] E. Kraft, "A Quaternion-Based Unscented Kalman Filter for Orientation Tracking," in *International Conference of Information Fusion*, pp. 47–54, IEEE, 2003.

[77] A. Barrau and S. Bonnabel, "Intrinsic Filtering on Lie Groups With Applications to Attitude Estimation," *IEEE Transactions on Automatic Control*, vol. 60, no. 2, pp. 436–449, 2015.

[78] G. Bourmaud, R. Mégret, A. Giremus, and Y. Berthoumieu, "Discrete Extended Kalman Filter on Lie Groups," in *European Signal Processing Conference (EUSIPCO)*, pp. 1–5, IEEE, 2013.

[79] J. J. Bohn, A. K. Sanyal, and E. A. Butcher, "Unscented State Estimation for Rigid Body Attitude Motion with a Finite-Time Stable Observer," in *Conference on Decision and Control (CDC)*, pp. 4698–4703, IEEE, 2016.

[80] G. Loianno, M. Watterson, and V. Kumar, "Visual Inertial Odometry for Quadrotors on SE(3)," in *International Conference onRobotics and Automation (ICRA)*, pp. 1544–1551, IEEE, 2016.

[81] J. R. Forbes and D. E. Zlotnik, "Sigma Point Kalman Filtering on Matrix Lie Groups Applied to the SLAM Problem," in *Geometric Science of Information*, pp. 318–328, Springer, 2017.

[82] J. Svacha, G. Loianno, and V. Kumar, "Inertial Yaw-Independent Velocity and Attitude Estimation for High-Speed Quadrotor Flight," *IEEE Robotics and Automation Letters (RA-L)*, vol. 4, no. 2, pp. 1109–1116, 2019.

[83] J. Solà, J. Deray, and D. Atchuthan, "A Micro Lie theory for State Estimation in Robotics," *arXiv*, 2018.

[84] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-Manifold Preintegration for Real-Time Visual-Inertial Odometry," *IEEE Transactions on Robotics*, vol. 33, no. 1, pp. 1–21, 2017.

[85] A. Barrau and S. Bonnabel, "Aligment Method for an Inertial Unit," 2016.

[86] Y. Wang and G. Chirikjian, "Error Propagation on the Euclidean Group with Applications to Manipulator Kinematics," *IEEE Transactions on Robotics*, vol. 22, no. 4, pp. 591–602, 2006.

[87] W. Park, Y. Liu, Y. Zhou, M. Moses, and G. Chirikjian, "Kinematic State Estimation and Motion Planning for Dtochastic Nonholonomic Systems using the Exponential Map," *Robotica*, vol. 26, no. 4, pp. 419–434, 2008.

[88] G. Chirikjian and M. Kobilarov, "Gaussian Approximation of Non-Linear Measurement Models on Lie Groups," in *Conference on Decision and Control (CDC)*, pp. 6401–6406, IEEE, 2014.

[89] T. Barfoot, J. Forbes, and P. Furgale, "Pose Estimation using Linearized Rotations and Quaternion Algebra," *Acta Astronautica*, vol. 68, no. 1-2, pp. 101–112, 2011.

[90] T. Barfoot and P. Furgale, "Associating Uncertainty With Three-Dimensional Poses for Use in Estimation Problems," *IEEE Transactions on Robotics*, vol. 30, no. 3, pp. 679–693, 2014.

[91] R. Hartley, M. G. Jadidi, J. W. Grizzle, and R. M. Eustice, "Contact-Aided Invariant Extended Kalman Filtering for Legged Robot State Estimation," in *Robotics Science and Systems*, 2018.

[92] N. Ko, W. Youn, I. Choi, G. Song, and T. Kim, "Features of Invariant Extended Kalman Filter Applied to Unmanned Aerial Vehicle Navigation," *Sensors*, vol. 18, no. 9, p. 2855, 2018.

[93] N. Y. Ko, G. Song, W. Youn, I. H. Choi, and T. S. Kim, "Improvement of Extended Kalman Filter Using Invariant Extended Kalman Filter," in *International Conference on Control, Automation and Systems*, pp. 948–950, 2018.

[94] M. Wang and A. Tayebi, "A Globally Exponentially Stable Nonlinear Hybrid Observer for 3D Inertial Navigation," in *Conference on Decision and Control (CDC)*, pp. 1367–1372, IEEE, 2018.

[95] M. Brossard, S. Bonnabel, and A. Barrau, "Unscented Kalman Filter on Lie Groups for Visual Inertial Odometry," in *International Conference on Intelligent Robots and Systems (IROS)*, pp. 649–655, IEEE/RSJ, 2018.

[96] M. Brossard, A. Barrau, and S. Bonnabel, "Exploiting Symmetries to Design EKFs with Consistency Properties for Navigation and SLAM," *Sensors*, p. 8, 2019.

[97] S. Heo, J. H. Jung, and C. G. Park, "Consistent EKF-Based Visual-Inertial Navigation Using Points and Lines," *IEEE Sensors*, vol. 18, no. 18, pp. 7638–7649, 2018.

[98] S. Hauberg, F. Lauze, and K. S. Pedersen, "Unscented Kalman Filtering on Riemannian Manifolds," *Journal of Mathematical Imaging and Vision*, vol. 46, no. 1, pp. 103–120, 2013.

[99] J.-P. Condomines, C. Seren, and G. Hattenberger, "Pi-Invariant Unscented Kalman Filter for Sensor Fusion," in *Conference on Decision and Control (CDC)*, pp. 1035–1040, IEEE, 2014.

[100] M. Wang and A. Tayebi, "Geometric Nonlinear Observer Design for SLAM on a Matrix Lie Group," in *Conference on Decision and Control (CDC)*, pp. 1488–1493, IEEE, 2018.

[101] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision Meets Robotics: The KITTI Dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[102] B. Nisar, P. Foehn, D. Falanga, and D. Scaramuzza, "VIMO: Simultaneous Visual Inertial Model-Based Odometry and Force Estimation," *IEEE Robotics and Automation Letters (RA-L)*, vol. 4, no. 3, pp. 2785–2792, 2019.

[103] M. Zefran, V. Kumar, and C. Croke, "Metrics and Connections for Rigid-Body Kinematics," *The International Journal of Robotics Research*, vol. 18, no. 2, pp. 243–258, 1999.

[104] J. Milnor, "Analytic Proofs of the Hairy Ball Theorem and the Brouwer Fixed Point Theorem," *The American Mathematical Monthly*, vol. 87, no. 7, pp. 521–524, 1978.

[105] X. Pennec, "Intrinsic Statistics on Riemannian Manifolds: Basic Tools for Geometric Measurements," *Journal of Mathematical Imaging and Vision*, vol. 25, no. 1, pp. 127–154, 2006.

[106] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The EuRoC Micro Aerial Vehicle Datasets," *The International Journal of Robotics Research*, vol. 35, no. 10, pp. 1157–1163, 2016.

[107] J. A. Hesch, D. G. Kottas, S. L. Bowman, and S. I. Roumeliotis, "Observability-Constrained Vision-aided Inertial Navigation," *University of Minnesota, Dept. of Comp. Sci. & Eng., MARS Lab, Tech. Rep*, vol. 1, 2012.

[108] K. Sun, K. Mohta, B. Pfrommer, M. Watterson, S. Liu, Y. Mulgaonkar, C. J. Taylor, and V. Kumar, "Robust Stereo Visual Inertial Odometry for Fast Autonomous Flight," *IEEE Robotics and Automation Letters (RA-L)*, vol. 3, no. 2, pp. 965–972, 2018.

[109] A. Mourikis and S. Roumeliotis, "A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation," in *International Conference on Robotics and Automation (ICRA)*, pp. 3565–3572, IEEE, 2007.

[110] J. Kelly and G. S. Sukhatme, "Visual-Inertial Simultaneous Localization, Mapping and Sensor-to-Sensor Self-Calibration," in *International Symposium on Computational Intelligence in Robotics and Automation*, pp. 360–368, IEEE, 2009.

[111] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-Based Visual–Inertial Odometry using Nonlinear Optimization," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2015.

[112] T. Qin, P. Li, and S. Shen, "VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[113] D. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*. Pearson, 2012. OCLC: ocn751787157.

[114] R. Van der Merwe and E. Wan, "The Square-Root Unscented Kalman Filter for State and Parameter Estimation," in *International Conference on Acoustics, Speech, and Signal Processing. Proceedings*, vol. 6, pp. 3461–3464, IEEE, 2001.

[115] S. Holmes, G. Klein, and D. Murray, "An O(N) Square Root Unscented Kalman Filter for Visual Simultaneous Localization and Mapping," *Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 7, pp. 1251–1263, 2009.

[116] J. Shi and C. Tomasi, "Good Features to Track," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 593–600, IEEE, 1994.

[117] S. J. Julier, "The Scaled Unscented Transformation," in *American Control Conference*, vol. 6, pp. 4555–4559, IEEE, 2002.

[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*, vol. 25, no. 2, pp. 264–280, 2009.

[119] P. Abbeel, A. Coates, M. Montemerlo, A. Ng, and S. Thrun, "Discriminative Training of Kalman Filters," in *Robotics: Science and Systems*, vol. 2, 2005.

[120] R. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Journal of Basic Engineering*, vol. 82, no. D, pp. 35–45, 1960.

[121] R. Shumway and D. Stoffer, "An Approach To Time Series Smoothing And Forecasting Using The Em Algorithm," *Wiley Blackwell*, vol. 3, no. 4, pp. 253–264, 1982.

[122] T. Powell, "Automated Tuning of an Extended Kalman Filter Using the Downhill Simplex Algorithm," *Journal of Guidance, Control, and Dynamics*, vol. 25, no. 5, pp. 901–908, 2002.

[123] Y. Oshman and I. Shaviv, "Optimal tuning of a Kalman filter using genetic algorithms," in *AIAA Guidance, Navigation, and Control*, 2000.

[124] Z. Chen, C. Heckman, S. Julier, and N. Ahmed, "Weak in the NEES?: Auto-Tuning Kalman Filters with Bayesian Optimization," in *International Conference on Information Fusion*, 2018.

[125] S. Barratt and S. Boyd, "Fitting a Kalman Smoother to Data," *arXiv*, 2019.

[126] F. Castella, "An Adaptive Two-Dimensional Kalman Tracking Filter," *Transactions on Aerospace and Electronic Systems*, vol. AES-16, no. 6, pp. 822–829, 1980.

[127] G. Bresson, Z. Alsayed, L. Yu, and S. Glaser, "Simultaneous Localization and Mapping: A Survey of Current Trends in Autonomous Driving," *IEEE Transactions on Intelligent Vehicles*, vol. 2, no. 3, pp. 194–220, 2017.

[128] OxTS, "Why it is Necessary to Integrate an Inertial Measurement Unit with Imaging Systems on an Autonomous Vehicle." https://www.oxts.com/technical-notes/why-use-ins-with-autonomous-vehicle/, 2018.

[129] F. Zheng, H. Tang, and Y.-H. Liu, "Odometry Vision Based Ground Vehicle Motion Estimation With SE(2)-Constrained SE(3) Poses," *IEEE Transactions on Cybernetics*, pp. 1–12, 2018.

[130] F. Zheng and Y.-H. Liu, "SE(2)-Constrained Visual Inertial Fusion for Ground Vehicles," *IEEE Sensors*, vol. 18, no. 23, pp. 9699–9707, 2018.

[131] M. Buczko, V. Willert, J. Schwehr, and J. Adamy, "Self-Validation for Automotive Visual Odometry," in *Intelligent Vehicles Symposium*, pp. 1–6, IEEE, 2018.

[132] G. Dissanayake, S. Sukkarieh, E. Nebot, and H. Durrant-Whyte, "The Aiding of a Low-cost Strapdown Inertial Measurement Unit Using Vehicle Model Constraints for Land Vehicle Applications," *Transactions on Robotics and Automation*, vol. 17, no. 5, pp. 731–747, 2001.

[133] D. Simon, "Kalman Filtering with State Constraints: A Survey of Linear and Nonlinear Algorithms," *Control Theory & Applications*, vol. 4, no. 8, pp. 1303–1318, 2010.

[134] C. Kilic, J. N. Gross, N. Ohi, R. Watson, J. Strader, T. Swiger, S. Harper, and Y. Gu, "Improved Planetary Rover Inertial Navigation and Wheel Odometry Performance through Periodic Use of Zero-Type Constraints," in *International Conference on Intelligent Robots and Systems (IROS)*, IEEE/RSJ, 2019.

[135] J.-E. Deschaud, "IMLS-SLAM: Scan-to-Model Matching Based on 3D Data," in *International Conference on Robotics and Automation (ICRA)*, pp. 2480–2485, IEEE, 2018.

[136] R. Mur-Artal and J. Tardos, "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[137] F. Tschopp, T. Schneider, A. W. Palmer, N. Nourani-Vatani, C. Cadena Lerma, R. Siegwart, and J. Nieto, "Experimental Comparison of Visual-Aided Odometry Methods for Rail Vehicles," *IEEE Robotics and Automation Letters (RA-L)*, vol. 4, no. 2, pp. 1815–1822, 2019.

[138] M. Mousa, K. Sharma, and C. Claudel, "Inertial Measurement Units-Based Probe Vehicles: Automatic Calibration, Trajectory Estimation, and Context Detection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 10, pp. 3133–3143, 2018.

[139] J. Wahlstrom, I. Skog, J. Rodrigues, P. Handel, and A. Aguiar, "Map-Aided Dead-Reckoning Using Only Measurements of Speed," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 3, pp. 244–253, 2016.

[140] A. Mahmoud, A. Noureldin, and H. S. Hassanein, "Integrated Positioning for Connected Vehicles," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–13, 2019.

[141] R. Karlsson and F. Gustafsson, "The Future of Automotive Localization Algorithms: Available, Reliable, and Scalable Localization: Anywhere and Anytime," *Signal Processing Magazine*, vol. 34, no. 2, pp. 60–69, 2017.

[142] K. Wu, C. Guo, G. Georgiou, and S. Roumeliotis, "VINS on Wheels," in *International Conference on Robotics and Automation (ICRA)*, pp. 5155–5162, IEEE, 2017.

[143] A. Brunker, T. Wohlgemuth, M. Frey, and F. Gauterin, "Odometry 2.0: A Slip-Adaptive EIF-Based Four-Wheel-Odometry Model for Parking," *IEEE Transactions on Intelligent Vehicles*, vol. 4, no. 1, pp. 114–126, 2019.

[144] M. Kok, J. D. Hol, and T. B. Schön, "Using Inertial Sensors for Position and Orientation Estimation," *Foundations and Trends® in Signal Processing*, vol. 11, no. 1-2, pp. 1–153, 2017.

[145] A. Ramanandan, A. Chen, and J. Farrell, "Inertial Navigation Aiding by Stationary Updates," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 1, pp. 235–248, 2012.

[146] M. Brossard, A. Barrau, and S. Bonnabel, "RINS-W: Robust Inertial Navigation System on Wheels," in *International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2019.

[147] F. Aghili and C.-Y. Su, "Robust Relative Navigation by Integration of ICP and Adaptive Kalman Filter Using Laser Scanner and IMU," *IEEE Transactions on Mechatronics*, vol. 21, no. 4, pp. 2015–2026, 2016.

[148] H. Yan, Q. Shan, and Y. Furukawa, "RIDI: Robust IMU Double Integration," in *European Conference on Computer Vision*, 2018.

[149] C. Chen, X. Lu, A. Markham, and N. Trigoni, "IONet: Learning to Cure the Curse of Drift in Inertial Odometry," in *AAAI*, 2018.

[150] R. Krishnan and U. Shalit, "Deep Kalman Filter," *arXiv*, 2015.

[151] R. Krishnan, U. Shalit, and D. Sontag, "Structured Inference Networks for Nonlinear State Space Models," *AAAI*, p. 9, 2017.

[152] C. Chen, C. X. Lu, B. Wang, N. Trigoni, and A. Markham, "DynaNet: Neural Kalman Dynamical Model for Motion Estimation and Prediction," *arXiv*, 2019.

[153] T. Haarnoja, A. Ajay, S. Levine, and P. Abbeel, "Backprop KF: Learning Discriminative Deterministic State Estimators," in *Advances in Neural Information Processing Systems*, 2016.

[154] S. Hosseinyalamdary, "Deep Kalman Filter: Simultaneous Multi-Sensor Integration and Modelling; A GNSS/IMU Case Study," *Sensors*, vol. 18, no. 5, p. 1316, 2018.

[155] K. Liu, K. Ok, W. Vega-Brown, and N. Roy, "Deep Inference for Covariance Estimation: Learning Gaussian Noise Models for State Estimation," in *International Conference on Robotics and Automation (ICRA)*, pp. 1436–1443, IEEE, 2018.

[156] M. Brossard and S. Bonnabel, "Learning Wheel Odometry and IMU Errors for Localization," in *International Conference on Robotics and Automation (ICRA)*, IEEE, 2019.

[157] B. Odelson, M. Rajamani, and J. Rawlings, "A New Autocovariance Least-Squares Method for Estimating Noise Covariances," *Automatica*, vol. 42, no. 2, pp. 303–308, 2006.

[158] A. Mohamed and K. Schwarz, "Adaptive Kalman Filtering for INS/GPS," *Journal of Geodesy*, vol. 73, no. 4, pp. 193–203, 1999.

[159] M. Susi, M. Andreotti, M. Aquino, and A. Dodson, "Tuning a Kalman Filter Carrier Tracking Algorithm in the Presence of Ionospheric Scintillation," *GPS Solutions*, vol. 21, no. 3, pp. 1149–1160, 2017.

[160] M. Tahk and J. Speyer, "Target Tracking Problems Subject to Kinematic Constraints," *IEEE Transactions on Automatic Control*, vol. 32, no. 2, pp. 324–326, 1990.

[161] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. The MIT press, 2016.

[162] D. Kingma and J. Ba, "ADAM: A Method for Stochastic Optimization," in *International Conference on Learning Representations*, 2014.

[163] G. Parisi, R. Kemker, J. Part, C. Kanan, and S. Wermter, "Continual Lifelong Learning with Neural Networks: A Review," *Neural Networks*, vol. 113, pp. 54–71, 2019.

[164] T. Qin, J. Pan, S. Cao, and S. Shen, "A General Optimization-based Framework for Local Odometry Estimation with Multiple Sensors," *ArXiv*, 2019.

[165] M. Ramezani and K. Khoshelham, "Vehicle Positioning in GNSS-Deprived Urban Areas by Stereo Visual-Inertial Odometry," *IEEE Transactions on Intelligent Vehicles*, vol. 3, no. 2, pp. 208–217, 2018.

[166] S. Heo, J. Cha, and C. G. Park, "EKF-Based Visual Inertial Navigation Using Sliding Window Nonlinear Optimization," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–10, 2018.

[167] N. Sünderhauf, O. Brock, W. Scheirer, R. Hadsell, D. Fox, J. Leitner, B. Upcroft, P. Abbeel, W. Burgard, M. Milford, and P. Corke, "The Limits and Potentials of Deep Learning for Robotics," *The International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 405–420, 2018.

[168] H. Thiébaux and F. Zwiers, "The Interpretation and Estimation of Effective Sample Size," 1984.

[169] A. Censi, "An Accurate Closed-form Estimate of ICP's Covariance," in *International Conference on Robotics and Automation (ICRA)*, pp. 3167–3172, IEEE, 2007.

[170] F. Pomerleau, F. Colas, and R. Siegwart, "A Review of Point Cloud Registration Algorithms for Mobile Robotics," *Foundations and Trends in Robotics*, vol. 4, no. 1, pp. 1–104, 2015.

[171] D. Holz, A. E. Ichim, F. Tombari, R. B. Rusu, and S. Behnke, "Registration with the Point Cloud Library: A Modular Framework for Aligning in 3-D," *Robotics & Automation Magazine*, vol. 22, no. 4, pp. 110–124, 2015.

[172] R. Dube, A. Gawel, H. Sommer, J. Nieto, R. Siegwart, and C. Cadena, "An Online Multi-robot SLAM System for 3D LiDARs," in *International Conference on Intelligent Robots and Systems (IROS)*, pp. 1004–1011, IEEE/RSJ, 2017.

[173] P. Geneva, K. Eckenhoff, Y. Yang, and G. Huang, "LIPS: LiDAR-Inertial 3D Plane SLAM," in *International Conference on Intelligent Robots and Systems (IROS)*, pp. 123–130, IEEE/RSJ, 2018.

[174] F. Pomerleau, A. Breitenmoser, M. Liu, F. Colas, and R. Siegwart, "Noise Characterization of Depth Sensors for Surface Inspections," in *International Conference on Applied Robotics for the Power Industry*, pp. 16–21, IEEE, 2012.

[175] D. Landry, F. Pomerleau, and P. Giguère, "CELLO-3D: Estimating the Covariance of ICP in the Real World," in *International Conference on Robotics and Automation (ICRA)*, 2019.

[176] F. Pomerleau, M. Liu, F. Colas, and R. Siegwart, "Challenging Data Sets for Point Cloud Registration Algorithms," *The International Journal of Robotics Research*, vol. 31, no. 14, pp. 1705–1711, 2012.

[177] F. Pomerleau, F. Colas, R. Siegwart, and S. Magnenat, "Comparing ICP Variants on Real-world Data Sets: Open-source Library and Experimental Protocol," *Auton. Robots*, vol. 34, no. 3, pp. 133–148, 2013.

[178] S. Pfister, K. Kriechbaum, S. Roumeliotis, and J. Burdick, "Weighted Range Sensor Matching Algorithms for Mobile Robot Displacement Estimation," in *International Conference on Robotics and Automation (ICRA)*, vol. 2, pp. 1667–1674, IEEE, 2002.

[179] M. Barczyk and S. Bonnabel, "Towards Realistic Covariance Estimation of ICP-based Kinect V1 Scan Matching: The 1D Case," in *American Control Conference*, pp. 4833–4838, IEEE, 2017.

[180] T. M. Iversen, A. G. Buch, and D. Kraft, "Prediction of ICP Pose Uncertainties Using Monte Carlo Simulation with Synthetic Depth Images," in *International Conference on Intelligent Robots and Systems (IROS)*, pp. 4640–4647, IEEE/RSJ, 2017.

[181] H. Yang, J. Shi, and L. Carlone, "TEASER: Fast and Certifiable Point Cloud Registration," *2001.07715*, 2020.

[182] Z. Wang, Y. Liu, Q. Liao, H. Ye, M. Liu, and L. Wang, "Characterization of a RS-LiDAR for 3D Perception," in *International Conference on Technology in Automation, Control, and Intelligent Systems*, 2018.

[183] J. Laconte, S.-P. Deschênes, M. Labussière, and F. Pomerleau, "LiDAR Measurement Bias Estimation via Return Waveform Modelling in a Context of 3D Mapping," in *International Conference on Robotics and Automation (ICRA)*, 2019.

[184] O. Bengtsson and A.-J. Baerveldt, "Robot Localization Based on Scan-Matching," *Robotics and Autonomous Systems*, vol. 44, no. 1, pp. 29–40, 2003.

[185] P. Biber and W. Strasser, "The Normal Distributions Transform: A New Approach to Laser Scan Matching," in *International Conference on Intelligent Robots and Systems (IROS)*, vol. 3, pp. 2743–2748, IEEE/RSJ, 2003.

[186] S. Bonnabel, M. Barczyk, and F. Goulette, "On the Covariance of ICP-Based Scan-Matching Techniques," in *American Control Conference*, pp. 5498–5503, IEEE, 2016.

[187] S. M. Prakhya, L. Bingbing, Y. Rui, and W. Lin, "A Closed-form Estimate of 3D ICP Covariance," in *International Conference on Machine Vision Applications*, pp. 526–529, IEEE, 2015.

[188] E. Mendes, P. Koch, and S. Lacroix, "ICP-based Pose-graph SLAM," in *International Symposium on Safety, Security, and Rescue Robotics*, pp. 195–200, IEEE, 2016.

[189] S. Julier, J. Uhlmann, and H. Durrant-Whyte, "A New Method for the Nonlinear Transformation of Means and Covariances in Filters and Estimators," *IEEE Transactions on Automatic Control*, vol. 45, no. 3, pp. 477–482, 2000.

[190] A. Long, K. Eckenhoff, M. Mashner, and G. Chirikjian, "The Banana Distribution is Gaussian: A Localization Study with Exponential Coordinates," *Robotics: Science and Systems*, 2012.

[191] F. Gustafsson and G. Hendeby, "Some Relations Between Extended and Unscented Kalman Filters," *Transactions on Signal Processing*, vol. 60, no. 2, pp. 545–555, 2012.

[192] J. Mangelson, M. Ghaffari, R. Vasudevan, and R. Eustice, "Characterizing the Uncertainty of Jointly Distributed Poses in the Lie Algebra," *IEEE Transactions on Robotics*, 2020.

[193] G. Ovchinnikov, A. L. Pavlov, and D. Tsetserukou, "Windowed Multiscan Optimization using Weighted Least Squares for Improving Localization Accuracy of Mobile Robots," *Autonomous Robots*, vol. 43, no. 3, pp. 727–739, 2019.

[194] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," in *Computer Vision and attern Recognition (CVPR)*, pp. 652–660, IEEE, 2017.

[195] H. Coskun, F. Achilles, R. DiPietro, N. Navab, and F. Tombari, "Long Short-Term Memory Kalman Filters: Recurrent Neural Estimators for Pose Regularization," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5524–5532, 2017.

[196] J. Ko, D. Kleint, D. Fox, and D. Haehnelt, "GP-UKF: Unscented Kalman Filters with Gaussian Process Prediction and Observation Models," in *International Conference on Intelligent Robots and Systems (IROS)*, pp. 1901–1907, IEEE/RSJ, 2007.

[197] W. Liu, D. Caruso, E. Ilg, J. Dong, A. Mourikis, K. Daniilidis, V. Kumar, and J. Engel, "IMU-Based Pedestrian Dead Reckoning with Learned Motion Model," *IEEE Robotics and Automation Letters (RA-L)*, 2020.

[198] S. Y. Loo, A. J. Amiri, S. Mashohor, S. H. Tang, and H. Zhang, "CNN-SVO: Improving the Mapping in Semi-Direct Visual Odometry Using Single-Image Depth Prediction," in *International Conference on Robotics and Automation (ICRA)*, pp. 5218–5223, IEEE, 2019.

[199] J. Jeong, Y. Cho, Y.-S. Shin, H. Roh, and A. Kim, "Complex Urban LiDAR Data Set," in *International Conference on Robotics and Automation (ICRA)*, pp. 6344–6351, IEEE, 2018.

[200] I. Skog, P. Handel, J. O. Nilsson, and J. Rantakokko, "Zero-Velocity Detection—An Algorithm Evaluation," *Transactions on Biomedical Engineering*, vol. 57, no. 11, pp. 2657–2666, 2010.

[201] Safran, "Inertial navigation systems." https://www.safran-group.com/fr/video/13612, 2018.

[202] D. Scaramuzza and Z. Zhang, "Visual-Inertial Odometry of Aerial Robots," *Encyclopedia of Robotics*, 2019.

[203] J. Rehder, J. Nikolic, T. Schneider, T. Hinzmann, and R. Siegwart, "Extending Kalibr: Calibrating the Extrinsics of Multiple IMUs and of Individual Axes," in *International Conference on Robotics and Automation (ICRA)*, pp. 4304–4311, IEEE, 2016.

[204] J. Rohac, M. Sipos, and J. Simanek, "Calibration of Low-cost Triaxial Inertial Sensors," *Instrumentation & Measurement Magazine*, vol. 18, no. 6, pp. 32–38, 2015.

[205] P. Furgale, J. Rehder, and R. Siegwart, "Unified Temporal and Spatial Calibration for Multi-Sensor Systems," in *International Conference on Intelligent Robots and Systems (IROS)*, pp. 1280–1286, IEEE/RSJ, 2013.

[206] R. Clark, S. Wang, H. Wen, A. Markham, and N. Trigoni, "VINet: Visual-Inertial Odometry as a Sequence-to-Sequence Learning Problem," *AAAI*, 2017.

[207] M. A. Esfahani, H. Wang, K. Wu, and S. Yuan, "OriNet: Robust 3-D Orientation Estimation With a Single Particular IMU," *IEEE Robotics and Automation Letters (RA-L)*, vol. 5, no. 2, pp. 399–406, 2020.

[208] Y. Almalioglu, M. Turan, A. E. Sari, M. R. Saputra, P. de Gusmão, A. Markham, and N. Trigoni, "SelfVIO: Self-Supervised Deep Monocular Visual-Inertial Odometry and Depth Estimation," *arXiv*, 2019.

[209] H. Yan, S. Herath, and Y. Furukawa, "RoNIN: Robust Neural Inertial Navigation in the Wild: Benchmark, Evaluations, and New Methods," *arXiv*, 2019.

[210] F. Nobre and C. Heckman, "Learning to Calibrate: Reinforcement Learning for Guided Calibration of Visual–Inertial Rigs," *The International Journal of Robotics Research*, vol. 38, no. 12-13, pp. 1388–1402, 2019.

[211] D. Schubert, T. Goll, N. Demmel, V. Usenko, J. Stuckler, and D. Cremers, "The TUM VI Benchmark for Evaluating Visual-Inertial Odometry," in *International Conference on Intelligent Robots and Systems (IROS)*, pp. 1680–1687, IEEE, 2018.

[212] P. Geneva, K. Eckenhoff, W. Lee, Y. Yang, and G. Huang, "OpenVINS: A Research Platform for Visual-Inertial Estimation," *International Conference on Intelligent Robots and Systems 'IROS) Workshop*, 2019.

[213] G. Lu and F. Zhang, "IMU-Based Attitude Estimation in the Presence of Narrow-Band Noise," *IEEE Transactions on Mechatronics*, vol. 24, no. 2, pp. 841–852, 2019.

[214] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *International Conference on Machine Learning*, vol. 37, 2015.

[215] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for Activation Functions," *International Conference on Learning Representation (ICLR)*, 2018.

[216] F. Yu and V. Koltun, "Multi-Scale Context Aggregation by Dilated Convolutions," *International Conference on Learning Representation (ICLR)*, 2016.

[217] Z. Zhang and D. Scaramuzza, "A Tutorial on Quantitative Trajectory Evaluation for Visual(-Inertial) Odometry," in *International Conference on Intelligent Robots and Systems (IROS)*, pp. 7244–7251, IEEE/RSJ, 2018.

[218] J. Delmerico and D. Scaramuzza, "A Benchmark Comparison of Monocular Visual-Inertial Odometry Algorithms for Flying Robots," in *International Conference on Robotics and Automation (ICRA)*, pp. 2502–2509, IEEE, 2018.

[219] I. Loshchilov and F. Hutter, "SGDR: Stochastic Gradient Descent with Warm Restarts," *International Conference on Learning Representation (ICLR)*, 2016.

[220] J. Delmerico, T. Cieslewski, H. Rebecq, M. Faessler, and D. Scaramuzza, "Are We Ready for Autonomous Drone Racing? The UZH-FPV Drone Racing Dataset," in *International Conference on Robotics and Automation (ICRA)*, pp. 6713–6719, IEEE, 2019.

[221] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6765–6816, 2017.

[222] J. Jeong, Y. Cho, Y.-S. Shin, H. Roh, and A. Kim, "Complex Urban Dataset with Multi-Level Sensors from Highly Diverse Urban Environments," *The International Journal of Robotics Research*, 2019.

[223] X. Roynard, J.-E. Deschaud, and F. Goulette, "Paris-Lille-3D: A Large and High-Quality Ground-Truth Urban Point Cloud Dataset for Automatic Segmentation and Classification," *The International Journal of Robotics Research*, vol. 37, no. 6, pp. 545–557, 2018.

[224] X. Roynard, J.-E. Deschaud, and F. Goulette, "Paris-Lille-3D: A Point Cloud Dataset for Urban Scene Segmentation and Classification," in *Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 2108–21083, IEEE, 2018.

[225] Y. Yang, P. Geneva, K. Eckenhoff, and G. Huang, "Degenerate Motion Analysis for Aided INS With Online Spatial and Temporal Sensor Calibration," *IEEE Robotics and Automation Letters (RA-L)*, vol. 4, no. 2, pp. 2070–2077, 2019.

[226] B. Wagstaff and J. Kelly, "LSTM-Based Zero-Velocity Detection for Robust Inertial Navigation," in *International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, 2018.

[227] D. K. McClish, "Analyzing a Portion of the ROC Curve," *Medical Decision Making*, vol. 9, no. 3, pp. 190–195, 1989.

[228] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, and D. Cournapeau, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[229] R. G. Thalagala, *Comparison of State Marginalization Techniques in Visual Inertial Navigation Filters*. PhD thesis, Memorial University of Newfoundland, 2019.

[230] V. Peretroukhin, L. Clement, and J. Kelly, "Inferring Sun Direction to Improve Visual Odometry: A Deep Learning Approach," *The International Journal of Robotics Research*, vol. 37, no. 9, pp. 996–1016, 2018.

[231] A. D. Maio and S. Lacroix, "On Learning Visual Odometry Errors," 2019.

CHAPTER 15

Appendix: Enhancing Kalman Filter with Deep Learning in Practice (Experience Feedback)

1 1 Introduction

This chapter presents a step-by-step guide for adding or improving an observation of a Kalman filter with deep learning. This part is build on the accumulation of acquired experiences in the previous chapters of Part II and Part III. We write this chapter in a tutorial and easy-to-follow form for the one that searches to improve its estimation algorithm with deep learning on the real but simple attitude estimation problem on the EuRoC dataset [106].

1.1 1.1 Retrospective Analysis

The results that we acquired in this thesis are demonstrative research works and there are still a gap between these algorithms, and a robust industrial solution.

It appears retrospectively that the zero velocity detector (see Chapter 11) and a neural network for denoising gyro (see Chapter 12) are closer to a concrete application than the works we presented in Chapter 8 and in [156]. Indeed, the first ones:

- are efficient to train and fine-tune, e.g. Chapter 12 trains neural networks in less than five minutes, and scales well with large amount data;
- are analyzable as we can compare the network outputs and the expected results, and see where results may be improved;
- and constitutes in itself a standalone block. Chapter 11 proposes a neural network to detect a car stop, that is invariant w.r.t. ad hoc filter modifications.

In contrast, the second part of works, albeit promising, are much difficult to generalize and optimize to large experiments, see similar works in [230,231]. Chapter 8 designs a neural network that must be adapted each time the filter changes, and [156] uses heavy and cumbersome deep Gaussian processes which are difficult to train.

1.2 1.2 Goals and Organization of the Chapter

In the following, we present an approach that should be used for enhancing a Kalman filter with deep neural networks. We consider the problem of attitude estimation from

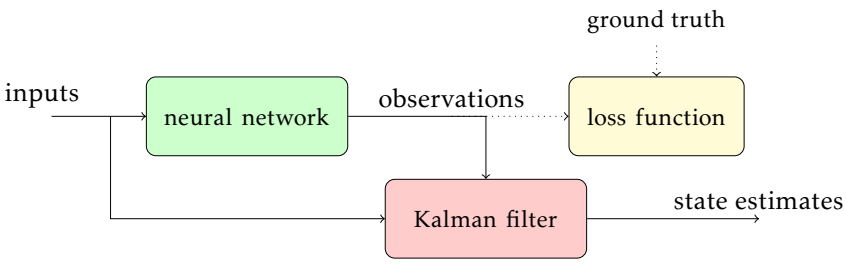


Figure 15.1: Overview of the proposed approach in a general situation. A neural network learns to regress observations from sensor inputs, which generally requires ground truth during training. A Kalman filter leverages learned observation to obtain reliable state estimates.

an IMU where we yet dispose of a (invariant) Kalman filter. We then apply a step-by-step recipe that search to obtain more reliable and efficient Kalman filter estimates, where the learning block provides to the filter observations and observation uncertainty, see Figure 15.1.

This chapter is organized as follow. Section 2 presents the problem of attitude estimation from an IMU and the related invariant Kalman filter that we search to improve. Section 3 describes the method for learning accurate gravity observations that are then feed to the filter. Section 4 evaluates and shows how the patient approach outperforms the fast-and-furious ones. Finally, Section 5 concludes the chapter.

2 2 Problem Modelling

This section briefly describes the model used in this chapter. The 3D orientation of a rigid platform is obtained by integrating gyro outputs of an IMU through

$$\mathbf{R}_n = \mathbf{R}_{n-1} \exp(\boldsymbol{\omega}_n dt), \tag{15.1}$$

where the rotation matrix $\mathbf{R}_n \in SO(3)$ at timestamp n maps the IMU frame to the global frame, the angular velocity $\boldsymbol{\omega}_n \in \mathbb{R}^3$ is averaged during dt, and with $\exp(\cdot)$ the $SO(3)$ exponential map, see Chapter 12.

The IMU is assumed unbiased and measurements are given as

$$\begin{bmatrix} \boldsymbol{\omega}_n^{\text{IMU}} \\ \mathbf{a}_n^{\text{IMU}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\omega}_n \\ \mathbf{a}_n \end{bmatrix} + \mathbf{w}_n, \tag{15.2}$$

where $\mathbf{w}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_n)$ is Gaussian noise. The model (15.2) assumes the sensor is calibrated and does contain the remaining parameters of Chapter 12.

Problem Statement: *given an initial attitude* $\mathbf{R}_0$, *performs in real time IMU attitude estimation.*

2.1 2.1 The Invariant Kalman Filter for Attitude Estimation

To obtain reliable estimates, we design an invariant Kalman filter based on the propagation model (15.1) and observation of gravity direction. Indeed, if velocity variation is negligible w.r.t. gravity, the accelerometer measures

$$\mathbf{a}_n^{\text{IMU}} \approx \mathbf{R}_n^T \mathbf{g}, \tag{15.3}$$

and we choose to consider the normalized acceleration measurement

$$\begin{aligned}\mathbf{y}_n &= \frac{\mathbf{a}_n^{\text{IMU}}}{\|\mathbf{a}_n^{\text{IMU}}\|} \\ &= \mathbf{R}_n^T \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \mathbf{R}_n^T \mathbf{g}', \end{aligned} \tag{15.4}$$

which contains orientation information around the gravity $\mathbf{g}$. Building on the right-invariant error

$$\mathbf{R}_n = \psi(\hat{\mathbf{R}}_n, \xi_n) \tag{15.5}$$

$$= \exp(\xi_n)\hat{\mathbf{R}}_n, \tag{15.6}$$

see Section 2 of Chapter 2, we obtain a filter whose equation are recapped in Algorithm 11, where

$$\mathbf{H} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{15.7}$$

Algorithm 11: invariant Kalman filtering for attitude estimation

```
// State and covariance propagation
```
1 $\hat{\mathbf{R}}_{n|n-1} = \hat{\mathbf{R}}_{n-1|n-1} \exp(\omega_n^{\text{IMU}})$;
2 $\mathbf{P}_{n|n-1} = \mathbf{P}_{n-1|n-1} + \mathbf{Q}_n$;
```
// Measurement update
```
3 $\mathbf{K} = \mathbf{P}_{n|n-1}\mathbf{H}^T \left(\mathbf{H}\mathbf{P}_{n|n-1}\mathbf{H} + \text{cov}(\mathbf{y}_n)\right)^{-1}$; // Kain
4 $\hat{\mathbf{R}}_{n|n} = \exp\left(\hat{\mathbf{R}}_{n|n-1}\mathbf{y}_n - \mathbf{g}'\right)\hat{\mathbf{R}}_{n|n-1}$;
5 $\mathbf{P}_{n|n} = (\mathbf{I} - \mathbf{K}\mathbf{H})\mathbf{P}_{n|n-1}(\mathbf{I} - \mathbf{K}\mathbf{H})^T + \mathbf{K}\,\text{cov}(\mathbf{y}_n)\mathbf{K}^T$; // Joseph form

This invariant Kalman filter mimics the complementary filter [17] with the advantage here that the gain $\mathbf{K}$ is optimal given $\mathbf{Q}_n$ and $\text{cov}(\mathbf{y}_n)$ while Jacobian matrices are constant. The measurement (15.4) provides information around the gravity direction, such that the roll and the pitch of the sensor are observable, while the yaw keeps non observable and its estimates drifts along time.

3 3 Enhancing the Kalman Filter with Deep Learning

We describe in this section how improving the Kalman filter observation with deep learning, as measurements (15.4) are noisy and setting its uncertainty require apparently manual tweaking. We first design a neural network and a training scheme to obtain reliable observations of the gravity orientation $\mathbf{y}_n$ along with the observation uncertainty $\text{cov}(\mathbf{y}_n)$, see Figure 15.2.

We first recalls two main features of deep learning that any user should be conscious:

- **Neural Network Training is a Leaky Abstraction:** it is allegedly easy to get started with training neural networks. Popular machine learning frameworks

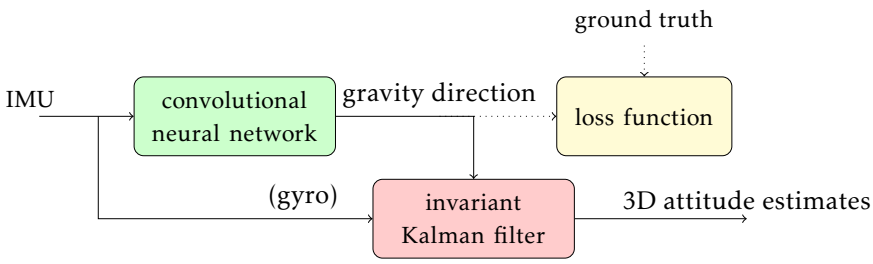


Figure 15.2: Overview of the proposed approach applied to the considered attitude estimation problem. A convolutional neural network learns to regress the gravity direction from inertial signals, which requires ground truth orientation during training. An invariant filter leverages estimated gravity direction with dead-reckoning orientation from gyro to obtain accurate estimates.

such as PyTorch proposes less than hour tutorial where less than 30 lines of miracle snippets solve an image classification problem, giving the (false) impression that deep learning is plug and play.

Unfortunately, neural networks are nothing like that. They are not "off-the-shelf" technology or theory the second one deviate slightly from training a vanilla neural network classifier. ADAM [162] does not magically make the network trains, batch normalization [214] does not magically make it converge faster, ... Using the technology and theory without understanding how it works are prone to fail.

- **Neural Network Training Fails Silently:** when the implementation of a filter contains mistakes, the filter often crash or gets definitely bad results. The situation where the filters "surprisingly" works whereas it should not (a misconfiguration or an incorrect propagation matrix for example) barely appears compared to deep learning.

  Indeed, when it comes to training neural networks, everything could be correct syntactically, but the whole thing can not be arranged properly, and it is really hard to tell. The possible error are large, logical (as opposed to syntactic), and very tricky to test. Therefore, a misconfigured neural network will most of the time train but silently work a bit worse.

In light of the above points, one should be highly defensive and obsessed with visualizations of basically every possible thing. The qualities that correlate most strongly to success are patience and attention to detail.

We now details chronologically the steps to successively obtain a neural network able to improve the estimates of the filter. This starts by being an expert of the full non-learning problem, i.e. the Kalman filter and data, that lead to appropriately defining what one can expect from the neural networks. Then, we follow all the gold standard of deep learning. In particular, we build from simple to complex and at every step of the way we make concrete hypotheses about what will happen and then either validate them with an experiment or investigate until we find some issue. Finally, we combine then the deep learning block and the Kalman filter, and evaluate the results of the enhanced filter.

3.1 3.1 Before Training: Visualizing the Non-Learning Problem

What we remark about leaky abstractions for neural networks also hold for any traditional problem. One must mastering its subject, possibly explains any strange behavior regarding data and the filter outputs, and thoroughly inspect data. This step is critical.

For our problem, we used the EuRoC dataset [106], that contains sequences of IMU data along with ground-truth. We search to understand the data, especially through visualization tools. Plotting inputs versus ground-truth or dead-reckoning estimates is essentials to perform basic checks: is the IMU vector contains first the gyro or the accelerometer, what are units, how is synchronization between ground-truth and inputs. This process is essential to detect fails and outliers in data. Concretely, it leads here to plot curves similar to the ones in Figure 15.3, and detect that the IMU is uncalibrated (see also Chapter 12), and break the model assumption (15.2). We use Chapter 12 to first calibrate the IMU sensor.

The same checking must be done for the Kalman filter. One must observe state estimates, state uncertainty ($\mathbf{P}_{n|n}$), residual, ... The filter itself must be clean: if its estimations are surprisingly bad after one minutes, it could be to $\mathbf{P}_{n|n}$ being non symmetric; or if $\mathbf{P}_{n|n}$ is too high, it is probably due to a misconfiguration of model parameters. Deep learning will not magically solve that.

At the middle of this step, both the dataset and the filter work as expected. Ones can then get a first idea of what to learn. This should be useful for the filter (improving something that works optimally is both difficult and inefficient) and possible. For the given problem, it is clear that the main limitation of the filter is the assumption done in (15.3), see Figure 15.4. We thus hope to learn (or correct) this observation. At least, we expect estimating its uncertainty, possibly better than a manual tweak on the accelerometer norm. Of course, this idea evolve along time. Learning uncertainty may just be to difficult if ground truth is insufficiently accurate.

3.2 3.2 Set Up the Training & Getting Baselines

Our next step is to set up a full training and evaluation skeleton and gain trust in its correctness via a series of experiments. At this stage it is best to pick some simple model as a very tiny CNN. We want to train it, visualize the losses, any other metrics (e.g. accuracy), model predictions, and perform a series of ablation experiments with explicit hypotheses along the way.

We follow the convolutional neural network structure of Chapter 12 that computes accelerometer correction $\tilde{\mathbf{a}}_n$ and uncertainty $\sigma_n \in \mathbb{R}$ from N past IMU measurements, where we first restrict it to one layer.

We now define the corrected and corrected measurement as

$$\hat{\mathbf{a}}_n = \mathbf{a}_n^{\text{IMU}} + \tilde{\mathbf{a}}_n \tag{15.8}$$

$$\hat{\mathbf{y}}_n = \frac{\hat{\mathbf{a}}_n}{\|\hat{\mathbf{a}}_n\|}, \tag{15.9}$$

where as in Chapter 12 the neural network is guided and initialized properly at $\mathbf{a}_n^{\text{IMU}}$. This thus provides us with the following simple baseline: the measurement where $\tilde{\mathbf{a}}_n = \mathbf{0}$.

We now define the loss function that must ensures measurement and uncertainty estimations as

$$\mathcal{L} = \sum_n \left(\frac{\mathbf{y}_n - \hat{\mathbf{y}}_n}{\sigma_n} \right)^2 + \log\left(2\pi\sigma_n^2\right), \tag{15.10}$$

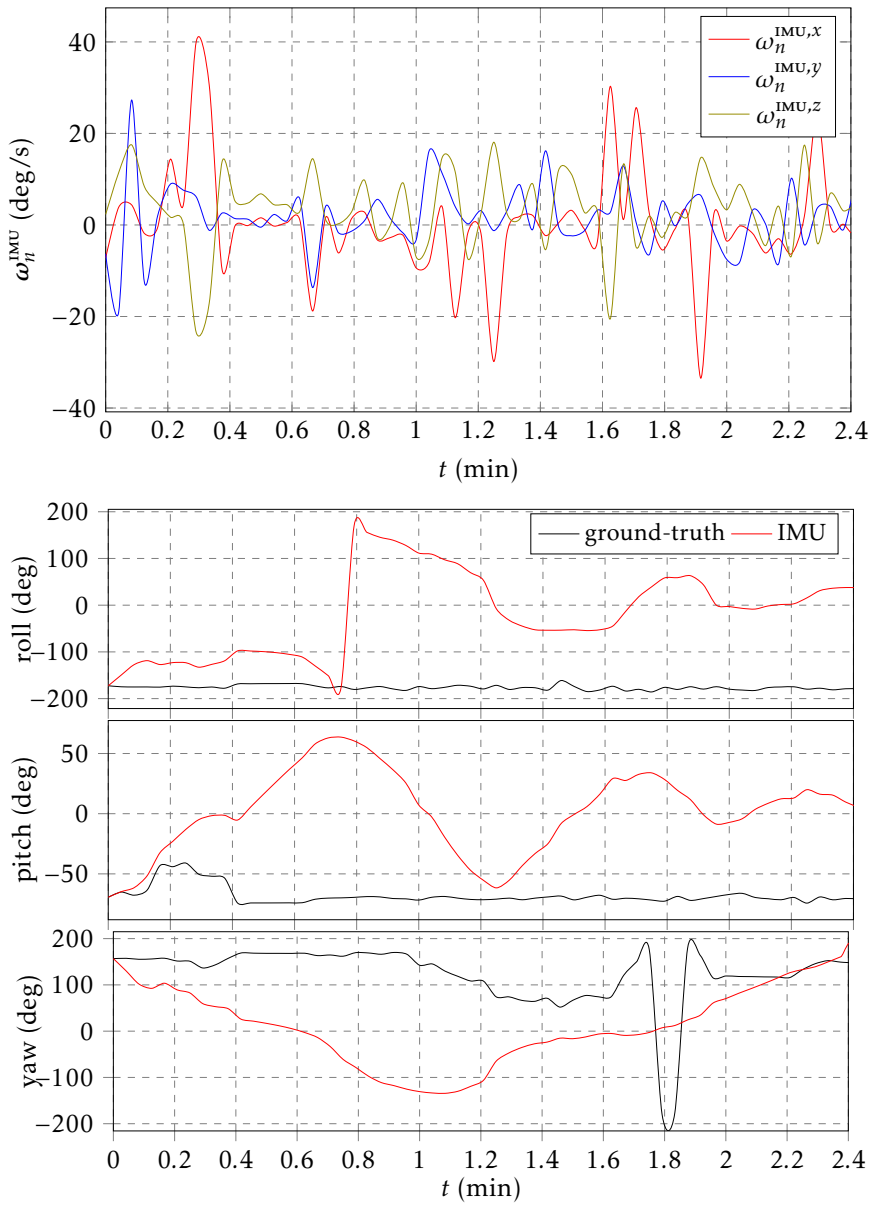


Figure 15.3: IMU measurements and dead-reckoning estimations. The IMU behavior is strange. Indeed, the IMU is is uncalibrated, and should be calibrated to correspond to the filter assumption.

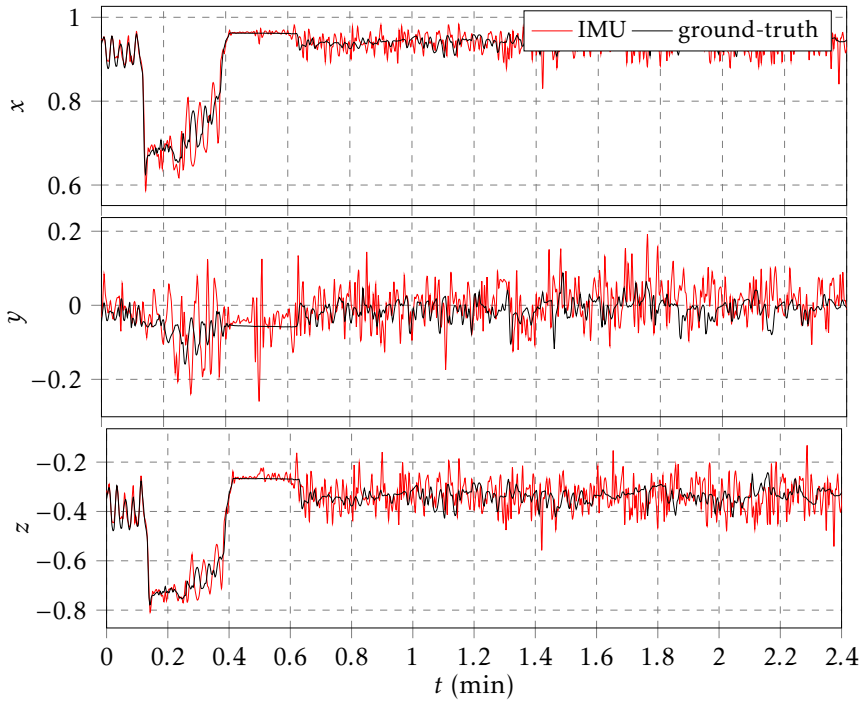


Figure 15.4: Accelerometer measurements nd true gravity direction along time on the sequence `V1_01_easy` of [106], where the acceleration assumption (15.3) is visibly broken. We observe that accelerometer is highly noisy to estimation gravity direction, even when the drone is static at the first secondes of the trajectory.

which is the negative log likelihood. At this stage, we are not certain that (15.10) is the best metric. Perhaps a robust l_1 metric is better, but this is fine-tunning question, see Section 3.3.

Now, we have a small network, a loss and a baseline. We train the network with vanilla optimizer and perform check. For instance, we verify loss at init and that it starts at the correct loss value, i.e. the one where $\tilde{\mathbf{a}}_n = \mathbf{0}$. Indeed, if the loss is very high, it just indicate we did not initialize the final layer weights correctly. Setting these correctly will speed up convergence and eliminate the first few iteration the network is basically just learning a bias.

We then monitor metrics other than loss that are human interpretable and checkable. The loss in (15.10) is difficult to understand, but we can compute and plot RMSE loss based on $\mathbf{y}_n - \hat{\mathbf{y}}_n$ and see how it evolves along training. We can also compare this metrics to the baseline.

Then we overfit a small batch. Overfit a single batch of only a few examples. To do so we increase the capacity of our model (here the kernel size) and verify that we can reach the lowest achievable loss (here minus infinity).

3.3 3.3 Training Fine-Tuning

Overfit & Regularize: At this stage we have a good understanding of the dataset and we have the full training and evaluation pipeline working. For any given model we reproducibly compute a metric that we trust. We are also armed with our performance for a baseline (that we should beat). The stage is now set for iterating on a good model.

The approach generally adopted to a good neural network model has two stages: first, we get a model large enough that it can overfit (i.e. focus on training loss) and then we regularize it appropriately (that increases the training but improves the validation loss). The advocated reason is that if we are not able to reach a low error rate with any model at all that may again indicate some issues, bugs, or misconfiguration. Then, we are at a place where we have a large model that is fitting at least the training set. We now regularize it and gain some validation accuracy by giving up some of the training accuracy.

We have two main advices:

- To reach a good training loss we choose an appropriate architecture for the data. We now do not search to be a hero. We were the first to be eager to get crazy and creative in stacking up the blocks of the neural net toolbox in various exotic architectures that make sense to us, but this was inefficient.

- ADAM is the safest optimized, which gas a large good learning rate region compared to other optimizers. We do not trust learning rate decay defaults, especially when we are re-purposing code from some other domain (mainly computer vision).

Network and Training hyperparameters Fine-Tunning: We are now “in the loop" with the dataset and explore a wide model space for architectures that achieve low validation loss. For simultaneously tuning multiple hyperparameters it may sound tempting to use grid search to ensure coverage of all settings, but it is best to use random search instead. Intuitively, this is because neural nets are often much more sensitive to some parameters than others. Indeed, there is a large number of fancy bayesian hyper-parameter optimization toolboxes around and our personal experience

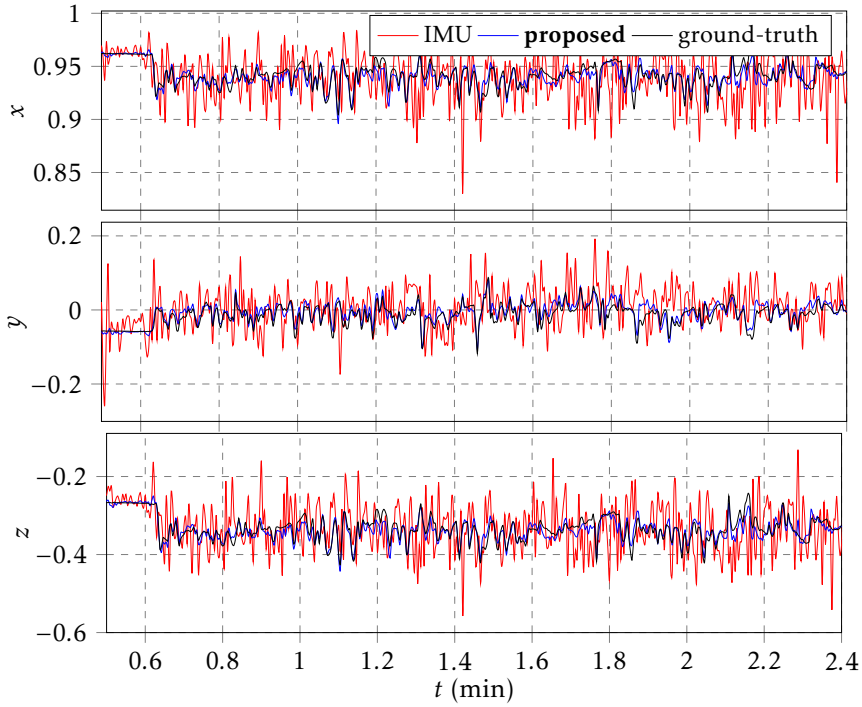


Figure 15.5: Accelerometer measurements nd true gravity direction along time on the sequence `V1 01 easy` of [106]. The neural network (blue) correctly estimates gravity direction.

is that the state-of-the art approach to exploring a nice and wide space of models and hyperparameters is to use trial-and-error.

Once you find the best types of architectures and hyper-parameters we now feed the filter with the networks. We then still massively visualizing the filter behavior and verify if measurements are correct and uncertainty relevant. For example, we found it is sufficient to learn the same parameters for each measurement $\boldsymbol{\sigma}_n = \sigma_n \mathbf{1}_3$. This has a spin-off advantage to make the Kalman filter really cheap in term of covariance matrix computation. All the gain can be parametrized with a scalar, see Appendix.

4 4 Results

We now dispose of a finely tunned neural network. We choose to evaluate the filter performances as evaluating the network alone is here difficult to properly correlate with the filter performance, and we show an example of network estimates in Figure 15.5. It visibly reduces the noise on the raw measurements.

4.1 4.1 Compared Methods & Metric

We compare the filter results with different configuration and learning methods, a.k.a. ablation study (in the context of deep learning, an ablation study refers to removing some feature of the training pipeline, and seeing how that affects performance.).

- **w/o calibration**, the proposed method without calibrating data;

- **w/o initialization**, the proposed method without properly initialing the network output in (15.9), i.e. $\hat{\mathbf{a}}_n = \tilde{\mathbf{a}}_n$;
- **w l_2 loss**, the proposed method where we adopt the l_2 loss $\sum_i \|\mathbf{y}_i - \hat{\mathbf{y}}_i\|^2$ during training instead of (15.10);
- **baseline**, the Kalman filter used without neural networks but calibrated data;
- **proposed**, the proposed method described in Section 3.

Each filter is parametrized as follow: $\mathbf{P}_0 = \mathbf{0}$, i.e. the state is initialized at ground truth, $\mathbf{Q}_n = q^2\mathbf{I}_3$ with $q = 1.10^{-5}$ rad/s and $\text{cov}(\mathbf{y}_n)$ is given by the neural network.

We evaluate the above methods on the *Absolute Orientation Error (AOE)*, which computes the mean square error between the ground truth and estimates for a given sequence as

$$\text{AOE} = \sqrt{\sum_{n=1}^{M} \frac{1}{M} \|\log\left(\mathbf{R}_n^T \hat{\mathbf{R}}_n\right)\|_2^2}, \tag{15.11}$$

with M the sequence length, $\log(\cdot)$ the $SO(3)$ logarithm map, and where the estimated trajectory has been aligned on the ground truth at the first instant $n = 0$. In contrast to Chapter 12, AOE error makes more sense as the filter is here able to recover roll and pitch.

4.2 4.2 Results

We provide to the filter the IMU input where the sensor is calibrated and the calibrated IMU of Chapter 12. Results are given in Figure 15.6. We observe

- The most important improvement occurs when we calibrate data, see difference between w/o calibration and baseline. It evidences that careful data analysis is required to provide yet accurate baseline;
- The assumption of constant speed in the measurement update indeed degrades performances for these small sequences if we compare baseline with results of Chapter 12. The assumption (15.3) is not really valid;
- Properly initializing the correction helps to improve results, as the network outputs weak values and we can thus regularize the neural network more efficiently;
- There are no importance of optimal loss here. Correctly fixed the measurement noise $\mathbf{R}$ is sufficient.

5 5 Conclusion

This chapter shows a general approach for safely improving a Kalman filter with deep learning, where a deep learning block is designed and provides the filter with valuable observations. This lead to a method that leads practically to improve the filter (in contrast to method that starts a deep neural network from scratch) and where multiple well trained learning blocks can be stacked. On the attitude estimation problems from an IMU, it leads to improve the estimation of the gravity measurements that can be used to obtain accurate attitude results with minimal computational requirement.

sequence	**w/o calibration**	**w/o initializing**	**w l_2 loss**	**baseline**	**proposed**
MH 02	15.4	1.36	1.31	7.69	**1.30**
MH 04	14.6	1.35	1.27	5.24	**1.25**
V1 01	10.9	**1.14**	**1.14**	2.13	1.15
V1 03	8.9	2.70	2.60	3.96	**2.54**
V2 02	7.7	3.65	**3.30**	4.85	3.35
average	11.5	2.04	2.10	4.76	**1.91**

Figure 15.6: Results in term AOE when filter inputs are provided by calibrated IMU. The improvement between the baseline and the proposed approach is noticeable.

Appendix: Efficient Kalman Filtering

In the particular case where the propagation noise covariance matrix $\mathbf{Q}_n = \mathbf{Q}$ is static and that the measurement noise covariance matrix $\text{cov}(\mathbf{y}_n) = \sigma_n^2\mathbf{I}$ is time-varying but isotropic, the Kalman filter only requires one parameter to compute the gain, which is computationally really efficient.

We assume the filter is initialized with $\mathbf{P}_0 = \mathbf{0}$, and that $\mathbf{Q} = q\mathbf{I}_3$, and $\text{cov}(\mathbf{y}_n) = \sigma_n^2\mathbf{I}$.

Proposition: only a scalar p_n is required to compute the Kalman gain.

Proof: the proof is shown by induction. The proposition hold for $n = 0$. We assume that if it hold for $n-1$, i.e.

$$\mathbf{P}_{n-1} = \begin{bmatrix} p_{n-1} & 0 & 0 \\ 0 & p_{n-1} & 0 \\ 0 & 0 & \tilde{p}_{n-1} \end{bmatrix} \tag{15.12}$$

, where $\tilde{p}_{n-1}$ is any scalar, then the proposition hold for n. After propagation, we obtain

$$\mathbf{P}_n = \mathbf{P}_{n-1} + \mathbf{Q} \Rightarrow p_n = p_{n-1} + q, \tilde{p}_n = \tilde{p}_{n-1} + q. \tag{15.13}$$

And at update, we compute

$$\mathbf{S} = \mathbf{H}\mathbf{P}_n\mathbf{H}^T + \text{cov}(\mathbf{y}_n) \Rightarrow \mathbf{S} = (p_n + \sigma_n^2)\mathbf{I}_2 \tag{15.14}$$

$$\mathbf{K} = \mathbf{P}_n\mathbf{H}^T\mathbf{S}^{-1} \Rightarrow \mathbf{K} = \frac{p_n}{p_n + \sigma_n^2}\mathbf{H}^T \tag{15.15}$$

$$\mathbf{P}_n^+ = (\mathbf{I} - \mathbf{K}\mathbf{H})\mathbf{P}_n \Rightarrow p_n^+ = p_n(1 - p_n/(p_n + r_n)) \tag{15.16}$$

$$p_n^+ = \frac{p_n r_n}{p_n + \sigma_n^2}, \tag{15.17}$$

which conclude the proof.

RÉSUMÉ

Cette thèse aborde les méthodes d'estimation d'état pour les véhicules équipés de divers capteurs tels que des caméras et des centrales inertielles. Le filtre de Kalman est un outil largement utilisé pour estimer l'état d'un système dynamique, qui soulève des questions théoriques pour les systèmes non linéaires présents dans la navigation, et qui repose sur des modèles physiques et des paramètres qui doivent être optimisés efficacement par l'utilisateur. La thèse contribue au filtrage de Kalman pour la navigation, où les contributions se divisent en deux contributions majeures.
La première contribution consiste à s'appuyer sur le récent filtre de Kalman étendu invariant pour résoudre les problèmes difficiles que sont l'inconsistence du filtre de Kalman étendu pour le problème de la localisation et de la cartographie simultanées, de la navigation avec des capteurs visuels, et la question plus générale du filtrage de Kalman sur les variétés. La deuxième contribution consiste à utiliser des outils récents du domaine de l'intelligence artificielle, à savoir l'apprentissage profond, pour améliorer les filtres de Kalman, notamment pour relier les mesures des capteurs à l'état du système, et pour régler le filtre efficacement, c'est-à-dire utiliser des techniques d'apprentissage automatique pour trouver une stratégie de réglage dynamique des paramètres du filtre de Kalman qui correspond aux données et qui est également capable de fournir de nouvelles informations au filtre.
La thèse introduit ainsi différents algorithmes de filtrage et réseaux de neurones profonds dont l'implémentation est rendue open-source.

MOTS CLÉS

fusion de capteurs, filtre de Kalman, SLAM, navigation, centrale inertielle, apprentissage profond

ABSTRACT

This thesis deals with state estimation for vehicles that are equipped with various sensors such as cameras and inertial measurement units. The Kalman filter is a widely used tool that estimates the state of a dynamical system, which raises theoritical questions for the nonlinear systems present in navigation, and that relies on physical models and parameters that need to be efficient tuned by the user. This thesis contributes to Kalman filtering for navigation, where the contributions focus on two different manners.
The first manner consists in building on the recent invariant extended Kalman filter to address challenging issues, namely the inconsistency of extended Kalman filter for the problem of simultaneous localization and mapping, navigation with vision sensors, and the more general question of Kalman filtering on manifolds. The second manner consists in using recent tools from the field of artificial intelligence, namely deep learining, to improve Kalman filters, notably to relate sensors' measurements to the state, and to tune the filter efficiently, that is, using machine learning techniques to find a dynamical tuning strategy of the Kalman filter parameters that best matches the data and that is also able to provide new information to the filter.
The thesis thus introduces differents filtering algorithms and deep neural networks whose implementation are made open-source.

KEYWORDS

Sensor fusion, Kalman filter, SLAM, navigation, IMU, deep learning