Towards New Sensing Capabilities for Legged Locomotion using Real-Time State Estimation With Low-Cost IMUs

Dinesh Atchuthan

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ FÉDÉRALE TOULOUSE MIDI-PYRÉNÉES

Délivré par :
l'Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)

Présentée et soutenue le *23/10/2018* **par :**

Dinesh ATCHUTHAN

Towards New Sensing Capabilities For Legged Locomotion Using Real-Time State Estimation With Low-Cost IMUs

---

JURY

Patrick RIVES	Directeur de Recherche	Rapporteur
Silvère BONNABEL	Professeur	Rapporteur
David FILLIAT	Directeur de Recherche	Examinateur
Manon KOK	Assistant Professor	Examinateur
Patrick WENSING	Assistant Professor	Examinateur
Cyril ROUSSILLON	Docteur Ingénieur	Examinateur
Nicolas MANSARD	Directeur de Recherche	Directeur de Thèse
Joan SOLÀ	Ramon y Cajal Researcher	Co-directeur de Thèse

École doctorale et spécialité :
EDSYS : Robotique 4200046
Unité de Recherche :
Laboratoire d'analyse et d'architecture des systèmes
Directeur(s) de Thèse :
Nicolas MANSARD et *Joan SOLÀ*
Rapporteurs :
Patrick RIVES et *Silvère BONNABEL*

Acknowledgments

For this work, I have been given the title of 'Doctor'. But It would be unfair to say that I did it alone. It is now my pleasure to thank all the people who contributed to it. Some of them already know that they helped me. But many simply do not realise it! Below will be given only an incomplete list of the people who made it possible to stand for three years.

First of all, it is customary to thank the directors Nicolas Mansard, Joan Solà and Olivier Stasse and I really think that it is deserved. This work would have been impossible without their advices, support, encouragements, explanations and the opportunity that they gave to me. Besides I learned a lot just talking with them while at the same time having fun.

Thank you Nicolas for your expertise, for your humanity and for trusting me when even I had doubts. Despite your busy schedule you always managed to find some time to not only take part to discussions but also to initiate some and to have an attentive look on the direction taken by this work.

Thank you also Joan for your involvement and the energy you put in this work from the very first day and even before the thesis started! You managed to have a close attention to what I was doing and to actually help me by getting your hands dirty. I think I will remember those intensive debug sessions we had to take care of for a long time!

Thanks to Olivier who supervised my work in the first year with enthusiasm for his advice! I think that this supervision team was really interesting since I had the opportunity to have different views of the same problem and different pedagogical approaches that helped me understand things!

My deepest thanks also to the reviewers of this thesis, Patrick Rives and Silvère Bonnabel, who took time to read the manuscript and give some valuable feedback. Of course I will not forget to thank also the examinators (David Filliat, Manon Kok, Patrick Wensing and Cyril Roussillon) for taking time to assist to the defense.

I would like to thank all my colleagues (not only from LAAS but also from IRI) and co-authors for their collaboration, their patience and the enlightening discussions we have had on many and varied subjects.

Thanks to 'Les barbus' team (that I may also call 'Nounours et C^{ie}') for making get out of the tiny manor I was living in and giving me the opportunity to visit places around Toulouse! Thanks also to the role play team for the adventures we lived towards gloomy lands with so much fun! Thanks to the tarot team for making everyday only more enjoyable. Many thanks to all these colleagues (not only from the robotics teams) who made the LAAS a friendly place where working is fun.

I also have to thank all the people who helped me to prepare my defense. There I need to mention my directors again but also Céline Pieters who did not only took

a critical look to my presentation and my speech but also gave me valuable ideas and materials! Special thanks also to Florent (*a.k.a.* Bernard) for all the energy he spent to help me prepare the post-defense cocktail!

My deepest thanks also to those friends I shared most of my time with during those three years. You provided a priceless help in times of need. You were always ready to listen to my concerns and give your own vision and advice. I need to mention at least Thomas, Bernard and Kevin there but the list is far to be limited to them.

Finally, I would like to thank my family for encouraging me to keep learning new things and the obstacles they overcame so that I can study in good conditions. And now I would like to express my most sincere thanks to the racoon-fangirl, Clémence (*a.k.a.* 'Madame Patate') and the Berserker for their moral support, availability and kindness. I already told you how important you were to me but it will make no harm to tell it one more time...

Contents

List of Figures

Introduction

Perception in living beings

Most living beings benefit from their capacity to successfully navigate in space largely thanks to their capacity of localise themselves. This localisation cannot be separated from perception and from the development of sensors with new capabilities. Good examples of the aforementioned sensors are the eyes of animal species which differ from each other (Fig. 1) according to needs in terms of survival and in terms of the environment in which they live (*e.g.* on the surface / under water/ ...).

However, living beings do not only rely on a single sensor but rather have a multitude of sensors that are used together in a complex way that has not been fully understood yet. A simple way to understand how crucial sensor fusion is in our everyday life is to consider the importance of the inner ear for navigation purposes. The inner ear (see Fig. 2) is not only responsible for sound detection and the balance of our body but also for the perception of the head's angular position and its acceleration. A failure in this sensor can cause the loss of balance and therefore difficulties to navigate in one's environment. Furthermore, the Human proprioception includes the sense of the relative position of one's own parts of the body and strength of effort being employed in movement [Glanze 1994]. Proprioceptors are present in our muscles, tendons and joints to provide us with the sense of pain, hunger or the position of our members and giving rise to the kinesthetic sense. [Bloom 1988] defines kinesthesis as part of the somatosensory system that is conscious bodily perception distributed throughout the whole body. Kinesthesis is the subject of various studies concerning its relation to movements [Young 1945] alone or with other senses [Lovelace 1989, Day 1964].

Let us take the example of a cheetah chasing a prey. Its sensors are acquiring much information about not only the world around (structure of the ground, position and velocity of the prey, ...) but also its internal state (position of limbs, forces applied on the body, interaction with the environment, ...). The central nervous system is then responsible for managing all the incoming information and taking a decision. In some cases, the central nervous system may even not take action as it is the case for reflex. This complex sensory system then allows to react to changes in the system, may it be internal changes or related to the environment itself. We can mention the specific case of the unintended (and maybe unexpected) slippage of paws during running. We could say with cautions that the cheetah is then able to dynamically adapt its motion to balance the effect introduced by the slippage.

Perception in robotics

As for human beings, estimating the position of a mobile robot and its configuration is a mandatory step performed through the processing of available information. In-

(a) Fly

(b) Frog

(c) Hogna Wolf spider

Figure 1: Photos of the eyes of different species.

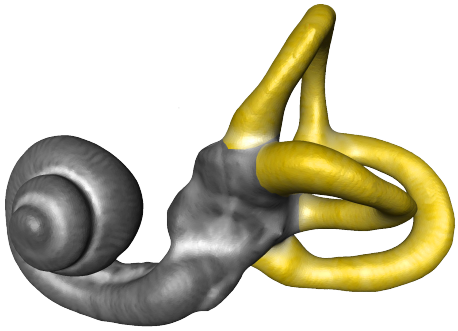


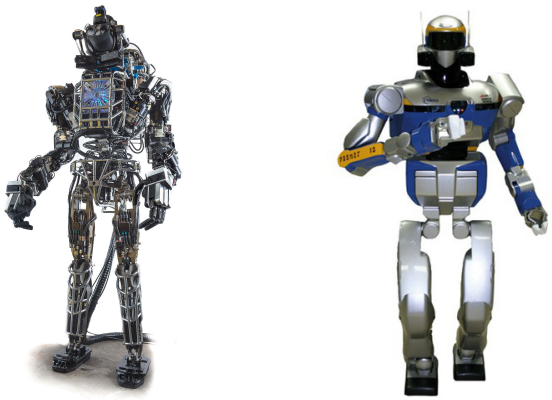
Figure 2: **Vestibular apparatus.** Tomography 3D of the vestibular system. The yellow parts are the three semicircular canals in charge of sensing rotational movements. Otolithic organs are located at the base of the semicircular canal system. They sense the linear accelerations of the head.

formation are provided by various sensors that can be separated into two categories:

proprioceptive sensors measuring values internal to the robot which can be for example the joint angles of the robot, the motor speed, its temperature or its battery voltage level. Examples of such sensors are encoders, gyroscopes, potentiometers, etc.

exteroceptive sensors acquiring measurements from their observation of the outside world. We can cite sonar sensors, ultrasonic distance sensors, cameras or accelerometers due to the measure of the gravity vector as examples. The gathered data must be interpreted to extract a meaningful feature.

The data must then be processed to have an estimation of the state of the robot that would be as close as possible to the reality. However, robots are required to



(a) Atlas
Boston Dynamics

(b) HRP-2
Kawada Industries

Figure 3: **Humanoid robots**. Illustration of some humanoid robots cited in this manuscript.

navigate in a changing environment and to interact with it to gain access to some parts inside it or in order to modify it. For this reason, we are likely to ask not only for a precise localisation but we may also need it at a high update rate.

Mobile robots usually have a wide variety of sensors (both exteroceptive and proprioceptive) integrated in their systems. Fig. 4 shows examples of sensors integrated in some mobile robots while Fig. 5 hightlights one possible solution for the design of a humanoid robot. The kinematic chain of the robot is formed of the succession of joints and encoders are used to move each segment. Force and torque sensors are usualy placed at end-effectors while a high-quality inertial measurement unit (IMU) is located in the chest of the robot. Cameras are also placed on the head of the robot. Other designs are possible and other sensors can be placed on the robot. At the end, these sensors will be providing some information that will be used in order to achieve specific tasks such as keeping the robot correctly balanced, reaching a point in space or localising the robot in its environment.

Regarding the localisation problem, each sensor has its own qualities and drawbacks. Localisation techniques such as Global Positioning System (GPS) have been investigated mostly for mobile robot navigation [Ohno 2003,Ohno 2004,Reina 2007]. GPS provides absolute measurements but is not well suited for indoor localisation. A popular alternative to GPS is the use of odometry which integrates information provided by a given sensor. Some examples are:

- Encoder odometry integrating all small displacements provided by robot encoders to describe the robot's trajectory. For example, in the case of a wheeled robot, the odometry uses the rotations of the wheels to estimate the trajectory of the robot.

- Visual odometry through the use of cameras extracting information from the

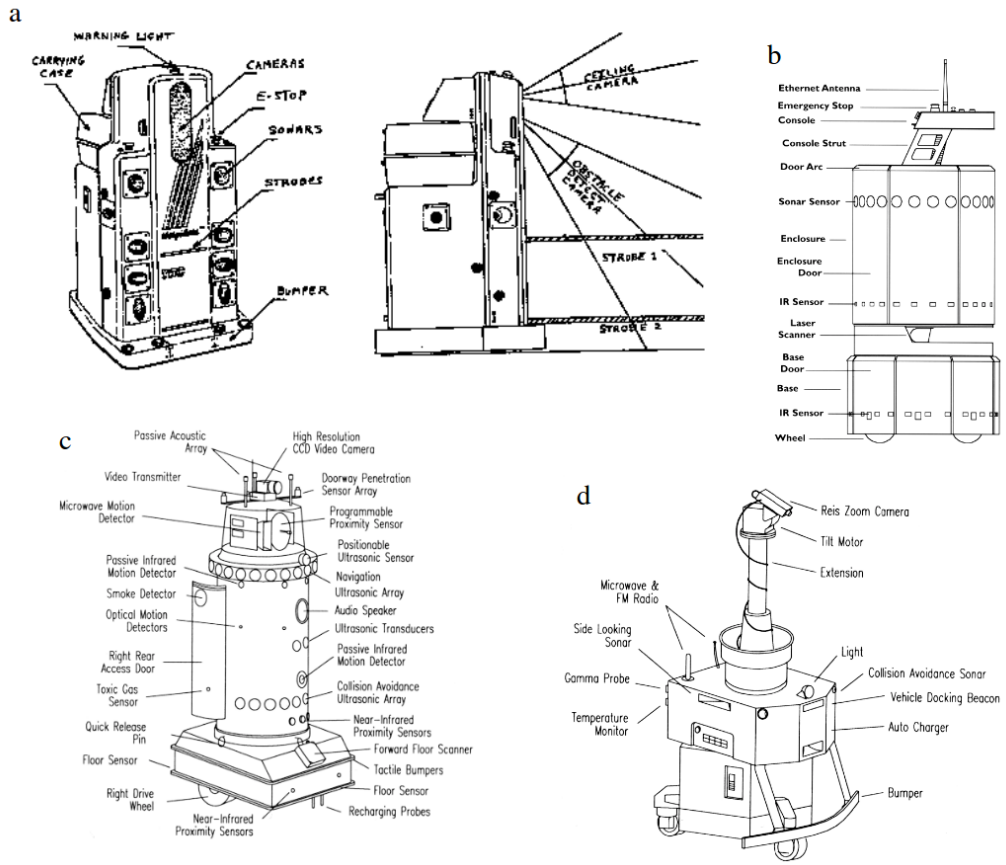


Figure 4: [Siegwart 2011] Examples of robots with multi-sensor systems: (**a**) HelpMate from Transition Research Corp, (**b**) B21 from Real World Interface, (**c**) Roboart II, built by H.R. Everett [Everett 1995], (**d**) The Savannah River Site nuclear surveillance robot

images (*e.g.* in [Comport 2010]). Visual odometry methods are still used not only in mobile robotics but also for unmanned aerial navigation, pedestrian navigation and others. For example, this method can be used for visual servoing [Tsakiris 1997, Chaumette 1991].

If odometry is a theoretically possible solution, the perfect sensor assumption makes its use unrealistic and results in estimations drifting from the real state. Imperfections and errors due to sensors can also be produced by the data processing step. Estimators tend to tackle these problems by defining sensor models working under more realistic assumptions used to simplify the problem. Besides, sensor fusion is used to get complementary information enhancing the state estimates. We can distinguish two main categories of estimators: filters and batch optimisation estimators. On the one hand, filtering methods compute the state of the robot using the data on top of the last estimate of the robot. During this estimation, all past

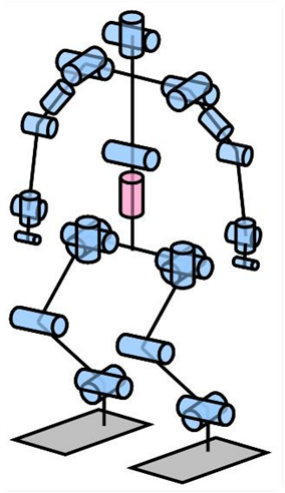


Figure 5: Simplified representation of the structure of a humanoid robot. This figure only shows one possible design.

poses are marginalised out and the information gained over time is summarised as a probability distribution. The idea of such filters is based on the principle of the Bayes Filter that will be introduced in Section 1.2. Kalman Filters are the most popular solution of this category. The popularity of Kalman Filter based solutions can be explained by both their performances and their simple implementation. On the other hand, methods belonging to the batch optimisation category tend not to marginalise out past poses but rather to keep this information in the system. The estimation does not only concern the last poses but rather all selected poses represented by the so called: "key-frames" along the trajectory. The most naive way to work with these methods is to estimate the states of all the key-frames. The more key-frames are estimated, the more computationally expensive and time consuming the estimation will be. Key-frames not involved in the estimation are rather ignored than marginalised out, implying a different sparsification of the problem when compared to filter based methods.

Thesis overview

Rationale

Sensory input and data management form a core problem in robotics since they are at the root of all solutions. The common solution for the design of legged robots is very similar to the one presented in Fig. 5. The encoders are then used to move the robot while the high-quality IMU located in the chest can be used to estimate the pose of the chest and cameras are used to estimate the pose of the robot in its environment. Although we can now find impressive demonstrations of legged robots running, jumping and performing various tasks, we can hardly claim to achieve locomotion tasks or to design perception systems that are as effective as

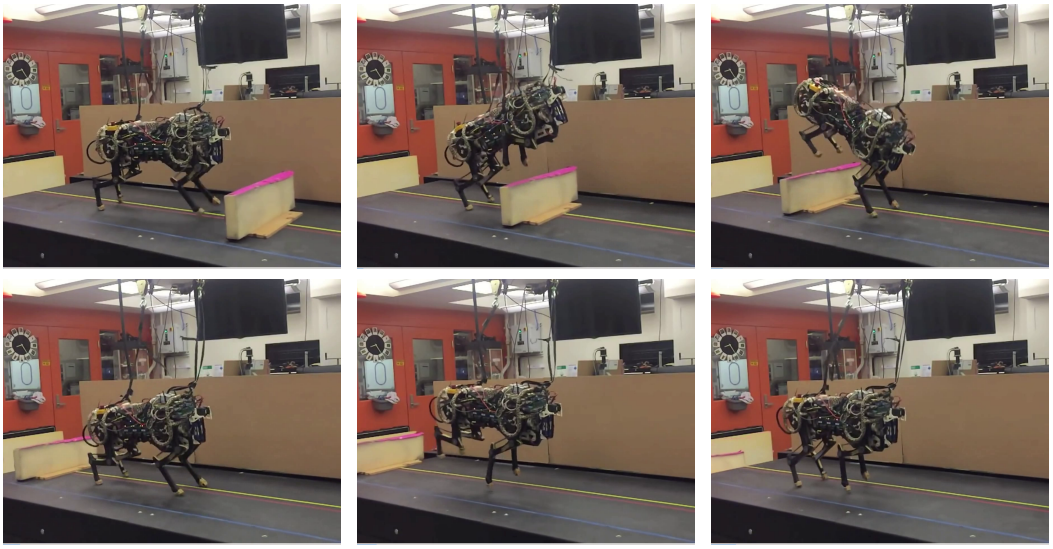


Figure 6: Snapshots of the cheetah robot jumping over an obstacle.

for living beings. Thus let us take again the slippage example while recalling that this phenomenon is mostly unexpected and undesired in robotics. Some strategies need to be used to tackle the slippage problem since it introduces problems in terms of balance, control but also localisation. In 2010, Boston Dynamics released a video showing the Bigdog robot walking in different outdoor scenarii including a frozen floor. However, to tackle efficiently the locomotion problem on slipping floors is still a challenge. In [Park 2015], the authors presented a framework that enables the Cheetah robot to autonomously run over obstacles up to 40 cm in height as illustrated in Fig. 6. We can also notice that the robot jumps again right after the landing phase before reaching the normal running phase. However, how would we manage slippage in this situation taking into account that motion planning methods usually make the no-slippage assumption? This question might be relevant in real case scnearii.

Therefore we can wonder whether a different design could help to achieve better performances. IMUs (introduced in Chapter 2) can provide a localised perception that we could benefit of at high rates. Thereby the question that we can raise is whether placing multiple IMUs on the robot and specifically on the end-effectors could be interesting. This localised perception would provide information that could be used along with the other sensors in the robot through sensor fusion methods to have a better estimate of the state of the system. More specifically, this perception could bring some information to be used along with more generalised perception method such as SLAM (Simultaneous Localization And Mapping) methods that will be introduced in Section 1.1. This thesis is an attempt to build the first block that may be used to answer this question.

In order to get an interesting solution, we may need a perception system that can acquire data at a high frequency so that the realtime estimates can be used in a

control loop. Since placing multiple IMUs providing high quality measurements may become very costly, we investigate the use of low-cost IMUs. Although we mentioned the performance related issues, evaluating them is application-dependant. However, we tend to require solutions that can run in real-time. Hence, in this thesis, we focus on the use of a single IMU with the following constraints:

- use a low-cost IMU
- data are provided at 1 kHz
- the solution should be usable for real-time applications

Despite the apparent weaknesses of batch optimisation based methods regarding real-time applications, we tend to think that these solutions are suited to achieve our goals when working in a coherent framework. If this can be achieved, then we would have an optimal estimator with all the advantages when compared to filtering methods but with some disadvantages regarding the time-consuming estimation part. Efforts on formulating the problem and on an efficient use of the information provided by sensors are required to reach our goals.

A more detailed presentation of both filtering and batch optimisation methods is provided in the upcoming chapters. Our contribution is to apply the batch optimisation approach to problems mostly tackled with filtering methods while proposing a method that will allow to efficiently integrate new sensors as well as the information they provide. Hence our approach will be used for pedestrian localisation problems and extended to its use with a humanoid robot (Fig. 3b).

Thesis organisation

Overview

The three contributions presented in this thesis are composed of:

- The development of the quaternion-based preintegration methods and its application to pedestrian tracking along with a factor graph formulation of the problem. This contribution is developed in **Chapter 3** and is published as a conference paper [Atchuthan 2018].
- The investigations of similar techniques for use on a humanoid robot.
- A self-calibration procedure extending lessons learned from the preintegration method to its use with other manifolds detailed in **Chapter 4** and published with extended materials in [Solà 2018].

Historical narrative

Since we focus primarily on the use of IMUs for pose estimation while formulating the problem with a factor graph, we first need an effective way to use

the IMU. This is the purpose of the preintegration method proposed in [Lupton 2009] and applied to IMUs using direct preintegration with Euler angles. The preintegration method was improved in [Forster 2017] with preintegration on the $SO(3)$ manifold. The novelty and interest of the work can be recognized through the different contributions using this preintegration method can it be for navigation [Caruso 2017, Liu 2017, Hartley 2018] or calibration techniques [Kim 2018]. This preintegration method is explained using rotation matrices and cumbersome derivatives regarding the jacobians that could possibly lead to misunderstandings. Thus we first propose a quaternion-based preintegration method due to specific interests regarding the quaternion representation to represent orientations and provide simpler and clear derivatives using the chain rule. We believe that this new and elegant formulation makes the understanding of the preintegration method easier, thus resulting in fewer errors and the possibility to imagine other applications with similar methods. Secondly, we generalise the preintegration method to its use with any kind of manifolds. This work is a step further towards splitting things and acting more rationnaly with the chain rule while giving a clear definition of Jacobians in manifolds as opposed to the algebraic way presented in [Forster 2017]. We illustrate the use of this generalised preintegration method through its use for sensor self-calibration.

Chapter organisation and contributions

This thesis is organised as follows. **Chapter 1** introduces the optimal estimation theory by underlining the similarities and differences between the main popular filtering methods and the batch optimisation approach. It is both a tutorial and a state-of-the-art chapter. It relies on [Sola 2016], [Sola 2007] and [Koller 2009] as main sources, yet the content of the chapter is not fully covered by these sources.

Chapter 2 then introduces the IMU (acronym for Inertial Measurement Unit) technology due to its importance in this thesis. **Chapter 3** details the IMU preintegration method derived from a proposal of [Forster 2016] and also introduces some implementation on a humanoid robot. The development of the method was conducted in collaboration with Joan Solà and I am the author of its software implementation. This chapter partially relies on a submitted conference paper of which I am the main author.

Chapter 4 extends some solutions driven by remarks upon the preintegration method proposed in Chapter 3 to the usage of any kind of manifold. Joan Solà is the leader of this work in which I contributed in a software implementation and experimentation point of view. This work is published with extended materials in [Solà 2018].

Finally, **Chapter 5** introduces the long-term project this work is part of in the GEPETTO team as well as the localisation library (WOLF) in on-going development to which I contributed equally with Joan Solà (lead developer), Joan Vallvé, Angel Santamaria-Navarro, and Jeremy Deray, as well as other secondary developers.

Chapter 1

Estimation as filtering or online batch-optimisation problems

Contents

1.1 Definition of the problem

Let us suppose that you are walking in a haunted house attraction. Everything goes fine until you enter a new room at $t_0 = 0$. You can see a door you want to reach on the opposite wall, it is automatically included in your mind map as a point of reference to localise yourself in the room along with other objects you may have noticed. Suddenly, the lights are turned off and you lose all visual information. Because you had a map of the room at t_0 (before the lights are turned off) and did not move, you can assume to know where the door is and you can navigate towards it in the dark. This may seem like a very easy experiment! However, experiments show that walking straight forward to an objective with no visual information is a

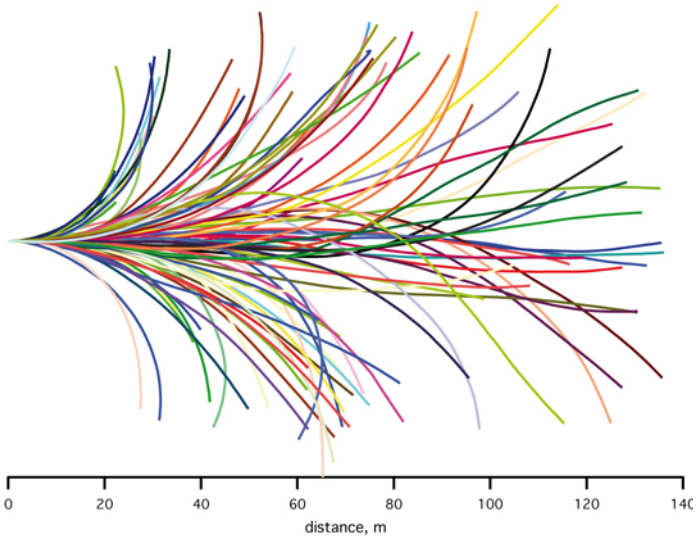


Figure 1.1: Representation of 88 trajectories acquired with 15 human subjects who were asked to walk straight forward during an experiment, ©Jean-René Cazalets

difficult task even for Humans as shown in Fig. 1.1 [Bestaven 2012]. In robotics, generating controls so that the robot follows a precise trajectory is still a challenging problem faced not only on the control part but also on the estimation point of view. When one generates a sequence of control designed to make the robot move from a place to another, the estimation of the state of the robot given only the initial position and the control sequence is called *odometry*.

Going back to our haunted house example, if the lights were still turned on then reaching the door would have been easy for us. This can be explained by the correction of the trajectory given our visual perception of the world. In robotics, this problem is called *Simultaneous Localisation And Mapping (SLAM)*.

This thesis is written in the context of estimation of the state of a mobile robot in an unknown environment. While our contributions mostly address the problem of localising the robot, and not that much of reconstructing the environment, we believe that our work naturally extends to the full localisation and mapping problem. It is important to understand the extended context before digging in our personal contributions.

1.1.1 Motivation and main concepts

The broader view is given by considering the problem of reconstructing the structure of the environment from camera motion. Structure-from-Motion (SfM) problems are dealing with the reconstruction of the three-dimensional (3D) structure of a (stationary) scene from a set of measurements extracted in two-dimensional (2D) images. Those measurements are specific features (point of interests, corners, lines, ...) detected in the images and projected onto the camera frame. The data-association step extracts matching features from one image (or frame) to another.

These associations are used to estimate the relative camera poses and thus camera motion between related frames. The minimisation of the re-projection error then leads to the reconstruction of the desired 3D structure. SfM is still a research field into which much effort is put. [Longuet-Higgins 1981] describes a simple algorithm to recover the 3D structure of a scene assuming that the association problem is solved, thus describing the problem as finding the relative orientations of the different view points.

SfM can be considered as a monocular visual Simultaneous Localisation And Mapping (SLAM) problem. Yet there is a slight difference between SfM and SLAM problems. While SfM problems are, generally speaking, about the reconstruction of the environment, SLAM is centred around the problem of localisation in the environment. In SLAM, the features extracted from images are stored to create a map of the environment. The map is not only used to infer the current position of the camera but it is also re-estimated due to new information added in the map. After all, the features are extracted from imperfect sensors and uncertainties about the relative positions of the features cannot be neglected. One way to express these uncertainties is to use a probabilistic formulation as it is often done for SLAM.

Since we are mostly concerned about the localisation part of the problem, the probabilistic formulation will be introduced to the reader using a SLAM example hereafter. Parallels to localisation and calibration problems treated in the next chapters are straightforward. State estimators are used in mobile robotics to estimate both intrinsic and extrinsic states. Their importance led to many efforts in the study of their limits and to improve them. We can distinguish two main categories of estimators: filters and batch optimisation methods. A nice review is proposed in [Strasdat 2010] and compares both categories applied to SfM problems. We discuss filtering and optimisation next.

Let's go back to our initial (haunted house) example and note $\mathbf{x}_t^T = \begin{bmatrix} x_t & y_t & \theta_t \end{bmatrix}$ the vector containing your pose (2D position and orientation) in the room at time t. Since the visual part and the mapping of the environment have not been used for this thesis, the formulation will take the example of a blind navigation for a better coherence with the rest of the thesis. However, the similarities make the transposition to a SLAM problem straightforward. Thus, the theory presented hereafter is inspired by SLAM courses [Stachniss 2013, Sola 2016].

1.1.2 Sequences of states and measurements

The pose of a robot is denoted by $\mathbf{x}_t$ where t denotes the time step we are considering and expressing both position and orientation relative to a reference frame. Under the flat ground assumption, the pose is a 3-components vector $\mathbf{x}_t = \begin{bmatrix} x_t & y_t & \theta_t \end{bmatrix}^T$. If we consider the exploration of a 3D environment, we get to a minimal representation of the pose containing six components. The position is described by $\mathbf{p}_t = \begin{bmatrix} x_t & y_t & z_t \end{bmatrix}^T$ while the orientation is also represented by a 3-components vector in the minimal form. One possible choice is the Euler angles $\mathbf{o}_t = \begin{bmatrix} \phi_t & \theta_t & \psi_t \end{bmatrix}^T$.

In this work, rotations will be represented by four dimension quaternion vectors introduced with more details in Appendix B. In both 2D and 3D cases, we still can consider the pose vector as a concatenation of both position and orientation vectors ($\mathbf{x}_t = \begin{bmatrix} \mathbf{p}_t^T & \mathbf{o}_t^T \end{bmatrix}^T$).

The whole trajectory of the robot from time step $t_0 = 0$ to time step T is the complete set of poses described by

$$\mathbf{x}_{0:T} = \{\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_T\} \tag{1.1}$$

While navigating in its environment, the robot may compute its odometry using embedded sensors, thus measuring the motion between time steps. This measure can be considered as an estimate of the trajectory and is usually noted $\mathbf{u}_t$ and interpreted as the control sequence from time step $t-1$ up to time step t. Thus the complete sequence of measurements from t_0 to T is given by

$$\mathbf{u}_{1:T} = \{\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_T\} \tag{1.2}$$

Under the assumption that the odometer is perfect, the relation between both poses and odometry sequences would be straightforward. Indeed, integrating the successive odometry measurements would describe the overall trajectory of the robot.

$$\mathbf{x}_1 = \mathbf{x}_0 \oplus \mathbf{x}_{0\to1} = \mathbf{x}_0 \oplus \mathbf{u}_1 \tag{1.3}$$

$$\mathbf{x}_n = \mathbf{x}_0 \oplus \sum_{i=1}^{n} \mathbf{x}_{i-1\to i} = \mathbf{x}_0 \oplus \sum_{i=1}^{N} \mathbf{u}_i \tag{1.4}$$

1.1.3 Some noise in the sensors

However, as explained in the Introduction chapter, measurements are imperfect not only due to the sensor but it may also not take into account some specific phenomena that can occur during navigation such as slipping. Fig. 1.1 illustrates the growth of uncertainty using odometry when a mobile robot has to navigate. The difference between both red and blue trajectories can be explained not only by imperfections in sensors but also by unexpected events due to the environment as for example the wheels slipping on the floor.

When considering real applications, imperfections in the sensor need to be taken into account. A popular assumption in robotics [Thrun 2005] when designing the sensor model is to consider that it is affected by white noise and thus it can be described by a Gaussian model with zero mean. Although parametric and non-parametric inference techniques were investigated for SLAM applications [Sim 2005, Eade 2006], the Gaussian distribution assumption has not only proven to be the most effective but also simplifies a lot the theory behind state estimation while it seems to represent uncertainties in an acceptable way. Thanks to the Gaussian distribution assumption, the probability density of a random variable $\mathbf{x}$ follows (A.8).

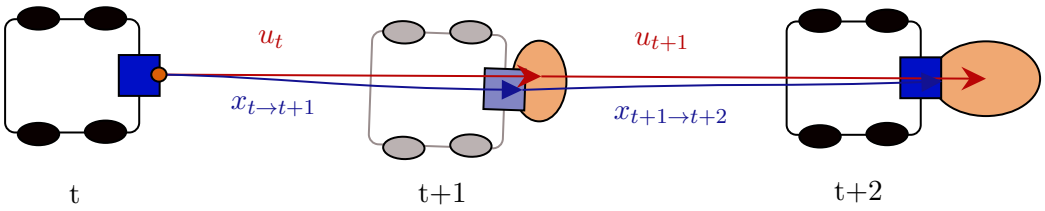


Figure 1.2: Representation of the growth of uncertainties in odometry due to sensor imperfections and the environment itself. The red trajectory is the one desired and thus the one the control sequence should lead to follow. However, due to errors and unexpected events, the real trajectory is described by the blue path. In orange ellipses here are represented the 2D covariances of the estimated states. Without any further information to better estimate the pose of the robot, this covariance only keeps growing with time.

Odometry sequences are not the only information available to most modern robots to estimate their position. We can cite for example the use of tactile sensors giving specific features and information about the environment [Kawasaki 1999, Kerpa 2003], or more generally RFID sensors that may be reading information [Agarwal 2018, Codas 2010]. That information could be anything like for example the current position of the robot or the position of the marker itself. Let us denote $\mathbf{z}_t$ the sensor measurements giving additional information related to the pose of the robot at time step t. We now consider two sources of measurements: $\mathbf{u}$ are the odometry measurements coming from the motion model that we know as a prior, $\mathbf{z}$ are the sensors in general (actually, $\mathbf{u}$ and $\mathbf{z}$ can be considered alike; however we find it convenient to distinguish them from the start). Given this information the state estimation problem is to estimate the posterior state given available information.

$$p(\mathbf{x}_t|\mathbf{u}_{1:T}, \mathbf{z}_{1:T}) \tag{1.5}$$

1.1.4 Refactoring using Markov and Bayes

The application of Bayes' Rule (A.9) then leads to

$$p(\mathbf{x}_t|\mathbf{u}_{1:t}, \mathbf{z}_{1:t}) = \eta p(\mathbf{z}_t|\mathbf{x}_t, \mathbf{u}_{1:t}, \mathbf{z}_{1:t-1}) p(\mathbf{x}_t|\mathbf{u}_{1:t}, \mathbf{z}_{1:t-1}) \tag{1.6}$$

An additional assumption made at this stage of the theory is the Markov assumption according to which the state $\mathbf{x}_t$ contains all the relevant information up to time t. In other words, given the present state of the system all the future states are independent of the past states. This assumption is the basis of the recursive estimation theory.

$$p(\mathbf{z}_t|\mathbf{x}_t, \mathbf{u}_{1:t}, \mathbf{z}_{1:t-1}) \overset{\text{Markov}}{=} p(\mathbf{z}_t|\mathbf{x}_t) \tag{1.7}$$

where $p(\mathbf{z}|\mathbf{x})$ is the measurement model. The second part of the equation (1.6) can

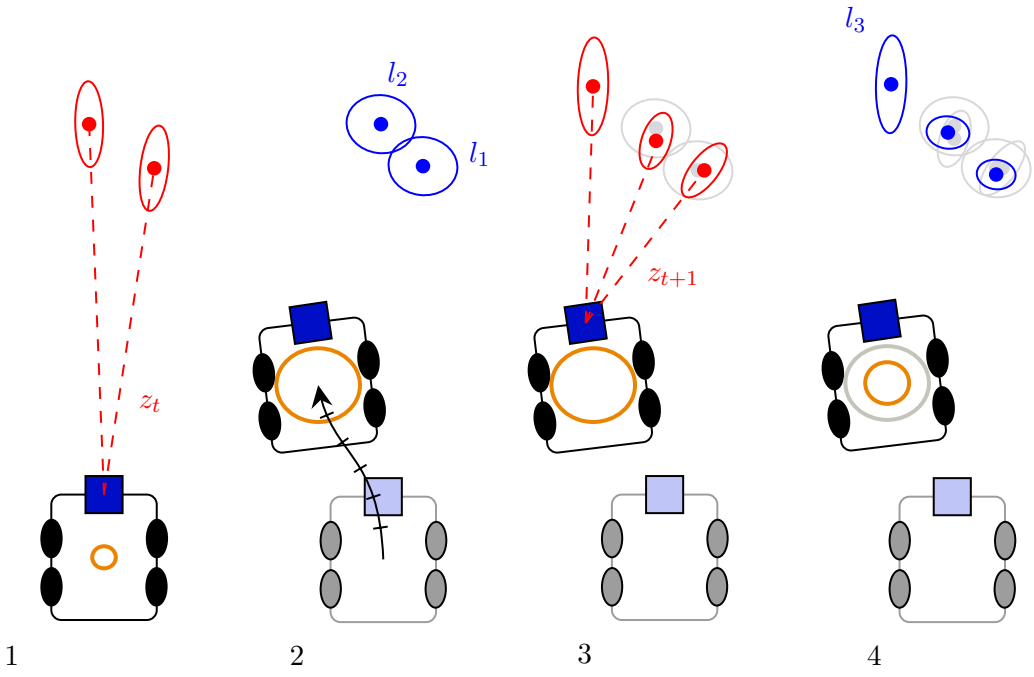


Figure 1.3: SLAM process representation. **1**: The problem is initialised. New landmarks are detected and added in the map with their respective covariance due to sensor specifications. **2**: Computation of the predicted states for the state composed of both robot's and landmarks' pose. These states are computed using the control sequence between *1* and *2* only thus resulting in an increase of pose uncertainties for the estimated states. **3**: Landmarks are detected and a data association method is used to assign the landmarks to those already present in the map. Assuming that the data association is correct for 100 % of the landmarks, the new observation is used to compute the most probable state for all the states. Newly detected landmarks are added to the map. **4**: The pose correction is associated to a decrease of uncertainties represented by the ellipses boundaries.

also be written as

$$p(\mathbf{x}_t|\mathbf{u}_{1:t}, \mathbf{z}_{1:t-1}) \overset{\text{Markov}}{=} p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t)p(\mathbf{x}_{t-1}|\mathbf{u}_{1:t}, \mathbf{z}_{1:t-1}) \tag{1.8}$$

This corresponds to the prediction at step $t-1$, or motion model. With (1.7) and (1.8) we can rewrite (1.6) as

$$p(\mathbf{x}_t|\mathbf{u}_{1:t}, \mathbf{z}_{1:t}) = \eta p(\mathbf{z}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t)p(\mathbf{x}_{t-1}|\mathbf{u}_{1:t}, \mathbf{z}_{1:t-1}) \tag{1.9}$$

Here, $\mathbf{u}_t$ can be omitted in $p(\mathbf{x}_{t-1}|\mathbf{u}_{1:t}, \mathbf{z}_{1:t-1})$ since it does not take part in estimating the state $\mathbf{x}_{t-1}$. Thus this equation becomes

$$p(\mathbf{x}_t|\mathbf{u}_{1:t}, \mathbf{z}_{1:t}) = \eta p(\mathbf{z}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t)p(\mathbf{x}_{t-1}|\mathbf{u}_{1:t-1}, \mathbf{z}_{1:t-1}) \tag{1.10}$$

One can note how the Markov assumption led to a recursive formulation for this estimation problem. Indeed, the last part of the right term of (1.9) is exactly the posterior probability expressed at one step in the past. Its expression can be written like (1.9) as

$$p(\mathbf{x}_{t-1}|\mathbf{u}_{1:t-1}, \mathbf{z}_{1:t-1}) = \eta p(\mathbf{z}_{t-1}|\mathbf{x}_{t-1})p(\mathbf{x}_{t-1}|\mathbf{x}_{t-2}, \mathbf{u}_{t-1})p(\mathbf{x}_{t-2}|\mathbf{u}_{1:t-1}, \mathbf{z}_{1:t-2}) \tag{1.11}$$

The recursive formulation in (1.9) can thus be expressed as

$$p(\mathbf{x}_t|\mathbf{u}_{1:t}, \mathbf{z}_{1:t}) = \eta p(\mathbf{z}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t)p(\mathbf{x}_{t-1}|\mathbf{u}_{1:t}, \mathbf{z}_{1:t-1}) \tag{1.12}$$

where η is a normalisation factor used to represent the denominator resulting in the application of Bayes' rule. Finally, by considering $p(\mathbf{x}_0)$ as a prior knowledge on the initial position of the robot and following the same method for all the past posterior probabilities, one can find the generalised recursive formulation for (1.9) which is

$$p(\mathbf{x}_t|\mathbf{u}_{1:t}, \mathbf{z}_{1:t}) = \eta p(\mathbf{x}_0) \prod_{i=1}^{t} p(\mathbf{z}_i|\mathbf{x}_i)p(\mathbf{x}_i|\mathbf{x}_{i-1}, \mathbf{u}_i) \tag{1.13}$$

To cite [Thrun 2005], "state estimation addresses the problem of estimating quantities from sensor data that are not directly observable, but that can be inferred". Solving the localisation and by extension the SLAM problems leads to finding the most likely (or probable) values that all the estimated states (robot states and features locations for map estimation in SLAM) would take given information provided by the sensors. This is the crucial and complex job of the estimator.

1.1.5 And the map?

The problem described above is a subset of the general SLAM problem formulation. Indeed, SLAM additionally includes the mapping part processed with fea-

tures extracted from exteroceptive sensors. The exact location of these features remains unknown and thus their positions in the environment have to be estimated (see Fig. 1.3). The map must be corrected with new estimations of features' locations. Thus, if we denote by $\mathbf{m}$ the map, it enters in the posterior formula as

$$p(\mathbf{x}_t, \mathbf{m} | \mathbf{u}_{1:T}, \mathbf{z}_{1:T}) \tag{1.14}$$

1.1.6 Should we filter or optimise?

Estimators are main components of any robotic systems. Their place for the system is just as important as the brain is to any living creature. Estimators are used not only to estimate internal and/or external states but also to realise the multi-sensor fusion. Their importance pushed the research community to put a lot of efforts in designing new estimators, study existing ones, correction some specific parts and trying to cancel limitations to get an estimator that would get better. However, at some point, two communities were formed based on the preferred philosophy that is using filtering methods or sparse optimisation. Both methodologies have been used extensively by SLAM and SfM communities.

[Azarbayejani 1995] already uses an Extended Kalman Filter (EKF), that is a recursive estimation method, to not only solve the SfM problem, that is recovering the motion and the structure of the environment, but also to correctly estimate the focal length of the camera from feature correspondences tracked through the image sequence. Although they use EKF, they also put emphasis on the fact that an equivalent solution can be found using other estimators. Closely related to this example are the work of [Oliensis 1991, Soatto 1993] that are using EKF as a smoothing filter. In the search of better performances, some work investigated the use of optimisation methods based on the minimisation of a nonlinear function [Kumar 1989, Spetsakis 1991, Weng 1993]. Making a choice between recursive estimation and sparse optimisation methods in the SfM community is still a problem nowadays. [Chiuso 2002] also uses recursive estimation method to improve SfM results due to the sub-minimal model presented in [Azarbayejani 1995]. While looking for a real-time application of the SfM problem, they chose the filtering methods because of the need to process data on-line.

[Triggs 1999] is giving an overview of different global bundle adjustment algorithms which objective is to solve the "problem of refining a visual reconstruction to produce jointly optimal structure and viewing parameter estimates". The subject is definitely complex due to all the parameters that have to be taken into account during the implementation of such algorithms with the hope to make the estimation as robust as needed and the computation as effective as possible. If we had to summarise in a single sentence, we would say that bundle adjustment aims at estimating the parameters of a large sparse problem by minimising cost functions. Further details about the optimisation-based methods are given in section 1.3. Bundle adjustment methods are often said to be extremely costly regarding the computation time. The reason may be not only because taking advantage of the

sparseness of the problem is not straightforward, but also because these methods require the inversion of linear systems whose sizes grow with the number of estimated parameters. Although there exist examples of SfM using bundle adjustment with nonlinear least square minimisation [Szeliski 1994, Shum 1999], using global bundle adjustment still remained a challenge due to the time-consuming estimation. Some authors approached a more efficient way to use bundle adjustment. [Steedly 2001] used an incremental reconstruction method, readjusting only a subset of parameters, or [Mouragnon 2009] using local bundle adjustment each time a new camera pose is added for real-time SfM.

This choice is also present in the SLAM community. [Davison 2003] uses the recursive filtering method and achieve real-time performances with specific strategies concerning the mapping part to take advantage of the sparseness of the problem and a top-down Bayesian estimation approach. More recently, [Roussillon 2011] proposed a generic and practical solution using the EKF for real-time visual SLAM on robots. Other approaches attempting to propose a SLAM method that would also be effective in large-scale navigation have been proposed [Eade 2006, Montemerlo 2007].

Some authors have also focused on using sparse optimisation methods in SLAM to achieve better accuracy than recursive estimation based ones. Efforts have also be made to reduce the computational cost of bundle adjustment estimation by reducing the complexity of the problem. Indeed [Konolige 2008] introduced the reduction of the parameter space in FrameSLAM through a marginalisation procedure. Other methods are using local maps for estimation purposes [Ni 2007, Ni 2010].

Finally, making a choice between bundle adjustment based methods or recursive filtering estimation is still today a question of trade-offs. On the one hand, recursive filter based methods are known to be faster to implement and allow real-time applications. But their main drawback as regards sparse optimisation based solutions seems to be the accuracy of the estimation. On the other hand, these last ones have the reputation to be complex to implement and to require higher computation time despite recent advances allowing to considerably reduce this last parameter. Trade-offs are also application dependent. For instance, both SfM and SLAM are subject specific trade-offs. For SfM, the problem may be summarised as to find a balance between an easier relative orientation estimation or an easier correspondence problem between features in the images. With disparate viewpoints between the images, finding the relative orientation is made easy but the features correspondence problem will be more complex. On the other hand, if the viewpoints are close enough then the correspondence problem will be easy but finding the relative orientation will be harder due to approximations and accuracy.

[Strasdat 2010, Strasdat 2012] first compare filtering and sparse optimisation methods applied to SLAM problems and underline the differences of these two approaches. The representation of both methods as given in Fig. 1.4 is a good start to understand the differences and what it implies to prefer one solution over the other one.

While some filtering methods and sparse optimisation will be presented with

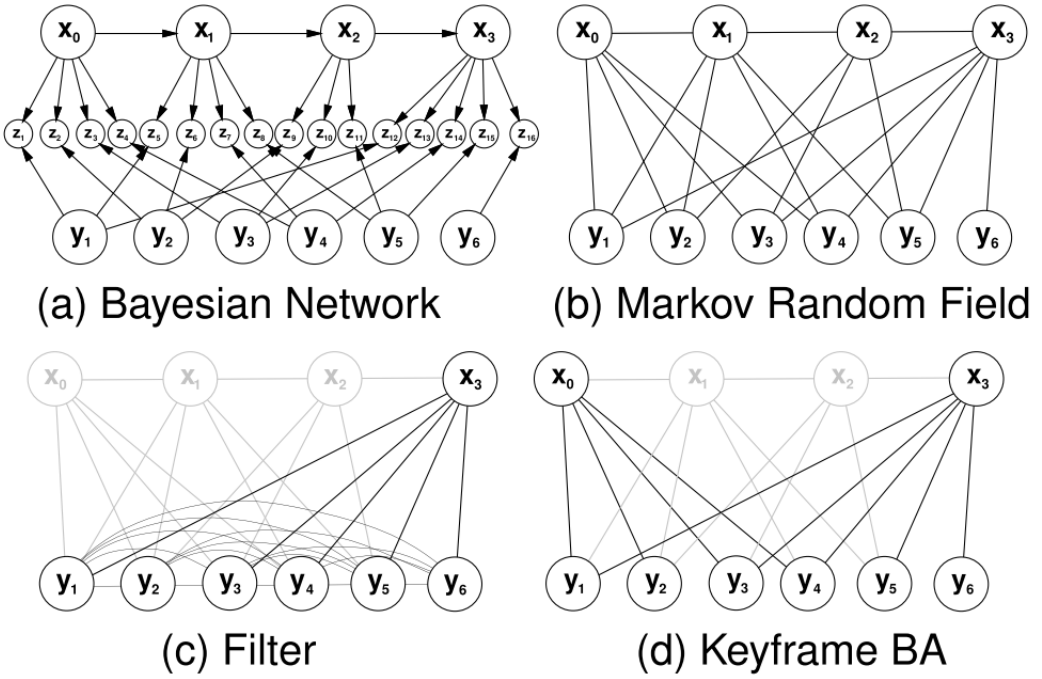


Figure 1.4: Problem representation and relation between data depending on the chosen estimation strategy [Strasdat 2010]. **(a)**: Bayesian network for a usual SLAM/SfM problem. **(b)**: Equivalent graph problem represented as a Markov random field without the measurements $\mathbf{z}_i$ **(c)**: Visualisation of the effect of marginalisation on the inference in filtering methods **(d)**: Visualisation of the effect of estimating selective states in key-frame-based bundle adjustment techniques on inference.

more details in the following sections, let us keep in mind some major points:

1. The sparseness of the problem is different in both cases. This detail has consequences on the computation of the joint probabilities and their propagation.

2. Filtering methods use the Markov assumption to marginalise out past poses and to summarise all the information gained over time with a probability distribution whereas sparse optimisation methods are using key-frames to select specific states to be estimated. Non-estimated key-frames are rather ignored than marginalised out, thus keeping the problem sparse and the optimisation relatively efficient.

3. Both methods are using approximations.

4. There exist a wide variety of methods mixing both philosophies. We could cite for example the multi-state Kalman Filter which is an attempt to overcome some limitations of filtering methods with inspiration from sparse optimisation

at the cost of more time-consuming estimations. More examples are given in subsection 1.3.4.

In the next two sections, we reformulate the estimation problem as either a filtering or an optimisation problem, by either marginalising over the past states (filtering) or extracting from the Bayesian network the underlying factor graph (optimisation). This will motivate our choice of using an optimisation approach in the rest of the thesis, but also explains how our contributions can in fact be quite directly transferred into a filtering method.

1.2 Filtering Approach

1.2.1 Bayes' Filter Algorithm

The *Bayes' filter* is the most general algorithm to compute posterior probabilities and gives a good overview of most recursive filters. The pseudo-algorithm of Bayes' filter is presented in Algorithm 1. It highlights the two steps that are at the core of the recursive estimation methods : the *prediction* and *correction* steps.

The prediction step

The prediction step aims at computing a prior density before using the measurement at time step t and right after using the control measurement $\mathbf{u}_t$. This step predicts the state $\mathbf{x}_t$ on the basis of $\mathbf{z}_{1:t-1}$ the measurement information up to time $t-1$ and the latest control measurement $\mathbf{u}_t$.

$$p(\bar{\mathbf{x}}_t|\mathbf{u}_{1:t}, \mathbf{z}_{1:t-1}) = \int \underbrace{p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t)}_{\text{state transition}} \underbrace{p(\mathbf{x}_{t-1}|\mathbf{u}_{1:t-1}, \mathbf{z}_{1:t-1})}_{\text{posterior probability at step t-1}} d\mathbf{x}_{t-1} \tag{1.15}$$

The state transition is given by the dynamics of the system. We can also easily give an expression of the posterior probability at step $t-1$ using (1.8). A way to interpret this prediction step is to say that it computes the most probable evolution for the state of the system given the previously computed posterior probability and the dynamic model of the system.

The correction step

Once the prediction is made, it is time for the system to use the last measurement $\mathbf{z}_t$ and correct the prediction with the newly provided information. This correction step, also called measurement update, is basically a product of the probability density functions describing the posterior probability computed in the prediction step (1.15) and the one associated with the measurement $\mathbf{z}_t$, $p(\mathbf{z}_t, \mathbf{x}_t)$. The computation of the new posterior probability is exactly given by (1.9). We give here the complete form of this equation.

$$p(\mathbf{x}_t|\mathbf{u}_{1:t}, \mathbf{z}_{1:t}) = \frac{p(\mathbf{z}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t)p(\mathbf{x}_{0:t-1}|\mathbf{u}_{1:t}, \mathbf{z}_{1:t-1})}{p(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t})} \tag{1.16}$$

Algorithm 1: BayesFilter($\boldsymbol{x}_{t-1}$, $\mathbf{u}_t$, $\mathbf{z}_t$)

1: **loop**
2: {Prediction($\mathbf{x}_{t-1}$, $\mathbf{u}_t$, $\mathbf{\Sigma}_{t-1}$):}
3: Get control sequence $\mathbf{u}_t$
4: $p(\bar{\mathbf{x}}_t \mid \mathbf{u}_{1:t}, \mathbf{z}_{1:t-1}) = \int p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_t) p(\mathbf{x}_{t-1} \mid \mathbf{u}_{1:t-1}, \mathbf{z}_{1:t-1}) d\mathbf{x}_{t-1}$
5: {Correction($\bar{\mathbf{x}}_t$, $\mathbf{z}_t$):}
6: Get measurements $\mathbf{z}_t$
7: $p(\mathbf{x}_t \mid \mathbf{u}_{1:t}, \mathbf{z}_{1:t}) = \eta \; p(\mathbf{z}_t \mid \mathbf{x}_t) p(\bar{\mathbf{x}}_t \mid \mathbf{u}_{1:t}, \mathbf{z}_{1:t-1})$
8: return $\mathbf{x}_t, \mathbf{\Sigma}_t$
9: **end loop**

Now that the *Bayes' Filter* is introduced, the next section will present the most popular estimators designed on the principles described above. All of them are built on top of the theory of Bayes' Filters. Variations can be introduced concerning the probabilities that have been described sooner. The reason why assumptions are made concerning the probabilities involved in Bayes' Filter is due to the impossibility to compute exacts beliefs for general problems. Thus assumptions are a way to approximate the posterior probabilities. However, most of the filters are using the Gaussian distribution to design the dynamics and the measurement models, meaning that all probability density functions can be described by (A.8). Thus, the following filters are also known as *Gaussian Filters*.

1.2.2 Kalman Filter

The Kalman Filter (KF) [Kalman 1960] is an implementation of Bayes' Filter that can be used to compute the posterior probabilities of continuous states only. In addition to both Markov and Gaussian distribution assumptions, this implementation assumes the dynamic and measurement models of the system to be linear. Mathematical translation of this new assumption gives

$$\mathbf{x}_t = \mathbf{f}(\mathbf{x}_{t-1}, \mathbf{u}_t) = \mathbf{A}_t \mathbf{x}_{t-1} + \mathbf{B}_t \mathbf{u}_t + \varepsilon_{\mathbf{x}_t} \tag{1.17}$$

where $\varepsilon_{\mathbf{x}_t}$ is the process noise expressing the uncertainty in the state transition through perturbations in the system. This process noise is defined by a Gaussian distribution ((A.8)). Similarly, the measurement model is given by the function

$$\mathbf{z}_t = \mathbf{h}(\mathbf{x}_t) = \mathbf{C}_t \mathbf{x}_t + \varepsilon_{\mathbf{z}_t} \tag{1.18}$$

where $\varepsilon_{\mathbf{z}_t}$ is the Gaussian measurement noise expressing the noise related to the sensor measurement (due to sensor specifications).

$$p(\varepsilon_{\mathbf{x}_t}) \sim \mathcal{N}(\varepsilon_{\mathbf{x}_t}, \mathbf{Q}_t) \tag{1.19}$$
$$p(\varepsilon_{\mathbf{z}_t}) \sim \mathcal{N}(\varepsilon_{\mathbf{z}_t}, \mathbf{R}_t) \tag{1.20}$$
$$p(\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_0, \mathbf{\Sigma}_0) \tag{1.21}$$

The linear evolution constraint leads to finite dimensions for matrices $\mathbf{A}_t$, $\mathbf{B}_t$ and $\mathbf{C}_t$. If we note n, m and p the dimensions of respectively $\mathbf{x}_t$, $\mathbf{u}_t$ and $\mathbf{z}_t$, then one can find that those three matrices are respectively of dimension $n \times n$, $n \times m$ and $p \times n$. We can also give an explanation of the meaning of the different matrices in this model. $\mathbf{A}_t$ describes how the state evolves from $t-1$ to t without controls or noises while $\mathbf{B}_t$ describes the changes in the control from $t-1$ to t. Finally, $\mathbf{C}_t$ can be understood as the description that allows one to map the state $\mathbf{x}_t$ to an observation $\mathbf{z}_t$ If we think about it, (1.17) relates to the state transition in (1.15) while (1.18) is related to $p(\mathbf{z}_t|\mathbf{x}_t)$ and we can express both densities as

$$p(\mathbf{x}_t|\mathbf{u}_{1:t-1}, \mathbf{z}_{1:t-1}) = \frac{1}{det(\mathbf{Q}_t)^{\frac{1}{2}}(2\pi)^{\frac{n}{2}}} \exp\left(-\frac{1}{2}(\mathbf{x}_t - \bar{\mathbf{x}}_t)^T \mathbf{Q}_t^{-1}(\mathbf{x}_t - \bar{\mathbf{x}}_t)\right) \tag{1.22}$$
$$p(\mathbf{z}_t, \mathbf{x}_t) = \frac{1}{det(\mathbf{R}_t)^{\frac{1}{2}}(2\pi)^{\frac{n}{2}}} \exp\left(-\frac{1}{2}(\mathbf{z}_t - \bar{\mathbf{z}}_t)^T \mathbf{R}_t^{-1}(\mathbf{z}_t - \bar{\mathbf{z}}_t)\right) \tag{1.23}$$

where

$$\bar{\mathbf{x}}_t \triangleq \mathbb{E}[\mathbf{x}_t] = \mathbf{A}_t\mathbf{x}_{t-1} + \mathbf{B}_t\mathbf{u}_t$$
$$\bar{\mathbf{z}}_t \triangleq \mathbb{E}[\mathbf{z}_t] = \mathbf{C}_t\mathbf{x}_t$$

the expression of the prior knowledge on the position of the robot is all that remains to be defined to be able to express (1.13). This prior is also described by a Gaussian distribution ($p(\mathbf{x}_0) \sim \mathcal{N}(\boldsymbol{\mu}_0, \mathbf{\Sigma}_0)$) due to the properties of Gaussian densities. Given all these details, we can say that the KF algorithm presented in Alg. 2 aims at computing the mean and the covariance of the states in a closed-loop of prediction-correction steps every time a new measurement and/or control sequence arrives. The interested reader is given a more detailed view of both prediction and correction steps in the following.

KF prediction step

With the Gaussian distribution assumption for probability densities, the prediction step (1.15) yields to

$$p(\bar{\mathbf{x}}_t|\mathbf{u}_{1:t}, \mathbf{z}_{1:t-1}) = \eta \int \exp\left(-\mathbf{L}_t\right) d\mathbf{x}_{t-1} \tag{1.24}$$

with

$$\mathbf{L}_t = \frac{1}{2}\left((\mathbf{x}_t - \bar{\mathbf{x}}_t)^T \mathbf{Q}_t^{-1}(\mathbf{x}_t - \bar{\mathbf{x}}_t) + (\mathbf{x}_{t-1} - \bar{\mathbf{x}}_{t-1})^T \mathbf{\Sigma}_{t-1}^{-1}(\mathbf{x}_{t-1} - \bar{\mathbf{x}}_{t-1})\right) \tag{1.25}$$

Going back to (1.17), one can show from (1.15) that

$$\hat{\mathbf{x}}_{t|t-1} \triangleq \mathbb{E}[\mathbf{x}_t|\mathbf{u}_{1:t}, \mathbf{z}_{1:t-1}] = \bar{\mathbf{x}}_t \tag{1.26}$$

$$\mathbf{\Sigma}_{t|t-1} \triangleq \mathbb{E}[(\mathbf{x}_t - \hat{\mathbf{x}}_{t|t-1})(\mathbf{x}_t - \hat{\mathbf{x}}_{t|t-1})^T] = \mathbf{A}_t \mathbf{\Sigma}_{t-1} \mathbf{A}_t^T + \mathbf{B}_t \mathbf{Q}_t \mathbf{B}_t^T \tag{1.27}$$

Although the details are not presented here, the interested reader can find the mathematical derivations leading to equations above in [Thrun 2005]. We will note here that the posterior probability at step $t-1$ is a prior probability used to predict the future mean and covariances of the state $\mathbf{x}_t$. This prior is described by a Gaussian distribution ($\mathcal{N}(\bar{\mathbf{x}}_{t-1}, \mathbf{\Sigma}_{t-1})$) and used in the state transition part. Furthermore, the state transition in itself is affected by a Gaussian noise of covariance $\mathbf{R}_t$ due to the imperfections in the control sequence so that $p(\mathbf{u}_t) \sim \mathcal{N}(\mathbf{u}_t - 0; \mathbf{Q})$. Another expression for $\mathbf{R}_t$ is $\mathbf{B}_t \mathbf{Q}_t \mathbf{B}_t^T$ under the Gaussian distribution assumption.

KF correction step

Now let's consider the correction step which probabilistic expression is given by (1.16) or alternatively by (1.9). Again, similarly to (1.24), the posterior probability $p(\mathbf{x}_t|\mathbf{u}_{1:t}, \mathbf{z}_{1:t})$ can be expressed with an exponential function as

$$p(\mathbf{x}_t|\mathbf{u}_{1:t}, \mathbf{z}_{1:t}) = \eta \exp\Big(-J_t\Big) \tag{1.28}$$

where

$$J_t = \frac{1}{2}\Big((\mathbf{z}_t - \mathbf{C}_t\mathbf{x}_t)^T \mathbf{Q}_t^{-1}(\mathbf{z}_t - \mathbf{C}_t\mathbf{x}_t) + (\mathbf{x}_t - \bar{\boldsymbol{\mu}}_t)^T \bar{\mathbf{\Sigma}}_t^{-1}(\mathbf{x}_t - \bar{\boldsymbol{\mu}}_t)\Big) \tag{1.29}$$

One can show that the covariance characterising the Gaussian distribution of the posterior probability in (1.28) is given by

$$\mathbf{\Sigma}_t = \mathbf{C}_t \ \mathbf{Q}_t^{-1} \ \mathbf{C}_t^T + \bar{\mathbf{\Sigma}}_t^{-1} \tag{1.30}$$

The correction step aims at incorporating measurement information in the system and correct the prediction. Therefore we need to be able to express how much new information is added to the system for the correction step to be complete and compute both mean and covariance of the posterior probability. This is done by using the *Kalman Gain* at time step t noted $\mathbf{K}_t$. This *Kalman Gain* can be found by minimising the first derivative of J_t and by noting that the value of $\mathbf{x}_t$ will actually be the mean $\boldsymbol{\mu}_t$ of $p(\mathbf{x}_t)$. Thus we can substitute both terms. This method leads

Algorithm 2: Kalman Filter($\boldsymbol{\mu}_{t-1}$, $\mathbf{u}_t$, $\mathbf{z}_t$, $\boldsymbol{\Sigma}_{t-1}$)

loop
  {Prediction($\boldsymbol{\mu}_{t-1}$, $\mathbf{u}_t$, $\boldsymbol{\Sigma}_{t-1}$):}
  $\bar{\boldsymbol{\mu}}_t = \mathbf{A}_t\ \boldsymbol{\mu}_{t-1} + \mathbf{B}_t\ \mathbf{u}_t$
  $\bar{\boldsymbol{\Sigma}}_t = \mathbf{A}_t\ \boldsymbol{\Sigma}_{t-1}\mathbf{A}_t^T + \mathbf{R}_t$
  {Correction($\bar{\boldsymbol{\mu}}_t$, $\mathbf{z}_t$, $\bar{\boldsymbol{\Sigma}}_t$):}
  $\mathbf{K}_t \triangleq \bar{\boldsymbol{\Sigma}}_t\ \mathbf{C}_t^T\ (\mathbf{C}_t\ \bar{\boldsymbol{\Sigma}}_t\ \mathbf{C}_t^T + \mathbf{Q}_t)^{-1}$
  $\boldsymbol{\mu}_t = \bar{\boldsymbol{\mu}}_t + \mathbf{K}_t\,(\mathbf{z}_t - \mathbf{C}_t\bar{\boldsymbol{\mu}}_t)$
  $\boldsymbol{\Sigma}_t = \bar{\boldsymbol{\Sigma}}_t(\mathbf{I} - \mathbf{K}_t\mathbf{C}_t)$
  return $\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t$
end loop

to the following expression of the corrected mean $\boldsymbol{\mu}_t$

$$\boldsymbol{\mu}_t = \underbrace{\bar{\boldsymbol{\mu}}_t}_{\text{predicted mean}} + \underbrace{\mathbf{K}_t(\mathbf{z}_t - \mathbf{C}_t\bar{\boldsymbol{\mu}}_t)}_{\text{correction}} \tag{1.31}$$

with

$$\mathbf{K}_t = \boldsymbol{\Sigma}_t\mathbf{C}_t^T\mathbf{Q}_t^{-1} \tag{1.32}$$

However, expressing $\mathbf{K}_t$ as a function of $\boldsymbol{\Sigma}_t$ is not convenient. A transformation leads us to another definition that makes the use of the gain possible in practice

$$\mathbf{K}_t \triangleq \bar{\boldsymbol{\Sigma}}_t\ \mathbf{C}_t^T\ (\mathbf{C}_t\ \bar{\boldsymbol{\Sigma}}_t\ \mathbf{C}_t^T + \mathbf{Q}_t)^{-1} \tag{1.33}$$

from (1.31) we can easily deduce a new expression of $\boldsymbol{\Sigma}_t$ expressed as a $\bar{\boldsymbol{\Sigma}}_t$ correction operation

$$\boldsymbol{\Sigma}_t = \bar{\boldsymbol{\Sigma}}_t(\mathbf{I} - \mathbf{K}_t\mathbf{C}_t) \tag{1.34}$$

From (1.31) and (1.34), an intuitive way to understand the role of the *Kalman Gain* is then to see it as a translation of how much the new measurement will be integrated in the system.

1.2.3 Extended Kalman Filter

The *Extended Kalman Filter* (EKF) aims at correcting the major drawback of KF that is the assumption of linear dynamic and observation models. Such a hard requirement makes the filter difficult to use in real systems that are subject to noise and nonlinearities. This is done by modelling both observation and dynamic of the

system with nonlinear functions so that

$$\mathbf{x}_t = \mathbf{f}(\mathbf{x}_{t-1}, \mathbf{u}_t) + \varepsilon_{\mathbf{x}_t} \tag{1.35}$$

$$\mathbf{z}_t = \mathbf{h}(\mathbf{x}_t) + \varepsilon_{\mathbf{z}_t} \tag{1.36}$$

$$p(\varepsilon_{\mathbf{x}_t}) \sim \mathcal{N}(\varepsilon_{\mathbf{x}_t} - \bar{\varepsilon}_{\mathbf{x}_t}, \mathbf{Q}_t) \tag{1.37}$$

$$p(\varepsilon_{\mathbf{z}_t}) \sim \mathcal{N}(\varepsilon_{\mathbf{z}_t} - \bar{\varepsilon}_{\mathbf{z}_t}, \mathbf{R}_t) \tag{1.38}$$

$$p(\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_0 - \bar{\mathbf{x}}_0, \mathbf{\Sigma}_0) \tag{1.39}$$

However, due to the introduction of nonlinear functions, the probability densities at stake here cannot be ensured to be Gaussian. Despite this, the EKF algorithm still considers the distributions to be Gaussian by approximations and most of the prediction-correction loop of the KF algorithm is applied.

In order to apply prediction-correction scheme of Bayes' Filter, the Gaussian distribution approximation is realised through a linearisation step on the most recent estimate we have. The best estimate is also the value the variable is the most likely to take, that is the mean of the distribution. Thus, $\mathbf{f}(.)$ and $\mathbf{g}(.)$ nonlinear functions are approximated locally by a linear function with assumed Gaussian posterior. The linearisation is realised in practice thanks to a *first-order Taylor expansion* defined as

$$\mathbf{f}(\mathbf{x}) = \mathbf{f}(\mathbf{x}_0) + \mathbf{F}_{\mathbf{x}}(\mathbf{x} - \mathbf{x}_0) \tag{1.40}$$

where $\mathbf{F}_{\mathbf{x}}$ is the Jacobian matrix defined for $\mathbf{f}(.)$ by

$$\mathbf{F}_{\mathbf{x}}(\mathbf{x} - \mathbf{x}_0) \triangleq \frac{\partial \mathbf{f}}{\partial \mathbf{x}^T}\bigg|_{\mathbf{x}_0} \tag{1.41}$$

EKF prediction step

The state prediction state begins with the linearisation of the state transition function $\mathbf{f}(.)$. This linearisation is realised around the most recent best estimate $\boldsymbol{\mu}_{t-1}$ so that

$$\mathbf{f}(\mathbf{x}_{t-1}, \mathbf{u}_t) \overset{\text{Taylor}}{=} \mathbf{f}(\boldsymbol{\mu}_{t-1}, \mathbf{u}_t) + \mathbf{F}_{\mathbf{x}}(\mathbf{x}_{t-1} - \boldsymbol{\mu}_{t-1}) \tag{1.42}$$

$$\mathbf{\Sigma}_{t|t-1} = \bar{\mathbf{\Sigma}}_t = \mathbf{F}_{\mathbf{x}} \mathbf{\Sigma}_{t-1} {\mathbf{F}_{\mathbf{x}}}^T + \mathbf{Q}_t \tag{1.43}$$

As underlined sooner, $\mathbf{Q}$ is a Gaussian approximation of the real distribution $\check{\mathbf{Q}}$ by means of the Taylor expansion.

EKF correction step

The correction step also implies the use of the linearisation procedure of the measurement function (1.36) around the best estimate for the input parameter $\mathbf{x}_t$ that

is $\bar{\boldsymbol{\mu}}_t$. Thus (1.36), (1.30), (1.46) and (1.31) become

$$\mathbf{h}(\mathbf{x}_t) \overset{\text{Taylor}}{=} \mathbf{h}(\bar{\boldsymbol{\mu}}_t) + \mathbf{H}_{\mathbf{x}}(\mathbf{x}_t - \bar{\boldsymbol{\mu}}_t) \tag{1.44}$$

$$\boldsymbol{\Sigma}_t = \mathbf{H}_{\mathbf{x}}\ \bar{\boldsymbol{\Sigma}}_t\ {\mathbf{H}_{\mathbf{x}}}^T + \mathbf{R} \tag{1.45}$$

$$\mathbf{K}_t = \boldsymbol{\Sigma}_t {\mathbf{H}_{\mathbf{x}}}^T \mathbf{Q}_t^{-1} \tag{1.46}$$

$$\boldsymbol{\mu}_t = \underbrace{\bar{\boldsymbol{\mu}}_t}_{\text{predicted mean}} + \underbrace{\mathbf{K}_t(\mathbf{z}_t - \mathbf{h}(\mathbf{x}_t))}_{\text{correction}} \tag{1.47}$$

The EKF algorithm is presented hereafter in Alg. 3. As for $\mathbf{K}_t$ in the KF algorithm (1.33), it is possible by transformation to deduce a new expression of the EKF Kalman Gain

$$\mathbf{K}_t \triangleq \bar{\boldsymbol{\Sigma}}_t\ \mathbf{H}_t^T\ (\mathbf{H}_t\ \bar{\boldsymbol{\Sigma}}_t\ \mathbf{H}_t^T + \mathbf{Q}_t)^{-1} \tag{1.48}$$

where

$$\mathbf{H}_t \triangleq \mathbf{H}_{\mathbf{x}}(\mathbf{x}_t - \bar{\boldsymbol{\mu}}_t) \tag{1.49}$$

Algorithm 3: Extended Kalman Filter($\boldsymbol{\mu}_{t-1}$, $\mathbf{u}_t$, $\mathbf{z}_t$, $\boldsymbol{\Sigma}_{t-1}$)

loop
  {Prediction($\boldsymbol{\mu}_{t-1}$, $\mathbf{u}_t$, $\boldsymbol{\Sigma}_{t-1}$):}
  $\bar{\boldsymbol{\mu}}_t = \mathbf{f}(\bar{\boldsymbol{\mu}}_t) + \mathbf{F}_{\mathbf{x}}(\bar{\boldsymbol{\mu}}_t)$
  $\bar{\boldsymbol{\Sigma}}_t = \mathbf{F}_{\mathbf{x}t}\ \boldsymbol{\Sigma}_{t-1}\ {\mathbf{F}_{\mathbf{x}t}}^T + \mathbf{R}_t$
  {Correction($\bar{\boldsymbol{\mu}}_t$, $\mathbf{z}_t$, $\bar{\boldsymbol{\Sigma}}_t$):}
  $\mathbf{K}_t \triangleq \bar{\boldsymbol{\Sigma}}_t\ \mathbf{H}_t^T\ (\mathbf{H}_t\ \bar{\boldsymbol{\Sigma}}_t\ \mathbf{H}_t^T + \mathbf{Q}_t)^{-1}$
  $\boldsymbol{\mu}_t = \bar{\boldsymbol{\mu}}_t + \mathbf{K}_t\,(\mathbf{z}_t - \mathbf{h}(\bar{\boldsymbol{\mu}}_t))$
  $\boldsymbol{\Sigma}_t = \bar{\boldsymbol{\Sigma}}_t(\mathbf{I} - \mathbf{K}_t\mathbf{H}_t)$
  return $\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t$
end loop

Despite the introduction of nonlinear functions in the system's model of the EKF, the linearisation process introduces growing errors through approximations as illustrated in Fig. 1.5. This undesired behaviour led to the development of alternative filters based on the EKF formulation *e.g.* the Robust Extended Kalman Filter [Einicke 1999, Kwon 2015], the Unscented Kalman Filter [Wan 2000, Brossard 2017] and the Particle Filter described hereafter.

1.2.4 The particle filter

The *Particle Filter* [Doucet 2001, Doucet 2000] is another method that is used to bypass the linearisation step of EKF. Besides, this method is designed to work with non-Gaussian distributions. Instead of linearisation, the *Particle Filter* uses a sequential Monte Carlo method that generates estimates of the state and innovations

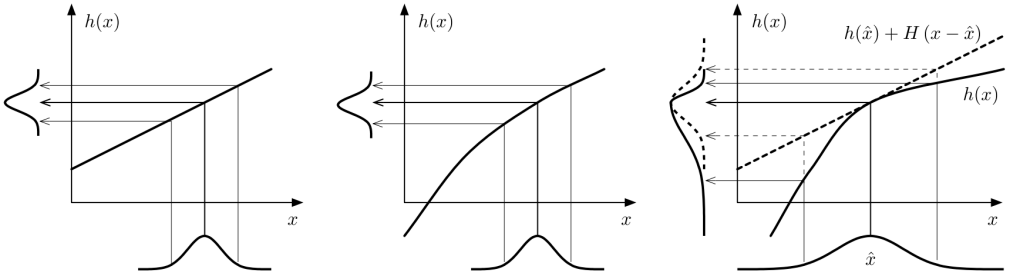


Figure 1.5: Possible introduction of errors in EKF due to linearisation and the Gaussian distribution assumption. **(Left)**: In the case of the linear system the EKF works as well as the EKF. **(Centre)**: The linearisation with Gaussian approximation works fine because the system is almost linear in this region. **(Right)**: The system is far from linear in the region of linearisation. Linearisation with the Gaussian approximation is not suited for this case. EKF will perform the operation sketched in dashed line to estimate the output while the true density is represented by the solid line. As a result, the estimates are biased and the filter diverges.(Source: [Sola 2007])

instead of deriving the analytic equations as Kalman does. This estimation with a set of *particles* that are independent random samples described by Dirac functions

$$\delta(x-a) \triangleq \lim_{\sigma \to 0} \mathcal{N}(x-a, \sigma^2) \tag{1.50}$$

The system can be described through the following equations

$$\mathbf{x}_t = \mathbf{f}(\mathbf{x}_{t-1}, \mathbf{u}_t) + \varepsilon_{\mathbf{x}_t} \tag{1.51}$$
$$\mathbf{z}_t = \mathbf{h}(\mathbf{x}_t) + \varepsilon_{\mathbf{z}_t} \tag{1.52}$$
$$p(\varepsilon_{\mathbf{x}_t}) = \text{any distribution} \tag{1.53}$$
$$p(\varepsilon_{\mathbf{z}_t}) \sim \mathcal{N}(\varepsilon_{\mathbf{z}_t} - \bar{\varepsilon}_{\mathbf{z}_t}, \mathbf{R}_t) \tag{1.54}$$
$$p(\mathbf{x}_0) = \sum_{i=1}^{N} \rho_i \delta(\mathbf{x}_0 - \mathbf{x}_0^i) \tag{1.55}$$

where the prior $p(\mathbf{x}_0)$ is described by a set of N particles.

Particle Filters prediction step

Let's consider (1.15), the aforementioned algorithms were predicting the next most likely state value given the assumption of Gaussian distribution for the posterior probabilities. This time, the prior posterior probability is a sum of N independent particles where each of them follows a randomly generated perturbation distribution so that

$$p(\mathbf{x}_t | \mathbf{z}_{0:t-1}) = \sum \rho_i \delta(\mathbf{x}_t - \mathbf{x}_i) \tag{1.56}$$

where ρ_i is the weight of each particle. At the end of this prediction stage, we don't have only a single most likely value but a set of N particles.

The prediction step consists of a sampling process (first step of any particle filter algorithm). The general particle filter algorithm is creating particles based on a proposal distribution (also called *conditional density*), which is a distribution that may not be the real probability distribution describing the random value we want to estimate but from which it is possible to sample efficiently (for example, the proposal distribution could be Gaussian). When it comes to localisation using Monte Carlo, this sampling process is done by applying the motion model (1.51). to each particle in the set (1.56) with a proposal distribution for $p(\varepsilon_{\mathbf{x}_t})$ so that we get a new set of N particles in which each sample has a state hypothesis, that is a possible prediction.

Particle Filters correction step

One may notice that the only assumption made until there is in the proposal distribution. The correction step aims at correcting this proposal using the measurement $\mathbf{z}_t$ thanks to which it is possible to compute the importance weight ρ_i of each particle. This importance weight takes into account the probability distribution we actually want to approximate and is defined by

$$\rho_t^{[i]} = \frac{\text{target distribution}}{\text{proposal distribution}} \propto \frac{p(\mathbf{x}_t^{[i]}|\mathbf{u}_{1:t}, \mathbf{z}_{1:t})}{p(\mathbf{x}_t^{[i]}|\mathbf{x}_{1:t-1}^{[i]}, \mathbf{u}_{1:t})} \propto p(\mathbf{z}_t|\mathbf{x}_t^{[i]}) \tag{1.57}$$

$$\rho_t^{[i]} = \frac{\rho(\mathbf{x}_{0:t}^{[i]})}{\sum_{j=1}^{N} \rho(\mathbf{x}_{0:t}^{(j)})} \tag{1.58}$$

where (1.58) gives the normalised version of the weights while (1.57) corrects each weight depending on how different the proposal and the real distributions are.

Before running the prediction step for the next step, the particle filter requires to resample the N particles given the current set of samples with replacement. There exists different methods for resampling [Douc 2005] but we will describe here the basic *Sequential Monte Carlo (SMC)* approach. This resampling step creates a new set that has as many particles as in the previous normalised one. The new particles are distributed according to the posterior probability $p(\mathbf{z}_t|\mathbf{x}_t)$ also meaning that the higher the weight of a particle the more likely it is to draw a particle in the corresponding area, and they all get a new weight $\frac{1}{N}$. This operation allows to keep a fixed size of particles and to eliminate the less likely possibilities that explain the trajectory of the robot, thus keeping the computational cost under some limits.

Particle Filter algorithms have been used extensively in robotics. One can find interesting use based on the particle filter in SLAM ([Grisetti 2007a, Montemerlo 2007, Dissanayake 2000]), or to track moving entities [Montemerlo 2002, Schulz 2001, Germa 2010]. Despite the major advantages of these methods, they also have limitations:

Algorithm 4: Particle Filter(Π_{t-1}, $\mathbf{u}_t$, $\mathbf{z}_t$)

loop
  {Prediction(Π_{t-1}, $\mathbf{u}_t$):}
  clean set: $\bar{\Pi}_t = \Pi_t = \varnothing$
  for $j = 1$ to N do **do**
    sample: $\mathbf{x}_t^{[j]}$ given $p(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{[j]}, \mathbf{u}_t)$
    compute particle weight: $w_t^{[j]} = p(\mathbf{z}_t \mid \mathbf{x}_t^{[j]})$
    add particle to the set: $\bar{\Pi}_t = \bar{\Pi}_t + < \mathbf{x}_t^{[j]}, w_t^{[j]} >$
  end for
  {Correction($\bar{\boldsymbol{\mu}}_t$, $\mathbf{z}_t$, $\bar{\boldsymbol{\Sigma}}_t$):}
  for $j = 1$ to N do **do**
    correct weight with observation: $\rho_t^{[j]} \propto p(\mathbf{z}_t | \mathbf{x}_t^{[j]})$
    draw particle j with probability $\propto \rho_t^{[j]}$
  end for
  Resample Π_t with N new particles
  return Π_t
end loop

- There is no assumption made on the distribution of the motion model.
- As a consequence of the previous point, there is no linearisation step.
- The performance depends on how many particles are used. The more particles are used better the accuracy will be.

The filtering methods presented in these sections highlight the recursive concept that is used to reduce the computational cost of these estimation techniques. Despite this can be a major advantage when considering real-time applications, it comes with drawbacks whose specificities will depend on the filter itself.

1.3 Online Batch Optimisation Based Approach

The optimal estimation approach relying on sparse optimisation techniques is based on a more general approach for data fusion and estimation. In particular, last decades saw the emergence of graph-based methods especially in the vision community since SLAM and SfM problems can be posed in terms of inference on a graph. Such methods require the use of two main blocs:

- the *front-end* is the problem building block taking care of all the data processing part. This part of the system is establishing the relationship between the sensor data. It is not only responsible for transforming data so that it can be used in an optimisation process but also takes care of the data association, sensor fusion and building the graph itself. This graph is then translated into a cost-function.

- the *back-end* part optimises the problem built by the *front-end* and extremises the cost-function. It is a complex task requiring a good mathematical understanding of various methods to choose the best suited for one application or another. State-of-the art methods often use iterative methods to solve the problem.

This division between front and back ends is indeed a nice property. It separates the expert knowledge about the estimation problem, written as a knowledge graph in the front end, from the agnostic solver, in the back end, which often requires less expert knowledge about the robotic problem, but strong skills in mathematics. Let's first see how this separation is implemented in the literature. We will then continue our tutorial by describing the methods behind both ends.

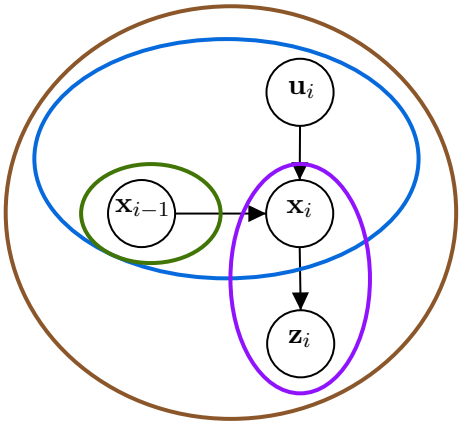
1.3.1 Back and front-ends in the literature

Every existing SLAM solution comes with its own *front-end* motivated by choices of features detection, outliers rejection, data association methods, etc. In [Endres 2012] the authors proposed their own front-end due to the novelty of their feature localisation and localisation approach. [Goncalves 2005] introduced a system to select unique and recognisable landmarks for pose estimation. [Olson 2008] designed a specific front-end in which the outliers rejection is based on spectral clustering. [Thrun 2006] relies on factorisation to reduce the dimensionality of the problem before solving. Relying on the information provided by the front-end, the accuracy and the efficiency of the back-end is also crucial. There exists a variety of back-end solutions implementing various strategies for a more efficient and accurate problem solving. Depending on the provided information, the back-end may fall into a local minimum or even diverge during optimisation. Lu and Milios in [Lu 1997] were the first to propose the optimisation-based approach for the SLAM problem. Since this work, new examples of back-ends have been proposed by the SLAM community. [Olson 2006] used a variant of stochastic Gradient Descent (SGD) to propose a back-end while improving the robustness to wrong initial guesses. This work was further improved to propose TORO [Grisetti 2007b] using a tree parameterisation. Examples of popular recent back-ends are also $\sqrt{\text{SAM}}$ [Dellaert 2006], iSAM [Kaess 2008b], iSAM2 [Kaess 2012] and g2o [Kümmerle 2011].

As we focused on the front-end part, it will be described with more details hereafter. The back-end will also be introduced for sake of completeness.

1.3.2 Graph-Based optimisation

Graphical model is now a common tool used to describe the complex structure of a problem so that one can intuitively understand the relationship between the variables playing a role in the estimation process. This graph is converted into a cost function that will be minimised during the optimisation process. Graphical models are commonly used in *smoothing* methods, as opposed to filtering estimation methods. In SLAM community, the graphical model abstractly describes the process of



$$p\left(\mathbf{x}_{i-1},\ \mathbf{x}_i \mid \mathbf{u}_i,\ \mathbf{z}_i\right) \ = \ p(\mathbf{x}_{i-1})p\left(\mathbf{x}_i \mid \mathbf{x}_{i-1},\ \mathbf{u}_i\right)\ p\left(\mathbf{z}_i \mid \mathbf{x}_i\right)$$

Figure 1.6: Representation of the DBN focusing on the relationship between state $\mathbf{x}_i$ (the current state to estimate) and the variables used for this purpose : $\mathbf{x}_{i-1}$ (prior), $\mathbf{u}_i$ (control sequence) and $\mathbf{z}_i$ (measurement). As a consequence this DBN is a representation of the joint probability $p(\mathbf{x}_{i-1}, \mathbf{x}_i|\mathbf{u}_i, \mathbf{z}_i)$. In this case, the prior is supposed to be independent from any other variable. This joint probability is a sub-part of the general equation given by (1.13).

the *front-end*, but also very literally guides the way it is implemented, in particular defining the way knowledge are stored in memory.

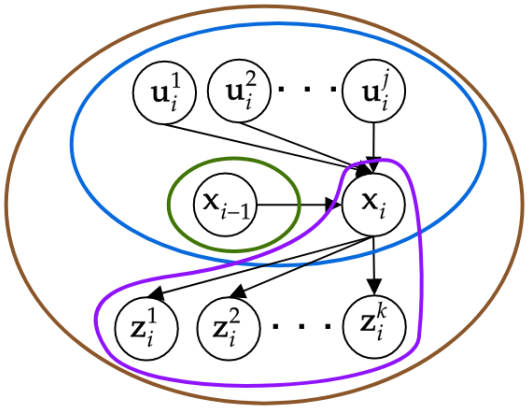
To introduce the graph formulation, we first need to introduce the Bayesian representation also known as *Dynamic Bayes Network* (DBN) before moving to the *Factor Graph* representation that we used in this work. For a detailed introduction to probabilistic graphical models, the interested reader can find explanations in [Koller 2009].

1.3.2.1 The Dynamic Bayes Network (DBN)

A DBN is a directed graph in which the variables are represented by nodes with their conditional dependencies represented by oriented edges. The dependency is represented with an arrow so that $A \rightarrow B$ means that the variable B depends on A. A graphical representation of (1.13) with a DBN in the case of a robot having only one sensor providing control sequences $\mathbf{u}$ and another sensor for measurements $\mathbf{z}$ is given in Fig. 1.6.

This representation can easily be adapted to a multi-sensor situation for both control sequences and measurements. To do this we keep in mind that the overall joint probability is obtained by the product of all the conditional dependencies. We can notice here that the prior is an independent variable. Therefore one can construct the DBN of a system carrying J controllers and K measurement sensors (see Fig. 1.7).

The equivalent graph in the case of a longer trajectory is straightforward



$$\prod_{j=1}^{J} p\left(\mathbf{x}_i \mid \mathbf{x}_{i-1}, \mathbf{u}_i^j\right)$$

$$\prod_{k=1}^{K} p\left(\mathbf{z}_i^k \mid \mathbf{x}_i\right)$$

$$p(\mathbf{x}, \mathbf{u}, \mathbf{z}) = p\left(\mathbf{x}_{i-1}, \mathbf{x}_i \mid \mathbf{u}_i^{j=1:J}, \mathbf{z}_i^{k=1:K}\right)$$
$$= p(\mathbf{x}_{i-1}) \prod_{j=1}^{J} p\left(\mathbf{x}_i \mid \mathbf{x}_{i-1}, \mathbf{u}_i^j\right) \prod_{k=1}^{K} p\left(\mathbf{z}_i^k \mid \mathbf{x}_i\right)$$

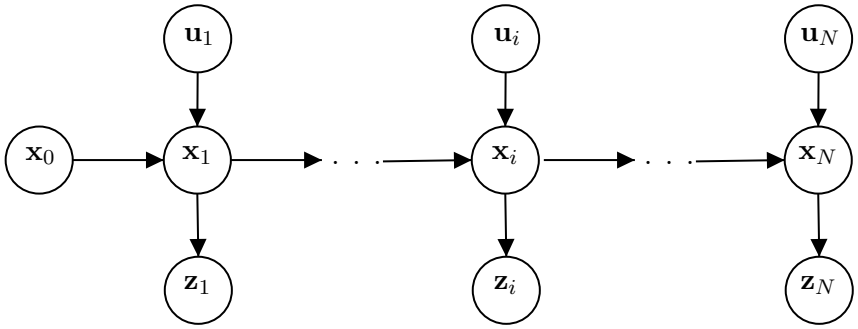
Figure 1.7: Representation of the DBN focusing on the multi-sensor aspect. The state $\mathbf{x}_i$ is not only dependent of the prior $\mathbf{x}_{i-1}$ (prior), but also on J control sequences $\mathbf{u}_i^{j=1:J}$ and K measurements $\mathbf{z}_i^{k=1:K}$. As a consequence this DBN is a representation of the joint probability $p(\mathbf{x}_{i-1}, \mathbf{x}_i | \mathbf{u}_i^{j=1:J}, \mathbf{z}_i^{k=1:K})$. In this case, the prior is supposed to be independent from any other variable.

(see Fig. 1.8). The graph for a multi-sensor case on a complete trajectory can be easily deduced from figures Fig. 1.7 and Fig. 1.8.

DBN represents the knowledge relations. For defining the optimisation problem, another graph, the *Factor Graph*, is extracted from the DBN. We describe it now.

1.3.2.2 The Factor Graphs

Factor graphs are bipartite and use two different representations to highlight the relationship between variables. Nodes represent state variables that are estimated while factors represent the relation between those states linked by edges. Whether it is in Fig. 1.6, 1.8 or 1.7, the joint probability is expressed as a product of factors of the type $p(\mathbf{x}_i | \mathbf{x}_{i-1}, \mathbf{u}_i)$. This allows the factorisation of all the conditional probabilities that imply two or more specific states. Indeed let us consider the following models that are assuming Gaussian distribution noises:



$$p(\mathbf{x}_{i=0:N}|\ \mathbf{u}_{1:N},\ \mathbf{z}_{1:N}) \ = \ p(\mathbf{x}_0)\prod_{i=1}^{N} p\left(\mathbf{x}_i \mid \mathbf{x}_{i-1},\ \mathbf{u}_i\right) p(\mathbf{z}_i \mid \mathbf{x}_i)$$

Figure 1.8: Representation of the DBN on a full trajectory. Here we made the assumption that each state except the prior has a dependency link with only one control sequence and one measurement. As a consequence this DBN is a representation of the joint probability $p(\mathbf{x}_{0:N}|\mathbf{u}_{1:N}, \mathbf{z}_{1:N})$ which is given by the product of every single joint probability as described in Fig. 1.6. The most general graph implying a different number of control sequences and measurements by states is straightforwardly obtained from this figure and Fig. 1.7.

$$\mathbf{x}_i = f_i(\mathbf{x}_{i-1}, \mathbf{u}_{i-1}) + \varepsilon_{\mathbf{x}_i} \tag{1.59}$$
$$\varepsilon_{\mathbf{x}_i} \sim \mathcal{N}(0, \mathbf{\Sigma}_{\mathbf{x}i}) \tag{1.60}$$
$$\mathbf{z}_i = h_i(\mathbf{x}_i) + \varepsilon_{\mathbf{z}_i} \tag{1.61}$$
$$\varepsilon_{\mathbf{z}_i} \sim \mathcal{N}(0, \mathbf{\Sigma}_{\mathbf{z}i}) \tag{1.62}$$

This set of equations yields to

$$\mathbf{e}(\mathbf{x}_{i-1}, \mathbf{x}_i) = \mathbf{x}_i - f_i(\mathbf{x}_{i-1}, \mathbf{u}_{i-1}) = \varepsilon_{\mathbf{x}_i} \tag{1.63}$$
$$\mathbf{e}(\mathbf{x}_i, \mathbf{z}_i) = \mathbf{z}_i - h_i(\mathbf{x}_i) = \varepsilon_{\mathbf{z}_i} \tag{1.64}$$

At this point, the Gaussian distribution assumption (see (A.7)) allows one to express the distributions in the general form

$$p(.|.) \propto exp(-\frac{1}{2}\mathbf{e}^T\mathbf{\Sigma}^{-1}\mathbf{e}) \tag{1.65}$$

where $\mathbf{e}$ refers to either (1.63) or (1.64) depending on the context. Similarly, $\mathbf{\Sigma}$ refers to either $\mathbf{\Sigma}_{\mathbf{x}i}$ or $\mathbf{\Sigma}_{\mathbf{z}i}$. $\mathbf{\Sigma}^{-1}$ is referred to as the *information matrix* and ϕ is called *factor*. We will use ϕ to denote a *factor* so that

$$\phi = \mathbf{e}^T\mathbf{\Sigma}^{-1}\mathbf{e} \tag{1.66}$$

A factor ϕ has the property to contain all the conditional dependencies related to two or more specific states by factorising all the factors into a single one as illustrated by Fig. 1.9. In this example we have the following set of probability distributions:

$$p(\mathbf{x}_1|\mathbf{x}_0, \mathbf{u}_1^1) \propto exp(-\frac{1}{2}\mathbf{e}_1^T {\mathbf{\Sigma}_{\mathbf{x}_1}}^{-1}\mathbf{e}_1)$$
$$p(\mathbf{x}_1|\mathbf{x}_0, \mathbf{u}_1^2) \propto exp(-\frac{1}{2}\mathbf{e}_2^T {\mathbf{\Sigma}_{\mathbf{x}_2}}^{-1}\mathbf{e}_2)$$
$$p(\mathbf{z}_1|\mathbf{x}_1, \mathbf{l}_1) \propto exp(-\frac{1}{2}\mathbf{e}_3^T {\mathbf{\Sigma}_{\mathbf{z}}}^{-1}\mathbf{e}_3)$$

The $\mathbf{u}_1$ factor of Fig. 1.9b factorising the first two probabilities so that

$$\begin{aligned} p(\mathbf{x}_1|\mathbf{x}_0, \mathbf{u}_1) &= p(\mathbf{x}_1|\mathbf{x}_0, \mathbf{u}_1^1)\ p(\mathbf{x}_1|\mathbf{x}_0, \mathbf{u}_1^2) \\ &\propto exp\bigg(-\frac{1}{2}(\mathbf{e}_1^T\ {\mathbf{\Sigma}_{\mathbf{x}_1}}^{-1}\ \mathbf{e}_1 + \mathbf{e}_2^T\ {\mathbf{\Sigma}_{\mathbf{x}_2}}^{-1}\ \mathbf{e}_2)\bigg) = exp(\phi_1 + \phi_2) \end{aligned}$$

As one can see, the Factor Graph allows to efficiently represent the relationship between the variables and thus have a better understanding of the problem. If we consider K to be the number of different factors ϕ in the graph, then the overall joint probability is given by

$$p(\mathbf{x}, \mathbf{z}) = \prod_{k=1}^{K} exp(\phi_k) \propto \prod_{k=1}^{K} exp(-\frac{1}{2}\ \mathbf{e}_k^T\ \mathbf{\Sigma}_k^{-1}\ \mathbf{e}_k) \tag{1.67}$$

In graph-based optimisation, the joint probability described by (1.67) then needs to be maximised which is equivalent to minimising the negative log-likelihood of this same function so that

$$-log\bigg(p(\mathbf{x}, \mathbf{z})\bigg) = \sum_{k=1}^{K} \phi_k \approx \sum_{k=1}^{K}\ \mathbf{e}_k^T\ \mathbf{\Sigma}_k^{-1}\ \mathbf{e}_k \tag{1.68}$$

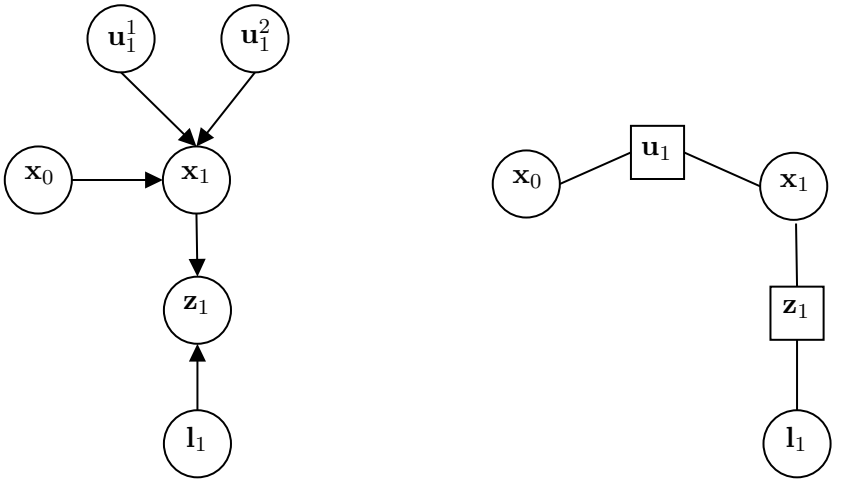
The problem as formulated here can be solved using nonlinear least square optimisation that will be introduced in the section coming hereafter.

1.3.3 Nonlinear least square optimisation

Solving the graph is in practice done by iterative nonlinear optimisation. An equivalence to (1.68) is given by

$$\mathbf{F}(\mathbf{x}) = \sum_{<i,j> \in \mathcal{C}} \phi_{ij}\ \approx \sum_{<i,j> \in \mathcal{C}} \mathbf{e}(\mathbf{x}_i, \mathbf{x}_j)^T\ \mathbf{\Omega}_{ij}\ \mathbf{e}(\mathbf{x}_i, \mathbf{x}_j) \tag{1.69}$$

where $\mathcal{C}$ is the set of all pairs of indices $< i, j >$ for which a constraint is defined in the graph and $\mathbf{\Omega}_{ij}$ is the information matrix so that $\mathbf{\Omega}_{ij} = \mathbf{\Sigma}_{ij}^{-1}$. The purpose of the optimisation and thus of solving the problem in a maximum likelihood approach



(a) DBN version (b) Factor Graph version

Figure 1.9: Factor graph and DBN representations of the same problem. We take an example derived from SLAM problems. $\mathbf{l}_1$ is a landmark that is measured by a sensor through $\mathbf{z}_1$ and which pose must be estimated. Two control sequences are used to estimate the transition from $\mathbf{x}_{i-1}$ to $\mathbf{x}_i$ to be used as an example. **Left**: DBN representation of the corresponding problem. **Right**: Factor graph representation of the same problem. Factors corresponding to $\mathbf{u}_1^1$ and $\mathbf{u}_1^2$ are encapsulated in a single factor $\mathbf{u}_1$.

is to find the configuration of states $\mathbf{x}^*$ such that the negative log-likelihood as expressed by (1.69) is minimised. Another way to say this is that the optimisation process aims at solving the following equation:

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}}\ \mathbf{F}(\mathbf{x}) \tag{1.70}$$

Keeping the objective in mind, the nonlinear least square methods aim at reducing iteratively the residual defined by the sum of the factors ϕ_k. The minimisation first involves approximating $\mathbf{F}(\mathbf{x})$ around an estimate $\check{\mathbf{x}}$ and to find a step $\Delta\mathbf{x}$ to approach the optimal configuration $\mathbf{x}^*$ while minimising the cost function. This step is used to update the estimate so that $\check{\mathbf{x}} \leftarrow \check{\mathbf{x}} + \Delta\mathbf{x}$ and approximate $\mathbf{F}(\mathbf{x})$ once again to repeat this process until convergence. By noting $\mathbf{x}_n$ and $\Delta\mathbf{x}_n$ with $n \in \mathbb{N}$ respectively the state estimate and step at iteration n, this iterative process can be described by

$$\mathbf{x}_n = \mathbf{x}_{n-1} + \Delta\mathbf{x}_n, \lim_{n \to \infty} \mathbf{x}_n = \mathbf{x}^* \tag{1.71}$$

Various iterative methods for nonlinear least square optimisation will be presented below starting with the *Gradient Descent* method. We will then introduce the *Gauss-Newton* and *Levenberg-Marquardt* methods. The introduction to the methods presented hereafter are taken from [Sola 2016].

1.3.3.1 The Gradient Descent method

The *Gradient Descent* method, also called *steepest descent* method in the literature is a minimisation method aiming at updating the state configuration in the direction given by the opposite of the gradient of the cost function $\nabla\mathbf{F}$. The first order Taylor expansion of the cost function around the current state estimate $\breve{\mathbf{x}}$ gives

$$\mathbf{F}(\breve{\mathbf{x}} + \Delta\mathbf{x}) \overset{\text{Taylor}}{=} \mathbf{F}(\breve{\mathbf{x}}) + \nabla\mathbf{F}\ \Delta\mathbf{x} \tag{1.72}$$

with

$$\nabla\mathbf{F} \triangleq \frac{\partial\mathbf{F}}{\partial\breve{\mathbf{x}}}\bigg|_{\breve{\mathbf{x}}} \tag{1.73}$$

By definition of the gradient, $\nabla\mathbf{F}$ gives the direction of the steepest ascend. Hence by going in the opposite direction that one can think of as the downhill direction, the estimation should converge towards the minimum. Therefore the optimum step is computed as

$$\Delta\mathbf{x}^*_{\text{GD}} = -\alpha\nabla\mathbf{F}^T \tag{1.74}$$

where the parameter α is used to determine the length of the step in the steepest descent direction which also has to be computed, *e.g.* by dichotomy. The search directions computed by this method are completely independent from one iteration to another. Fixing a priori α results in poor convergence rate. This behaviour can be fixed by changing the value of the parameter α with line search method. Yet, even with the optimal α (which is typically too costly to compute) the rate of convergence is linear (*i.e.* super slow). Newton-based methods would be better alternatives in particular with better rates (at least in the convex basin).

1.3.3.2 The Newton method

In the Newton method, the cost function, also called objective function, is approximated by its second order Taylor expansion around the current estimate $\breve{\mathbf{x}}$

$$\mathbf{F}(\breve{\mathbf{x}} + \Delta\mathbf{x}) \overset{\text{Taylor}}{=} \mathbf{F}(\breve{\mathbf{x}}) + \nabla\mathbf{F}\ \Delta\mathbf{x} + \frac{1}{2}\ \Delta\mathbf{x}^T\ \mathbf{H}_\mathbf{F}\ \Delta\mathbf{x} \tag{1.75}$$

where

$$\mathbf{H}_\mathbf{F} = \nabla^2\mathbf{F} \tag{1.76}$$

The function given in (1.75) is a paraboloid from which we want to find the minimum to find the optimum step $\Delta\mathbf{x}^*$. This search for the minimum implies to differentiate (1.75) and find the step so that

$$\nabla\mathbf{F}^T + \mathbf{H}_\mathbf{F}\ \Delta\mathbf{x}^* = 0 \tag{1.77}$$

yielding to a second-order step when $\mathbf{H_F} > 0$ defined by

$$\Delta\mathbf{x}^* = -{\mathbf{H_F}}^{-1}\ \nabla\mathbf{F}^T \tag{1.78}$$

This method can show quadratic (*i.e.* fast and effective) convergence when the initial guess is close enough to the solution. Like the gradient descent, Newton method is local, *i.e.* would be trapped in the first local minimum, however it is also more sensitive than the gradient descent, in particular requesting the cost function to be strictly convex. If not sufficiently convex, some eigen values of H will collapse, leading to improper numerical behaviours. A first situation illustrating this undesired behaviour is the one where the curvature of the cost function is too low, implying that the Hessian $\mathbf{H_F}$ will get too small and thus its inverse will be bigger than desired. The second situation is involving the cases when $\mathbf{H_F}$ is negative. This can happen in concave zones and then the step $\Delta\mathbf{x}$ goes to the direction opposite to the minimum, thus increasing the cost function. To prevent unacceptable effects when some eigen values become non-positive, the hessian would be regularised, *i.e.* a small positive value is added to the hessian to guarantee its strict positivity. Before discussing regularisation let's first mention why in our case we at least have non-negative values.

1.3.3.3 The Gauss-Newton method

The Gauss-Newton method can be applied to objective function that square function of the error $\mathbf{e}(\mathbf{x})$ so that

$$\mathbf{F}(\mathbf{x}) = \frac{1}{2}\mathbf{e}(\mathbf{x})^T\mathbf{\Omega}\mathbf{e}(\mathbf{x}) \tag{1.79}$$

where $\mathbf{\Omega}$ is a symmetric positive-definite matrix. This method is similar to Newton's method excepted that it does not require the second derivatives to compute the optimum step as we will see now. The gradient vector $\nabla\mathbf{F}$ and the Hessian matrix $\mathbf{H_F}$ are defined from (1.79) as

$$\nabla\mathbf{F} = \left.\frac{\partial\mathbf{F}}{\partial\mathbf{x}}\right|_{\check{\mathbf{x}}} = \left.\left(\mathbf{e}^T\Omega\frac{\partial\mathbf{e}}{\partial\mathbf{x}}\right)\right|_{\check{\mathbf{x}}} = \check{\mathbf{e}}^T\Omega\ \mathbf{J} \tag{1.80}$$

$$\mathbf{H_F} = \left.\frac{\partial^2\mathbf{F}}{\partial\mathbf{x}^2}\right|_{\check{\mathbf{x}}} = \left.\left(\frac{\partial\mathbf{e}}{\partial\mathbf{x}}^T\Omega\frac{\partial\mathbf{e}}{\partial\mathbf{x}} + \mathbf{e}^T\Omega\frac{\partial^2\mathbf{e}}{\partial\mathbf{x}^2}\right)\right|_{\check{\mathbf{x}}} = \mathbf{J}^T\mathbf{\Omega}\ \mathbf{J} + \check{\mathbf{e}}^T\mathbf{\Omega}\ \mathcal{H} \tag{1.81}$$

where $\check{\mathbf{e}}$, $\mathbf{J}$ and $\mathcal{H}$ are respectively the column error vector, the jacobian matrix of first derivatives and the Hessian tensor of second derivative around current state

estimate $\check{\mathbf{x}}$ so that

$$\check{\mathbf{e}} \triangleq \mathbf{e}(\check{\mathbf{x}}) \tag{1.82}$$

$$\mathbf{J} \triangleq \left.\frac{\partial \mathbf{e}}{\partial \mathbf{x}}\right|_{\check{\mathbf{x}}} \tag{1.83}$$

$$\mathcal{H} \triangleq \left.\frac{\partial^2 \mathbf{e}}{\partial \mathbf{x}^2}\right|_{\check{\mathbf{x}}} \tag{1.84}$$

Supposing $\check{\mathbf{e}}$ being small and $\mathbf{J}$ to vary slowly (hence the Hessian is small as well), we can neglect the second order term in (1.81). Thus, we do not need to compute the Hessian tensor $\mathcal{H}$ to approximate the Hessian matrix $\mathbf{H_F}$ as

$$\mathbf{H_F} \approx \mathbf{J}^T \mathbf{\Omega}\ \mathbf{J} \tag{1.85}$$

By replacing those terms in (1.75), we get the following approximation of the cost function defined by (1.79),

$$\mathbf{F}(\check{\mathbf{x}} + \Delta\mathbf{x}) \approx \frac{1}{2}\ \check{\mathbf{e}}^T \mathbf{\Omega}\ \check{\mathbf{e}} + \check{\mathbf{e}}^T \mathbf{\Omega}\ \mathbf{J}\ \Delta\mathbf{x} + \frac{1}{2}\ \Delta\mathbf{x}^T\ \mathbf{J}^T\ \mathbf{\Omega}\ \mathbf{J}\ \Delta\mathbf{x} \tag{1.86}$$

And again by substituting the terms in (1.78) by their definition, we get the following expression of the optimum step in the Gauss-Newton method

$$\Delta\mathbf{x}^*_{\mathrm{GN}} = -(\mathbf{J}^T \mathbf{\Omega}\ \mathbf{J})^{-1}\ (\check{\mathbf{e}}^T \Omega\ \mathbf{J})^T = -(\mathbf{J}^T \mathbf{\Omega}\ \mathbf{J})^{-1}\ \mathbf{J}^T\ \Omega\ \check{\mathbf{e}} \tag{1.87}$$

where the matrix

$$\mathbf{J}^{\#\mathbf{\Omega}} \triangleq -(\mathbf{J}^T \mathbf{\Omega}\ \mathbf{J})^{-1}\ \mathbf{J}^T\ \Omega \tag{1.88}$$

is the *left weighted generalised inverse* of $\mathbf{J}$, leading to

$$\Delta\mathbf{x}^*_{\mathrm{GN}} = \mathbf{J}^{\#\mathbf{\Omega}}\ \check{\mathbf{e}} \tag{1.89}$$

This descent step does not imply the computation of second derivatives. Moreover it induces super-linear convergence rate, *i.e.* nearly as good as plain Newton (and much better than Gradient Descent) when computing the optimum step as defined here.The weighted inverse $\mathbf{J}^{\#\mathbf{\Omega}}$ should not be evaluated explicitly. The linear system should be solved using methods designed to reduce the complexity of the problem such as the *QR* or the *Cholesky* factorisation methods. These factorisation methods will not be discussed here but the reader will find plenty of resources in the literature ([Golub 2012, Sola 2016, Guennebaud 2010]) for details about QR and Cholesky methods.

1.3.3.4 The Levenberg-Marquardt method

Both Newton and Gauss-Newton methods are valid when the starting point is close enough to the solution, *i.e.* in the same convex basin of $\mathbf{F}$. If this condition is not

satisfied then the computed step length $\Delta\mathbf{x}$ may lead the estimation into a local minimum due to a too low curvature of the approximated paraboloid given by the approximation of the cost function. The idea behind the *Levenberg-Marquardt* is to act more like the Gradient Descent algorithm when the current state estimate is far from the minimum and take advantage of the Newton-based methods convergence as we approach this minimum. To get to this behaviour, the first idea is to change the optimum step computation as given in (1.78) so that

$$\Delta\mathbf{x}^*_{\mathrm{L}} = -\alpha(\mathbf{H_F} + \mu\mathbf{I})^{-1}\ \nabla\mathbf{F}^T \tag{1.90}$$

where α can be used to change the step length and μ is the parameter used to give a Gradient Descent like behaviour. Indeed, when the curvature of the paraboloid is too low, the Hessian approximated as $\mathbf{H_F} = \mathbf{J}^T\mathbf{\Omega}\ \mathbf{J}$ is very small and thus if μ is large enough then the Hessian can be neglected and the optimum step is given by the gradient $\nabla\mathbf{F}^T$. Small values of the parameter μ result in a Gauss-Newton update. μ is usually initialised to be large, resulting in small steps for the first updates in the direction given by the gradient descent algorithm. If at some point the computed optimum step results in a higher cost, then this parameter is increased. Otherwise μ is decreased as the solution improves to approach the behaviour of the Gauss-Newton method and get a faster convergence towards the local minimum [Lourakis 2005]. This optimum step is improved by Marquardt's idea [Marquardt 1963] to replace the identity term $\mathbf{I}$ by the diagonal of the approximated Hessian.

$$\Delta\mathbf{x}^*_{\mathrm{LM}} = -\alpha(\mathbf{H_F} + \mu\ \mathrm{diag}(\mathbf{H_F}))^{-1}\ \nabla\mathbf{F}^T \tag{1.91}$$

so that the damping of the approximated Hessian matrix affects each direction of the state differently depending on the curvature of the cost function along that direction.

1.3.4 Conclusion

In robotics, we are facing the double challenge of facing a complex problem with the objective of running it on real-time. Filtering methods are not only known for being able to produce real-time estimation, they are also easier to implement, well studied in the literature and the estimation one can get may be satisfying depending on the application. However the remaining issue would be to find the best suited approach among all the different filters that now exist as each new filter often corresponds to some particular new hypothesis or approximation, each of them bringing some drawbacks, limitations or condition of use. In contrast, batch-optimisation methods are indisputable for off-line treatments. In visual SLAM, it is now accepted that batch-optimisation only offers advantages with respect to filtering, except for minimal systems [Strasdat 2012]. Is this also true when less rich sensors and no map are involved? In [Mirzaei 2008] the authors compared the results of an EKF algorithm for an IMU-Camera calibration problem with a bundle Adjustment that they consider as giving the 'best achievable (offline) estimates.

Their results indicate that the EKF method and the batch least-square estimator are achieving similar results. However, they are not mentioning anything about the computation time. The main distinction between both filtering and online batch-optimisation methods might be the way to treat past data. Filtering methods tend to summarise the information gained with past historic of poses and measurements and propagate the joint probability distributions through time. Batch-optimisation methods discard unused measurements and historic poses to reduce the complexity of the problem and make the optimisation possible while the propagation of the probability distributions through time is made unnecessary.

There is also a broad middle ground between filtering and smoothing methods. However, if one tries to define the best possible filter by modifying the standard approach, one would converge more and more towards online batch-optimisation [Strasdat 2012]. On the one hand, filtering methods have shown promising improvements these last years in terms of estimation accuracy. In [Mourikis 2007] the authors proposed a multi-state Kalman Filter and applied the estimator to a vision-aided inertial navigation problem providing results of nearly equal quality to batch-based methods. [Li 2013] is yet another example of EKF-based algorithm showing similar results than batch-optimisation methods in real-time. On the other hand, batch-optimisation methods have also been able to reach realtime operation [Leutenegger 2015, Mourikis 2009, Leutenegger 2015].

In this thesis we will focus on estimating the state of one rigid body under IMU, contact and odometry measurements. Considering only this information, we might have considered filtering approach for its simplicity. However we have worked in the broader context of later enriching the IMU measurements with visual data (yet left as a perspective, but on which we are currently working). We are also willing to contribute to a generic framework of estimation in robotics, by considering a system that does not need to be deeply refactored each time a new sensor is attached to the robot. We do not see that happening with filtering. For all these reasons, our work is bound to the online batch-optimisation approach, where our contributions take place. Let us insist again on the fact that filtering and optimisation are finally two faces of the same coin, as filtering is often a linear numeric solver on top of some clever marginalisation, but with the same underlying knowledge graph. Our contributions are then likely to be nearly directly useful in filtering.

Chapter 2

The technology behind IMUs

Contents

2.1 Measurements

The IMU provides a measurement of the *proper acceleration* as well as the *angular velocity* of the sensor itself in an inertial frame. Thus the IMU is a combination of at least both 3-axis accelerometer and 3-axis gyroscope. A 3-axis magnetometer measuring the magnetic field intensity along the axis of the IMU is sometimes also included. The acceleration is measured through the displacement of a mass retained by a spring as illustrated in Fig. 2.2a. The magnitude of this acceleration is then proportional to a force via the relation to the elasticity of the spring, itself proportional to the acceleration via the principle of dynamics. A simplified scheme of both gyroscopes and accelerometers in IMU is given in Fig. 2.1. An IMU at rest on the surface of the Earth will measure an upward acceleration of 9.81 $m.s^{-1}$ due to the gravity force. The angular velocity is obtained by detecting the changes of dynamics of an oscillating or rotating system via the Coriolis forces (Fig. 2.2b).

IMU sensors are very popular in robotics as they have been used for inertial-only navigation [Barshan 1995, Nilsson 2014a], visual-inertial navigation [Roussillon 2011, Konolige 2008], attitude estimation [Hamel 2006] or even applications using smartphones [Li 2013]. Indeed, by means of time integration of the provided measurements, it is possible to estimate the trajectory of the IMU or to detect events

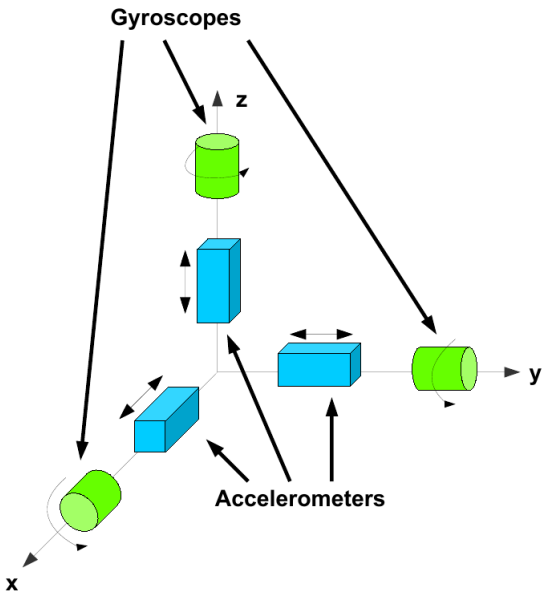
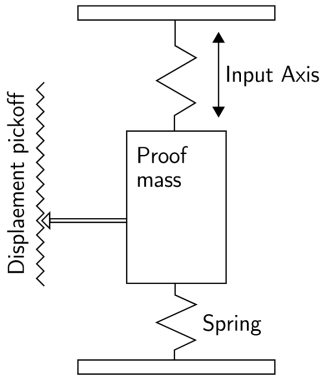
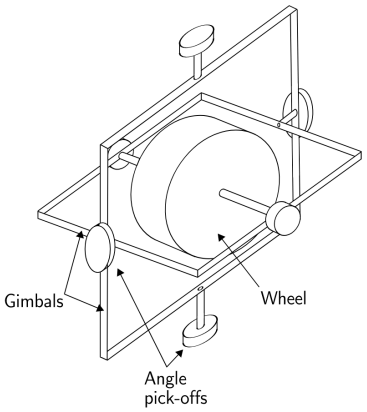


Figure 2.1: Simplified scheme of gyroscopes and accelerometers in an IMU [Tedaldi 2013].

such as slippage in the applications we are concerned with. In our case, we are using IMUs based on MEMS (micro electromechanical systems) technology resulting in micro scale sensors. There has been a rapid growth in the development of such sensors due to their extensive use in machines as the computers and smartphones while the production cost kept decreasing. These sensors have advantages such as compactness and relatively high bandwidth. However, they are also suffering from drawbacks such as non-neglect able biases in their measurements and also their sensitivity to temperature. Several characteristics must be taken into account when choosing an IMU. In addition to its precision, the operating range is an important criterion and can be limited due to physical factors coming from the mechanical elements the IMU is built with and that have their own limitations. Other criteria that can be taken into account are usually the axis misalignments, the biases ranges or the scale factor errors. The calibration process refers to the identification of these parameters. There are commercial IMUs that are factory calibrated. However, this implies that the manufacturer performed data acquisition with IMUs attached to a specific machine and performing motions so that all these errors are made observable to estimate it. Naturally, these processes increase considerably the cost of the IMUs since each sensor is usually sold with its own calibration parameters stored inside a non-volatile memory or in the firmware itself. As a result, the hardware itself is only a fraction of the final cost of these IMUs but the process leading to provide off the shelf accurate measurement is what people are actually buying to these companies.



(a) A mechanical accelerometer (source: [Titterton 2004])



(b) A mechanical gyroscope (source: [Titterton 2004])

Figure 2.2: Mechanical representation of an accelerometer and a gyroscope.

2.2 The sources of errors

2.2.1 Misalignment errors

In an ideal IMU, the 3 axes of the accelerometers are orthogonal and aligned with the 3 axes of the gyroscope. The accelerometer measures the accelerations distinctly along these 3 axes while the gyroscope measures the angular velocity around each of the 3 distinct axes. However, in real MEMS IMUs and due to assembly inaccuracies, the axes may be not only non-orthogonal but also misaligned (see Fig. 2.3). Let us call the ideal orthogonal accelerometer and gyroscope frames respectively A_B and W_B and the real (non-orthogonal) frames A_S and W_S. By thinking separately about the accelerometer and the gyroscope, these frames can be described by the sets of axes $(\mathrm{x}^B, \mathrm{y}^B, \mathrm{z}^B)$ for A_B (respectively W_B) and $(\mathrm{x}^S, \mathrm{y}^S, \mathrm{z}^S)$ for A_S (respectively W_S) in Fig. 2.3. Acceleration (respectively angular velocity) measurements are acquired in the non-orthogonal frame A_S (respectively W_S). Depending on the amplitude of the misalignments and how far the frame is from an orthogonal one, a correction might be required to express the output measurements in the orthogonal frame. For small angles, this correction is expressed by a transformation matrix $\mathbf{M}$ given angles as described in Fig. 2.3. Thus if $\mathbf{a}^S$ denotes the 3D acceleration measurements expressed in A_S, then $\mathbf{a}^B$ the corresponding measurement expressed in A_B is given by

$$\mathbf{a}^B = \mathbf{M}\,\mathbf{a}^S \tag{2.1}$$

with

$$\mathbf{M} = \begin{bmatrix} 1 & -\beta_{yz} & \beta_{zy} \\ \beta_{xz} & 1 & -\beta_{zx} \\ -\beta_{xy} & \beta_{yx} & 1 \end{bmatrix} \tag{2.2}$$

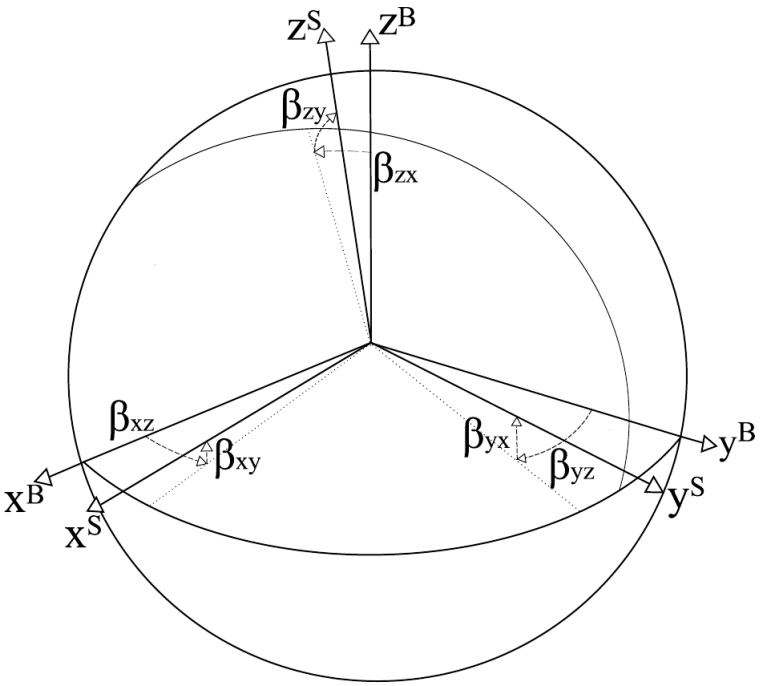


Figure 2.3: Representation of the misalignment and non-orthogonality of accelerometer and gyroscope axes in an IMU [Tedaldi 2013]. The 3D frame formed by the set of axes $(\mathrm{x}^B, \mathrm{y}^B, \mathrm{z}^B)$ is orthogonal while the frame formed by the axes $(\mathrm{x}^S, \mathrm{y}^S, \mathrm{z}^S)$ is misaligned and non-orthogonal.

The exact same relation can be written for the gyroscope measurements. Finally, misalignments and orthogonality for accelerometers and gyroscopes are different. Thus specific transformation matrices $\mathbf{M}^a$ and $\mathbf{M}^\omega$ might be required to correct respectively acceleration and rate of turn data.

$$\mathbf{a}^B = \mathbf{M}^a \, \mathbf{a}^S \ , \ \boldsymbol{\omega}^\omega = \mathbf{M}^\omega \, \boldsymbol{\omega}^S \tag{2.3}$$

In the ideal case $\mathbf{M}^a$ and $\mathbf{M}^\omega$ are the identity matrix.

2.2.2 Scale factor errors

The scale factor error is the relation between the input and the output. In an ideal IMU, we expect the input and the output to be equal. However in a real IMU, the output might be affected by a linear effect resulting in an output that is proportional to the input. The scale factors on each axis of the IMU are independent so that we can define two scaling matrices $\mathbf{K}^a$ and $\mathbf{K}^\omega$ so that the accelerometer and gyroscope outputs can be described respectively by

$$\mathbf{a}^B = \mathbf{K}^a \, \mathbf{a}^S \tag{2.4}$$

$$\boldsymbol{\omega}^B = \mathbf{K}^\omega \, \boldsymbol{\omega}^S \tag{2.5}$$

where

$$\mathbf{K}^a = \begin{bmatrix} s_x^a & 0 & 0 \\ 0 & s_y^a & 0 \\ 0 & 0 & s_z^a \end{bmatrix}, \ \mathbf{K}^\omega = \begin{bmatrix} s_x^\omega & 0 & 0 \\ 0 & s_y^\omega & 0 \\ 0 & 0 & s_z^\omega \end{bmatrix} \tag{2.6}$$

With (2.3), (2.4) and (2.4), we get the following IMU model error

$$\mathbf{a}^B = \mathbf{M}^a \ \mathbf{K}^a \ \mathbf{a}^S \tag{2.7}$$
$$\boldsymbol{\omega}^B = \mathbf{M}^\omega \ \mathbf{K}^\omega \ \boldsymbol{\omega}^S \tag{2.8}$$

Scale factor effects are most apparent in times of high acceleration and rotation.

2.2.3 Biases

The output measurements of IMUs are affected by biases for both acceleration and rate of turn components. Each axis of the IMU is affected by its own offset that is independent from the other axis. Thus the bias can be modelled with six components or by two vectors $\mathbf{a}_b$ and $\boldsymbol{\omega}_b$ defined as

$$\mathbf{a}_b = \begin{bmatrix} \mathbf{a}_{b,x} \\ \mathbf{a}_{b,y} \\ \mathbf{a}_{b,z} \end{bmatrix}, \ \boldsymbol{\omega}_b = \begin{bmatrix} \boldsymbol{\omega}_{b,x} \\ \boldsymbol{\omega}_{b,y} \\ \boldsymbol{\omega}_{b,z} \end{bmatrix} \tag{2.9}$$

These biases take part in the sensor error model as additive terms directly on the inputs so that

$$\mathbf{a}^S_{\text{output}} = \mathbf{a}^S_{\text{input}} + \mathbf{a}_b \tag{2.10}$$
$$\boldsymbol{\omega}^S_{\text{output}} = \boldsymbol{\omega}^S_{\text{input}} + \boldsymbol{\omega}_b \tag{2.11}$$

The complexity of the bias estimation can come from two major points. First of all, for each power up of the IMU, the initial bias is different. These changes are partially due to modifications of the physical properties of the IMU itself and different initial conditions. A higher *bias repeatability* tends to allow a better initial tuning of the IMU's parameters thus leading to a faster convergence towards good bias estimates. At the opposite, a higher variability leads to more difficult bias estimations. Furthermore, the biases change over time making their estimation even more important and difficult. The changes in bias can be related to temperature, time or even mechanical stress applied on the system. The evolution of the biases over time can be modelled as a random walk which depends on the quality of the IMU. The higher the quality of the IMU the stabler we can expect the biases to be over time. For a proper use of the sensor, the biases need to be removed from the sensor measurements before the data are used. Using equations (2.7), (2.8), (2.10)

and (2.11) we get the following IMU error model.

$$\mathbf{a}^B = \mathbf{M}^a \ \mathbf{K}^a \ \mathbf{a}^S + \mathbf{a}_b \tag{2.12}$$

$$\boldsymbol{\omega}^B = \mathbf{M}^\omega \ \mathbf{K}^\omega \ \boldsymbol{\omega}^S + \boldsymbol{\omega}_b \tag{2.13}$$

2.2.4 Noises

Finally, IMUs do not make an exception among sensors as they are also affected by measurement noises. The noise affecting each axis of the IMU is independent from the others thus also resulting in two noise vectors $\mathbf{a}_n$ and $\boldsymbol{\omega}_n$ so that

$$\mathbf{a}^S_{\text{output}} = \mathbf{a}^S_{\text{input}} + \mathbf{a}_n \tag{2.14}$$

$$\boldsymbol{\omega}^S_{\text{output}} = \boldsymbol{\omega}^S_{\text{input}} + \boldsymbol{\omega}_n \tag{2.15}$$

with

$$\mathbf{a}_n = \begin{bmatrix} \mathbf{a}_{n,x} \\ \mathbf{a}_{n,y} \\ \mathbf{a}_{n,z} \end{bmatrix} \ , \ \boldsymbol{\omega}_n = \begin{bmatrix} \boldsymbol{\omega}_{n,x} \\ \boldsymbol{\omega}_{n,y} \\ \boldsymbol{\omega}_{n,z} \end{bmatrix} \tag{2.16}$$

where each component can be described with a Gaussian distribution.

$$\mathbf{a}_n \sim \mathbb{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\boldsymbol{a},\boldsymbol{n}}) \ , \ \boldsymbol{\omega}_n \sim \mathbb{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\boldsymbol{\omega},\boldsymbol{n}}) \tag{2.17}$$

$\boldsymbol{\Sigma}_{\boldsymbol{a},\boldsymbol{n}}$ and $\boldsymbol{\Sigma}_{\boldsymbol{\omega},\boldsymbol{n}}$ are different and depend on the sensor itself. Finally, using (2.12), (2.13) and (2.16) the complete sensor error model becomes

$$\mathbf{a}^B = \mathbf{M}^a \ \mathbf{K}^a \ \mathbf{a}^S + \mathbf{a}_b + \mathbf{a}_n \tag{2.18}$$

$$\boldsymbol{\omega}^B = \mathbf{M}^\omega \ \mathbf{K}^\omega \ \boldsymbol{\omega}^S + \boldsymbol{\omega}_b + \boldsymbol{\omega}_n \tag{2.19}$$

2.2.5 Used IMU sensor model

The effects of aforementioned errors are not the same given the experimental conditions. The scale factor error in IMU is usually not a large contributor to the total errors. The contribution of the accelerometer scale-factor error increases with the input acceleration. Furthermore, it causes position and velocity calculation errors through the integration of the errors but also inaccuracies on the estimation of pitch and roll angles. In a similar way, the gyroscope's scale-factor error is larger when the input increases, that is during motions with large angular velocities. The effect one can expect from this source of error is difficulties to track highly dynamic motions. Velocity and position errors due to the accelerometer scale-factor can be expressed as

$$\mathbf{v}_{\text{error}} = \mathbf{a}_{\text{sf\_error}} * \mathbf{g} * t \tag{2.20}$$

$$\mathbf{p}_{\text{error}} = \frac{\mathbf{g} \ * \ t^2}{2} * \mathbf{a}_{\text{sf\_error}} \tag{2.21}$$

where $\mathbf{a}_{\text{sf\_error}}$ and $\boldsymbol{\omega}_{\text{sf\_error}}$ are respectively the accelerometer and gyroscope scale-factor errors while $\mathbf{g}$ and t are the gravity and the integration time.

Compared to the other possible sources of errors, both accelerometer and gyroscope biases are significant contributors to position, velocity and orientation errors. This major contribution is due not only to the difficulty to estimate time-varying biases made only more complex through their observability conditions requiring various poses and motion dynamics but also to the integration of these errors for position and velocity estimation. Besides, these errors occur during any motion of the IMU. Thus we decide to neglect other sources of error than biases and noises. The resulting IMU error model is described by

$$\mathbf{a}^S_{\text{output}} = \mathbf{a}^S_{\text{input}} + \mathbf{a}_b + \mathbf{a}_n \tag{2.22}$$

$$\boldsymbol{\omega}^S_{\text{output}} = \boldsymbol{\omega}^S_{\text{input}} + \boldsymbol{\omega}_b + \boldsymbol{\omega}_n \tag{2.23}$$

2.3 Calibration of the sensor

2.3.1 Introduction

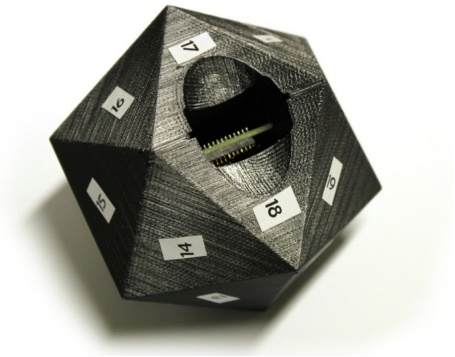
The calibration process usually consists in placing the device in different configurations to apply different stimuli on it. These different excitements introduced as input measures to the IMU enhance the observability of the different parameters thus allowing to determine the actual error in each measurement. There are different calibration methods to estimate the parameters of the IMU. The first methods were based on the use of mechanical equipment. Then came methods going towards estimation using sensor fusion instead of a specific costly equipment.

2.3.2 Calibration based on the use of mechanical equipment

Factory calibration processes are still using specific and costly mechanical equipment. Different equipment may be required to estimate the different parameters of the IMU. Thus accelerometer calibration can be performed by rotating the sensor so that each axis is aligned with the known gravity. This process can be done using a dividing-head precision device that is able to rotate precisely. The gyroscope calibration can be performed using a high precision rate table. There exist different types of rate tables according to the number of degrees of freedom they can operate in. The higher the degree of freedom the more complex the generated motions can be and the quicker the calibration process is expected to be. Once all the measurements are made, a batch linear least square or a Kalman Filter can be used to estimate the optimal calibration parameters while minimising the errors.

In [Magnussen 2015] the authors are describing the calibration process using a 6 degree of freedom hexapod for a strapdown integration of the IMU on an Unmanned Aerial Vehicle (UAV). This calibration process is used to estimate the internal parameters of the IMU and the position offset due to the pose of the IMU on the UAV. [Skog 2006] proposes a calibration method that requires no me-

(a) 3-Axis Rate Table Series AC3337-TC from ACUTRONIC. The rate table is specifically designed for testing IMUs. Accuracy and high rate control as well as rate stability are very important features for rate tables.



(b) The IMU array is placed inside the icosahedron. The different faces of the icosahedron allow to measure the Earth's gravity when the IMU is placed in different orientation while leaving the sensor still in an even distribution of unknown orientations. [Nilsson 2014b]

Figure 2.4: Tools designed for calibration purposes

chanical equipment for the accelerometer but a rate table for the gyroscope. The proposed method is based on the measurement properties of the IMU resulting in an equality of the norm of the measured output of the accelerometers (respectively of the gyroscopes) clusters and the magnitude of force (the rotational velocity respectively). Further calibration methods using mechanical equipment are described in [Cho 2005, Pittelkau 2006, Beravs 2012] with different methods.

2.3.3 Calibration without mechanical equipment

With the increase of potential applications using IMUs, more effort has been made towards easier calibration of inertial measurements units. [Fong 2008] proposed a calibration method for both accelerometer and gyroscope that does not require any mechanical equipment. This method is also used to compensate both axis misalignment and scale factor errors using the Earth's gravity as a stable physical force applied on the IMU and arbitrary motions. To be more specific, the idea behind is to calibrate the gyroscope by comparing the estimated orientation and the output of the accelerometer measurements. A similar idea is used in [Cheuk 2012] to extend the calibration to the magnetometers. The importance of calibration procedures for low-cost IMUs also increased the interests for an easy estimation of IMU's parameters. Thus [Nilsson 2014b] proposed a device to calibrate single-chip IMUs that can be used to correct the misalignment errors. This method is based on static measurements taken while the IMU is placed in different orientations. The measurements are made easy thanks to a printable device that is an icosahedron solid (see Fig. 2.4b) where the IMU can be placed in and available for download.

The recorded data used to estimate the misalignment of the axis, the gain and biases using a maximum likelihood based approach. Unfortunately, the method requires the IMU to be still in the solid designed for a specific array on which it is placed.

2.3.4 Self-calibration

Alternatively, calibration can be performed online, in real-time alongside all other operations of the system, in what is known as self-calibration. For this, the system needs to use additional sensors that, when operating together, render the parameters to calibrate observable.

We will be presenting self-calibration methods with more details in Chapter 4, after the IMU particular case developed in Chapter 3. Indeed, in Chapter 3 we apply these methods to the self-calibration of the IMU in $SO(3)$ in applications of pedestrians and humanoid walking. In Chapter 4, we generalise the methodology for self-calibrating motion sensors in the framework of graph-based optimisation on manifold.

Chapter 3

Graph based IMU preintegration on the S^3 manifold, with application to localisation

Contents

3.1 Introduction

3.1.1 Context

Indoor localisation is an open challenge in various situations: location-based life improving services, firefighters localisation and navigation, patient tracking motion monitoring, medical observation, accident monitoring, mobility and independence of partially sighted or blind persons, etc. As GPS are not available indoor, and because relying on a network of fixed sensors (cameras, RFID) also raised many open questions, an appealing way to localize a body in space is to use odometry information measured by embedded inertial measurement units (IMUs). In this context, while performing the integration it is mandatory to take into account the IMU biases. As the biases vary with time and physical conditions, it must be estimated on-line while processing the measurements. Furthermore, it is desirable that additional information coming from other sensors can be integrated in the same estimation process. Similarly to the strategies adopted in simultaneous localisation and mapping (SLAM), sparse measurements or additional information (*e.g.* coming from intermittent absolute localisation, or from a sparse sensor network) would benefit to the localisation process when available. With these requirements coming from the context in mind, we propose to define an estimator based on graphical models, able to accurately and efficiently integrate inertial measurements while estimating the IMU biases. Thanks to the graphical model, the estimator will then be able to fuse additional measurements coming from other sensors or additional prior information provided by the application. The long-term goal is to merge this method with visual SLAM estimates. Indeed, the method that we propose is suitable for merging with any other source of information. While the fusion with visual data is still work in progress, we propose here two simpler -yet useful- experimental contexts: pedestrian indoor tracking and legged robotics.

3.1.2 Related Work

One of the most common ways to describe the trajectory of a body in space is to use odometry, which can be derived from a multitude of sensors, from encoders to GPS and cameras. The quality of this information is also known to be dependent on the sensor specifications and to accumulate measurement errors. The development of SLAM (Simultaneous Localization And Mapping) is mature enough to resolve this dependency over time with loop closure strategies. However, other strategies are required to comply with the particularities of legged locomotion and physical interactions in unstructured environments. This is particularly necessary in outdoor or industrial applications with cluttered environment. In this case, assuming a flat floor is not always reasonable and being able to observe the foot pose allows to implement advanced reactive locomotion strategies.

While this problem is an open challenge in various domains, like indoor pedestrian tracking for life service, we mostly focus here our analysis of the litterature to the robotics domain. In the DRC [Johnson 2017, Marion 2017, Karumanchi 2017]

most of the teams used laser or RGB-D cameras to build a reconstruction of the floor surface in order to plan the next feet pose. For some teams the level of accuracy was not always sufficient and involved dramatic failures despite intensive training [Kaneko 2015]. On the other hand, the IHMC team [Johnson 2017] reported an important gain in terms of accuracy using their state estimator alone, reaching an impressive 1 cm drift per every three steps for the pelvis horizontal position, and 5 mm per every nine steps for the pelvis vertical position. Using a quite simple state estimator, it is very interesting to note the following reported factors for reaching this level of precision besides bug fixing: the redesign of Atlas, coming with a significant reduction of backlash in the leg joints, improving measurements using kinematics and a walking gait reducing the amount of foot slipping and bouncing. Although the IHMC tested a localization algorithm [Pomerleau 2013], occasional localisation errors were a problem in the overall behavior and SLAM happened to be not necessary in this situation.

For robots where the design choices introduce flexibilities such as HRP-2 [Nakaoka 2007], or backlashes, the system needs a state estimator being able to take into account this uncertainty. Furthermore, the design of a robot gets even more complex with the increase in the number of degrees of freedom (dof). Estimating the state of the robot usually requires the use of successive transformation using the kinematic chain, thus reconstructing the pose of a specific joint from the estimation of another point in the robot such as the free flyer. In a first instance, errors due to encoders measurements can be neglected although they accumulate over time. However, a good pose estimation of the robot in its environment is still needed for effective interactions with the environment, and especially for behavior involving manipulation or multiple contact locomotion. This estimation can then be used to compute the commands that will be sent to actuators. An accurate model of the robot can then be considered to get better pose estimations. However, although the identification of rigid robot's dynamics is well studied, it is still a complex and time-consuming task.

Localisation can be used to perform real-time planning and model predictive control. To achieve this goal, fusion strategies with information coming from different sensors is needed. Fusion strategies have always been used on humanoid robots to improve the pose estimation. [Stasse 2006] is considered as the first implementation of real-time vision-based 3D SLAM applied to a humanoid robot and using both pattern generator and inertial information with loop closing capabilities. In [Ahn 2012] the authors propose a method for vision-based 3D motion estimation using on-board odometry information and inertial sensing. The proposed method is divided into two submodules that are a EKF-based odometry estimation module and a vision-based SLAM method. They claim to improve the pose estimation given by the second method by augmenting it with the odometry obtained by using the forward kinematics model of the robot and information provided by encoders. The IMU is a main component of the proposed method and operated at 100 Hz. Fallon et al. [Fallon 2014] go further by using leg kinematics, inertial measurements and visual perception provided by LIDAR sensor to perform drift-free state estimation

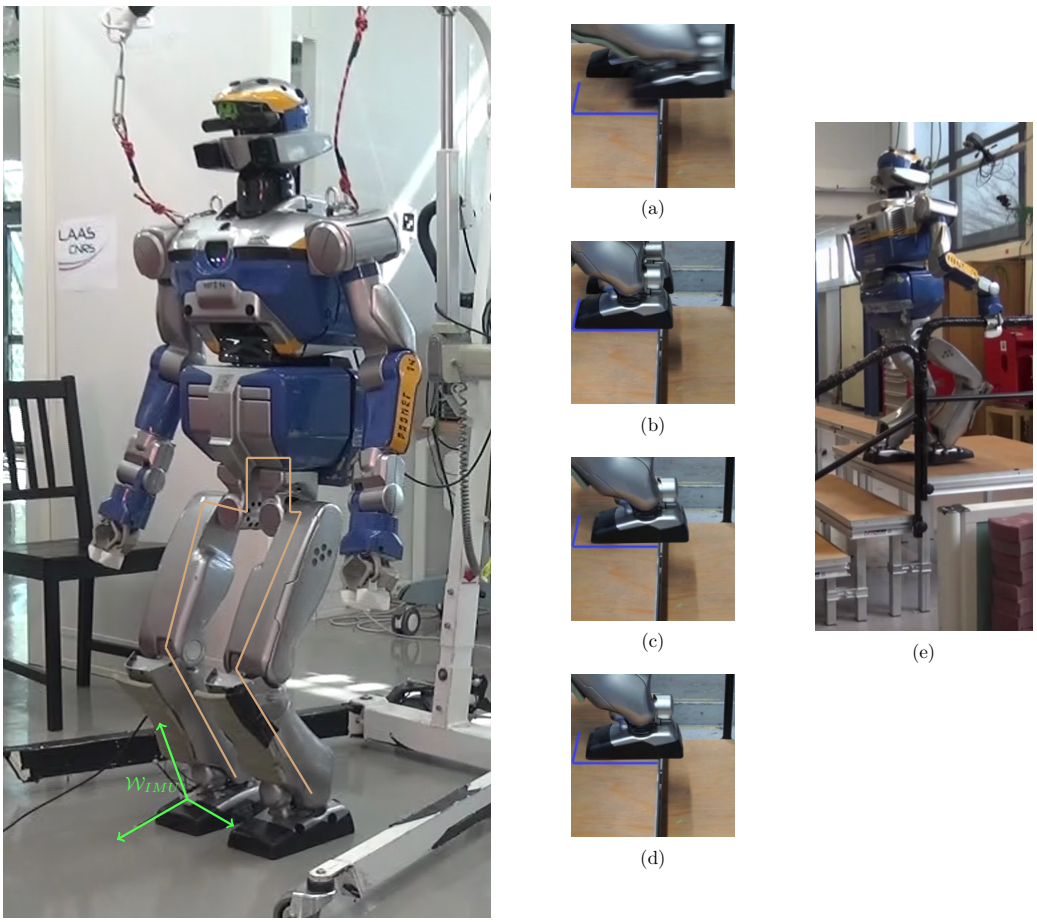


Figure 3.1: The flying foot trajectory is reconstructed through an IMU set on the foot (in green), and by fusing the information coming from the kinematic chain from the support foot to the flying one (in light brown). In the middle images (a)-(d) shows slippage while performing a multicontacts motion depicted on the right (e) (from [Carpentier 2016]).

applied to ATLAS humanoid robot (Fig. 3a) while localising it in a prior map. The distinct sensing modalities are used with two different state estimators. The first one provides position and velocity estimates with high rate and low latency so that these can be used for control feedback. The second estimator is using vision sensors for long-term localisation while accepting small short-term corrections. The presented method allows them to have the robot walking over uneven terrain without stopping for ten minutes.

Graphical methods have been extensively used to implement such fusion strategies [Thrun 2006, Kaess 2008a]. They have been used for large modelling estimation problems by means of sparse networks of constraints. In robotics, the problems of visual odometry, and simultaneous localisation and mapping, have reached a high degree of maturity, in great part thanks of the graphical representation. This is so, among other aspects, because of the power of the graphical representation to accurately model complex estimation problems. These often involve dynamics, proprioceptive measures, exteroceptive measures, and self-calibration. The graphical representation also allows for the design of powerful nonlinear estimation solvers, which can be built taking into account the needs for accuracy, robustness and CPU-performance. In order to keep the problem tractable and maintain real-time performance a key point is to avoid the graph to be too large for a given time window. However, trajectory correction using SLAM and more generally visual information implies to revise the control sequences and the planning at a much lower frequency due to sensor imperfections and the need in SLAM for a loop-closure to propagate uncertainty corrections.

3.1.3 Methodology

Graphical methods have been extensively used to implement such fusion strategies [Thrun 2006, Kaess 2008a]. They are well-suited to gather information from sensors and draw conclusions. The underlying principle is to consider that despite all the information gathered from the sensors, we still have uncertainties about the true state of the world due to imperfections of the sensors. Several states of the world can thus be considered as probable. Relying on probabilistic formulations is a way to find out the most probable one. Furthermore, graphical representations are able to accurately model complex estimation problems [Koller 2009] in a versatile way.

Graphical models have been used for large modelling estimation problems by means of sparse networks of constraints and particularly in robotics where SLAM and visual odometry problems have reached a high degree of maturity in great part thanks to these tools. The graphical representation also allows for the design of powerful nonlinear estimation solvers, which can be built taking into account the needs for accuracy, robustness and computational performances.

In order to keep the problem tractable and maintain real-time performance, a key point is to prevent the graph from being too large given a time window. IMUs are challenging in this regard, as their high frequency measurement rate

creates large sets of data. Pre-integration of IMU measurements helps to reduce the size of the underlying graph by squeezing 100 to 1000 measures into a single pre-integrated Bayesian node [Lupton 2009]. It was later suggested to preintegrate data in the $SO(3)$ manifold instead of using Euler angles [Forster 2015]. The IMU measurements can simply be disregarded even when the initial "pre-integration" conditions change while the numerical solver optimised the maximum-likelihood trajectory.

3.1.4 Contributions

We propose an alternative estimation method following a similar methodology. We define a graphical model where pre-integrated data are added at key-frame instants (about 2 Hz), summarising IMU measurements captured at about 1 kHz. The state that we want to estimate considers the position, orientation and linear velocity (dimension 9) of the IMU attached to the foot, along with the time-varying accelerometer and gyroscope biases (dimension 6). We also implement the prior knowledge that the foot lands horizontally, by adding Bayesian nodes when the foot lands and takes off. This prior knowledge makes the considered state observable. We then use a numerical solver to maximise the likelihood of the measurements and prior knowledge along the past trajectory.

The first contribution is to reformulate the pre-integration method introduced in [Forster 2015] using quaternion representation and to give a detailed and simpler algebraic derivations using the chain rule. The result of this preintegration method is a clear definition of Jacobians as well as a better understanding and thus easier integration for future applications. The second contribution is to apply this method for estimating the human foot-pose during walking. To this end, we build a graphical model formulation of the problem by gathering information. Once the formulation is completed, a nonlinear least-squares optimiser (Google Ceres [Agarwal]) is used to solve the problem and find the most probable solution in the least-square sense.

3.2 Graph-based inertial-kinematic odometry

3.2.1 Graph-based iterative optimisation

As explained in 1.3.2, the problem is represented as a graph, where the nodes refer to the variables, and the edges, are called factors that represent the geometrical constraints between variables due to measurements. We define here the graphical model proposed to handle the IMU. The state $\mathbf{x}$ is modelled as a multi-variate Gaussian distribution, and in our case it includes poses and velocities $(\mathbf{p}, \mathbf{q}, \mathbf{v})$ of the body to which the IMU is attached to and IMU biases $(\mathbf{a}_b, \boldsymbol{\omega}_b)$. This state is sampled at selected key-frames along the trajectory (see Fig. 3.2). For each factor, we have to define the corresponding measurement residual, *i.e.* an error or a residual $\mathbf{r}$ as the discrepancy between a measurement $\mathbf{z}$ and its expectation given

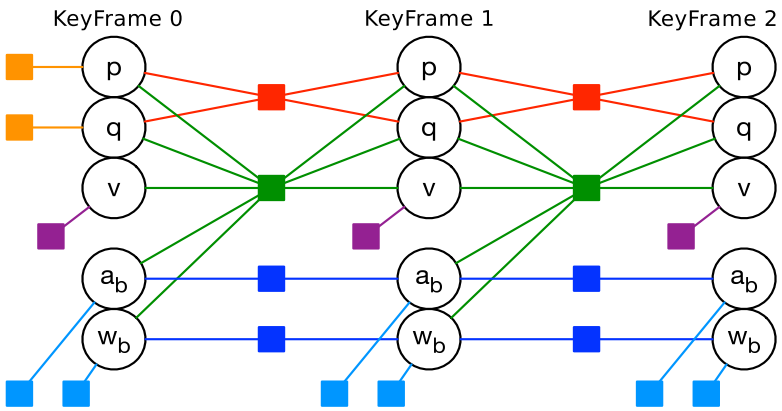


Figure 3.2: Detailed factor graph for the initial keyframe and two steps. *Circles*: state blocks for position ($\mathbf{p}$), orientation quaternion ($\mathbf{q}$), velocity ($\mathbf{v}$), accelerometer bias ($\mathbf{a}_b$), gyrometer bias ($\boldsymbol{\omega}_b$). *Orange*: initial pose factor. *Red*: kinematic factor (deduced from additional sensors). *Purple*: zero-velocity factor. *Green*: IMU's delta pre-integration factor. *Blue*: bias drift factor. *Cyan*: bias absolute factor.

the involved state variables,

$$\mathbf{r}(\mathbf{x}) = h(\mathbf{x}) + \mathbf{v} - \mathbf{z}, \qquad \mathbf{v} \sim \mathcal{N}(0, \boldsymbol{\Omega}^{-1}) \tag{3.1}$$

being $h(\mathbf{x})$ the sensor measurement model and $\boldsymbol{\Omega}$ the information matrix of the measurement Gaussian noise $\mathbf{v}$. One of the difficulties brought by this estimation problem is that the variables defined on manifolds, such as quaternions or rotation matrices. We then must rewrite (3.1) as

$$\mathbf{r}(\mathbf{x}) = (h(\mathbf{x}) \oplus \mathbf{v}) \ominus \mathbf{z} \tag{3.2}$$

The $\oplus$ and $\ominus$ symbols correspond to the addition and subtraction operators on the manifold that we more completely introduce in C.1.3.

As explained in 1.3.3, the maximum a posteriori estimation is obtained by iteratively minimising the Mahalanobis squared norm of all linearised errors

$$\Delta\mathbf{x}^* = \arg\min_{\Delta\mathbf{x}} \sum_k \|\mathbf{r}_k(\check{\mathbf{x}}) + \mathbf{J}_k \Delta\mathbf{x}\|^2_{\boldsymbol{\Omega}_k^{-1}} \tag{3.3}$$

being $\check{\mathbf{x}}$ the state estimate at the current iteration, and $\mathbf{J}_k$ the Jacobian of the k-th residual $\mathbf{r}_k(\mathbf{x})$ (with $\mathbf{J}_k = \partial(h_k(\mathbf{x}) \ominus \mathbf{z}_k)/\partial\Delta\mathbf{x}$ in the case of variables lying on a manifold) and $\boldsymbol{\Omega}_k$ is the information matrix of the k-th measurement. Here we do not have to enter in the details of the optimisation scheme, that we kept generic (we literally used the optimiser on manifold Ceres without any additional flavour). We just have to precisely define this function, and its derivatives $\mathbf{J}$.

3.2.2 Where to put Keyframes?

During a biped walk, we take profit of certain situations where precise and reliable assumptions can be made. For example, the foot velocity is null during its support phase. At these selected instants, we create the keyframes that will produce a chain of states. These states are linked by the measurements, forming our factor graph (Fig. 3.2). Each keyframe $\mathbf{f}_i$ contains the following state blocks: the position of the foot, velocity and orientation data, plus the IMU accelerometer and gyrometer biases,

$$\mathbf{f}_i = \begin{bmatrix} \mathbf{p}_i & \mathbf{v}_i & \mathbf{q}_i & \mathbf{a}_{b,i} & \boldsymbol{\omega}_{b,i} \end{bmatrix}^\top . \tag{3.4}$$

3.2.3 Description of factors

The types of factor considered in our graph are illustrated in Fig. 3.2. Each factor k requires its own information matrix $\boldsymbol{\Omega}_k$, and its residual function $\mathbf{r}_k(\mathbf{x})$. These residual functions are detailed hereafter.

3.2.3.1 Absolute factors

These include initial position and orientation (orange in the figure), zero velocity (purple), and bias magnitude (cyan). Each residual depends on a single state block, which is compared against a reference $\mathbf{z}_k$,

$$\mathbf{r}_k(\boldsymbol{\phi}_i) = \boldsymbol{\phi}_i - \mathbf{z}_k \tag{3.5}$$

where $\boldsymbol{\phi}_i$ is one among $\{\mathbf{p}_i, \mathbf{v}_i, \mathbf{a}_{b,i}, \boldsymbol{\omega}_{b,i}\}$. For the quaternion we implement the residual using the operator $\ominus$ on the sphere of dimension 3 manifold, denoted S^3 (see (C.5) in Appendix C.1 for further details),

$$\mathbf{r}_k(\mathbf{q}_i) = \mathbf{q}_i \ominus \mathbf{z}_k = \mathrm{Log}(\mathbf{z}_k^* \otimes \mathbf{q}_i) \tag{3.6}$$

3.2.3.2 Bias drift factors (blue)

These are relative factors that allow the bias estimates to drift with time at a controlled rate. Each bias drift residual depends on two state blocks, namely

$$\begin{aligned} \mathbf{r}(\mathbf{a}_{b,i}, \mathbf{a}_{b,j}) &= \mathbf{a}_{b,j} - \mathbf{a}_{b,i} \\ \mathbf{r}(\boldsymbol{\omega}_{b,i}, \boldsymbol{\omega}_{b,j}) &= \boldsymbol{\omega}_{b,j} - \boldsymbol{\omega}_{b,i} \end{aligned} \tag{3.7}$$

3.2.3.3 Complementary factors (red)

These relate position and orientation between two consecutive steps as it can be provided by other sensors than IMU or methods using human walking specificities,

$$\mathbf{r}(\mathbf{p}_i, \mathbf{q}_i, \mathbf{p}_j, \mathbf{q}_j) = \begin{bmatrix} \mathbf{q}_i^* \odot (\mathbf{p}_j - \mathbf{p}_i) - \mathbf{y}_k \\ \mathrm{Log}(\mathbf{z}_k^* \otimes \mathbf{q}_i^* \otimes \mathbf{q}_j) \end{bmatrix} \tag{3.8}$$

where $\mathbf{y}_k$ and $\mathbf{z}_k$ are respectively the relative position and quaternion measurements.

3.2.3.4 IMU pre-integrated factors (green)

Due to the different rates of IMU data and keyframe creations, hundreds of IMU measurements need to be integrated to generate a motion factor linking two consecutive keyframes. In addition to that, the integration of the motion equations in an absolute reference frame strongly depends on the initial conditions of orientation, velocity and IMU bias. Therefore, the changes in the estimates of these magnitudes (inherent to the iterative nature of the optimisation) affect the whole motion integral. Delta pre-integration theory was developed to avoid the need of re-integrating all IMU data at each iteration [Lupton 2009, Forster 2015]. On the one hand, this theory defines new motion magnitudes called *deltas*, which are independent of the initial conditions for orientation and velocity, and thus depend *only* on the IMU data and bias. On the other hand, the effect of the changes in the bias estimates is linearised so that the deltas can be corrected a posteriori, *i.e.* when computing the residual, using pre-computed Jacobians.

In the coming section, we revise the IMU pre-integration theory, providing three contributions: 1) a segmentation of the computation pipeline (from measurements, to body magnitudes, to the current delta, and to the integrated delta); 2) a physical interpretation of the delta magnitudes; and 3) a simpler yet rigorous algebraic approach, valid for both the S^3 (quaternion) and $SO(3)$ (rotation matrix) manifolds, which takes profit of the pipeline segmentation and the chain rule to compute the otherwise cumbersome Jacobians [Forster 2015].

Some background that are not so common in robotics can be found in appendix Appendix C.1 to help follow the developments of the preintegration equations.

3.3 IMU pre-integration in S^3 and SO(3)

3.3.1 State integration in the absolute reference frame

We define the world-referenced states of the IMU by $\mathbf{x} = (\mathbf{p}, \mathbf{v}, \mathbf{q})$ where $\mathbf{p}$ stands for the position, $\mathbf{v}$ for the velocity and $\mathbf{q}$ for the orientation encoded as a quaternion. The time evolution of $\mathbf{x}$ is governed by the kinematic equation,

$$\begin{aligned} \dot{\mathbf{p}} &= \mathbf{v} \\ \dot{\mathbf{v}} &= \mathbf{g} + \mathbf{q} \odot \mathbf{a} \\ \dot{\mathbf{q}} &= \frac{1}{2}\mathbf{q} \otimes \boldsymbol{\omega} \end{aligned} \tag{3.9}$$

where $\mathbf{g}$ denotes the gravity vector and we identify $\mathbf{b} = (\mathbf{a}, \boldsymbol{\omega})$ as the *body magnitudes*, that is, the magnitudes of acceleration and angular velocity measured by the IMU and expressed in its reference frame. These body magnitudes are obtained at

discrete times t_j from biased and noisy IMU measurements, *i.e.*,

$$\begin{aligned} \mathbf{a}_j &\triangleq \mathbf{a}_{m,j} - \mathbf{a}_{b,j} - \mathbf{a}_n \\ \boldsymbol{\omega}_j &\triangleq \boldsymbol{\omega}_{m,j} - \boldsymbol{\omega}_{b,j} - \boldsymbol{\omega}_n \ , \end{aligned} \tag{3.10}$$

with $\bullet_m$ the measurements, $\bullet_b$ the biases, and $\bullet_n$ the noises. Assuming constant body magnitudes within the IMU sampling period $\delta t \triangleq t_k - t_j$, we have the discrete-time relation:

$$\begin{aligned} \mathbf{p}_k &= \mathbf{p}_j + \mathbf{v}_j \delta t + \frac{1}{2}\mathbf{g}\delta t^2 + \frac{1}{2}\mathbf{q}_j \odot \mathbf{a}_j \delta t^2 \\ \mathbf{v}_k &= \mathbf{v}_j + \mathbf{g}\delta t + \mathbf{q}_j \odot \mathbf{a}_j \delta t \\ \mathbf{q}_k &= \mathbf{q}_j \otimes \mathrm{Exp}(\boldsymbol{\omega}_j \delta t/2) \end{aligned} \tag{3.11}$$

3.3.2 Delta definitions

Consider a non-rotating reference frame that is free-falling at the acceleration of gravity $\mathbf{g}$, and name it $\mathcal{G}_t$. An ideal (unbiased and noiseless) IMU glued to this frame would measure null linear accelerations and angular velocities. Any non-null measurements would be due to a relative motion of the IMU with respect to $\mathcal{G}_t$. Thus, we identify the IMU deltas as being physically identifiable as the motion increments with respect to a free-falling frame, that is, a frame that falls with gravity as illustrated in Fig. 3.3.

At a given keyframe instant t_i, we initialize $\mathcal{G}_i$ at $\mathbf{x}_i = (\mathbf{p}_i, \mathbf{v}_i, \mathbf{q}_i)$. At a later keyframe instant t_j $(j > i)$, $\mathcal{G}_j$ has fallen according to $\mathbf{g}$, and the state of our moving body is now at $\mathbf{x}_j = (\mathbf{p}_j, \mathbf{v}_j, \mathbf{q}_j)$. The motion variation, denoted Δ_{ij}, is defined as the state variation in position, velocity and orientation of our body between $\mathcal{G}_i$ and $\mathcal{G}_j$, that is,

$$\begin{aligned} \Delta\mathbf{p}_{ij} &= \mathbf{q}_i^* \odot \left(\mathbf{p}_j - \mathbf{p}_i - \mathbf{v}_i \Delta t_{ij} - \frac{1}{2}\mathbf{g}\Delta t_{ij}^2\right) \\ \Delta\mathbf{v}_{ij} &= \mathbf{q}_i^* \odot (\mathbf{v}_j - \mathbf{v}_i - \mathbf{g}\Delta t_{ij}) \\ \Delta\mathbf{q}_{ij} &= \mathbf{q}_i^* \otimes \mathbf{q}_j \end{aligned} \tag{3.12}$$

where $\Delta t_{ij} \triangleq t_j - t_i$ is the time duration between the two keyframes. Notice that this definition of Δ_{ij} is the same as provided in [Lupton 2009, Forster 2015], and we have given it here a clear physical meaning. It is worth to notice that the deltas form a group under the composition law $\Delta_{ik} \triangleq \Delta_{ij} \boxplus \Delta_{jk}$ (Fig. 3.4), defined by:

$$\begin{aligned} \Delta\mathbf{p}_{ik} &= \Delta\mathbf{p}_{ij} + \Delta\mathbf{v}_{ij}\Delta t_{jk} + \Delta\mathbf{q}_{ij} \odot \Delta\mathbf{p}_{jk} \\ \Delta\mathbf{v}_{ik} &= \Delta\mathbf{v}_{ij} + \Delta\mathbf{q}_{ij} \odot \Delta\mathbf{v}_{jk} \\ \Delta\mathbf{q}_{ik} &= \Delta\mathbf{q}_{ij} \otimes \Delta\mathbf{q}_{jk} \end{aligned} \tag{3.13}$$

with identity $\Delta_0 = [(0,0,0),(0,0,0),(1,0,0,0)]$, and inverse $\Delta_{ji} \triangleq \Delta_{ij}^{-1}$ such that $\Delta^{-1} \oplus \Delta = \Delta \boxplus \Delta^{-1} = \Delta_0$. At any time j we can recover the state estimate $\mathbf{x}_j$

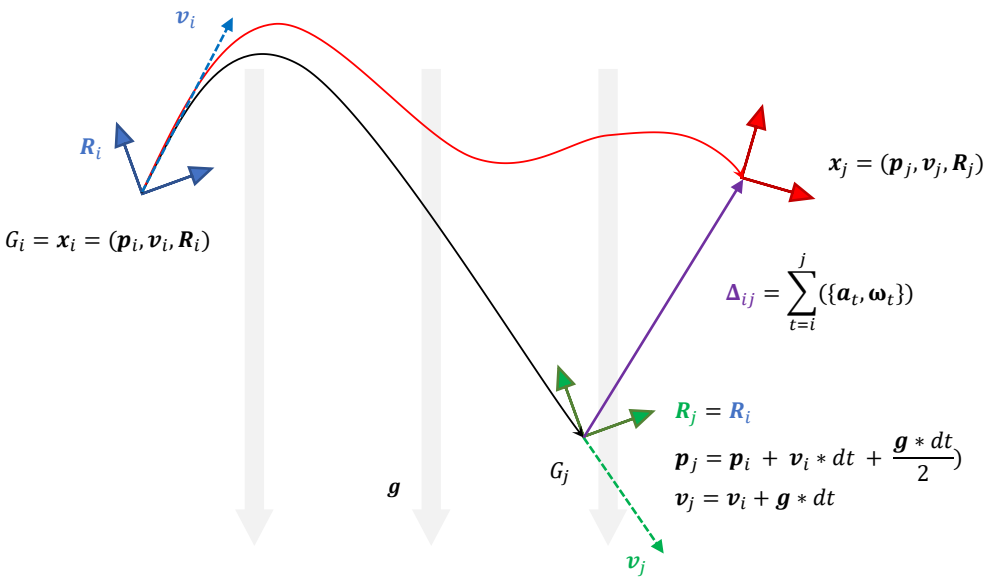


Figure 3.3: A physical interpretation of the deltas. The blue frame G_i is the initial frame at time $t = i$. At this moment, the IMU is moved with an instant velocity v_i from the pose given by $\{p_i, R_i\}$. If the rigid body the IMU is attached on does not move with a proper acceleration and rate of turn from time i to j, then the trajectory of the IMU would be given by the free-falling frame and the device would be on the frame G_j at time $t = j$. However if the IMU moves with a proper acceleration and rate of turn, then its pose at time j is given by Δ_{ij} with respect to the free-falling frame.

from the state estimate $\mathbf{x}_i$ and the motion delta Δ_{ij}:

$$\begin{aligned}\mathbf{p}_j &= \mathbf{p}_i + \mathbf{v}_i\Delta t_{ij} + \frac{1}{2}\mathbf{g}\Delta t_{ij}^2 + \mathbf{q}_i \odot \Delta\mathbf{p}_{ij} \\ \mathbf{v}_j &= \mathbf{v}_i + \mathbf{g}\Delta t_{ij} + \mathbf{q}_i \odot \Delta\mathbf{v}_{ij} \\ \mathbf{q}_j &= \mathbf{q}_i \otimes \Delta\mathbf{q}_{ij}\end{aligned} \tag{3.14}$$

3.3.3 Incremental delta pre-integration

Substituting the integration Eq. (3.11) in the delta definitions (3.12), we obtain the incremental delta pre-integration,

$$\begin{aligned}\Delta\mathbf{p}_{ik} &= \Delta\mathbf{p}_{ij} + \Delta\mathbf{v}_{ij}\delta t + \frac{1}{2}\Delta\mathbf{q}_{ij} \odot \mathbf{a}_j\delta t^2 \\ \Delta\mathbf{v}_{ik} &= \Delta\mathbf{v}_{ij} + \Delta\mathbf{q}_{ij} \odot \mathbf{a}_j\delta t \\ \Delta\mathbf{q}_{ik} &= \Delta\mathbf{q}_{ij} \otimes \mathrm{Exp}(\boldsymbol{\omega}_j\delta t)\end{aligned} \tag{3.15}$$

with $\Delta_{ii} = \Delta_0$. Interestingly, (3.15) is analogous to the motion of a body *in an inertial frame* under constant acceleration and rotation rate. Notice that by letting

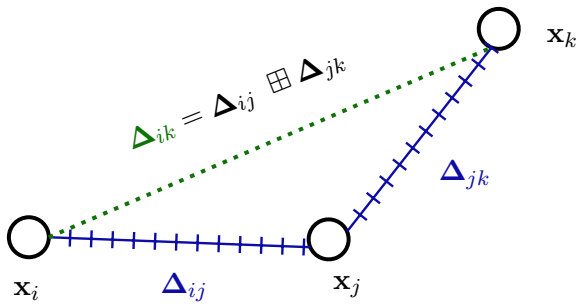


Figure 3.4: A representation of the $\boldsymbol{\Delta}$ composition law.

the reference frame fall with gravity, we get rid of the dependence on gravity in the integration equations, and only the body magnitudes drive the integral. Indeed, we can define a proper delta δ_{jk} from the current body magnitudes $\mathbf{b}_j = (\mathbf{a}_j, \boldsymbol{\omega}_j) \triangleq \mathbf{b}_{m,i} - \mathbf{b}_{b,j} - \mathbf{b}_{n,j}$ at time t_j,

$$\begin{aligned} \delta\mathbf{p}_{jk} &= \frac{1}{2}\mathbf{a}_j \delta t^2 \\ \delta\mathbf{v}_{jk} &= \mathbf{a}_j \delta t \\ \delta\mathbf{q}_{jk} &= \mathrm{Exp}(\boldsymbol{\omega}_j \delta t) \end{aligned} \tag{3.16}$$

and write the integration (3.15) as the composition

$$\Delta_{ik} = \Delta_{ij} \boxplus \delta_{jk} \tag{3.17}$$

described in (3.13). Typically, we take the biases at the keyframe time t_i, that is, $\mathbf{b}_{b,j} = \mathbf{b}_{b,i}$. In the following, we will identify Δ with the pre-integrated delta, and δ with the current delta.

3.3.4 Jacobians

We note all Jacobians with $\mathbf{J}_x^y \triangleq \partial y / \partial x$ and refer the reader to Appendix C.1 for details on the development of all non-trivial Jacobian blocks in this section.

3.3.4.1 Jacobians of the body magnitudes

from Eq. (3.10) we have:

$$\mathbf{J}^{\mathbf{b}}_{\mathbf{b}_m} = \mathbf{I}_6 \qquad \mathbf{J}^{\mathbf{b}}_{\mathbf{b}_b} = -\mathbf{I}_6 \qquad \mathbf{J}^{\mathbf{b}}_{\mathbf{b}_n} = -\mathbf{I}_6 \ . \tag{3.18}$$

3.3.4.2 Jacobians of the current delta

We have from (3.16),

$$\mathbf{J}^{\delta_{jk}}_{\mathbf{b}_j} = \begin{bmatrix} \frac{1}{2}\mathbf{I}\delta t^2 & \mathbf{0} \\ \mathbf{I}\delta t & \mathbf{0} \\ \mathbf{0} & \mathbf{J}_r(\boldsymbol{\omega}_j \delta t)\delta t \end{bmatrix} \in \mathbb{R}^{9\times 6} \tag{3.19}$$

where we develop the lower-right block as in Appendix C.1.5.1.

3.3.4.3 Jacobians of the delta composition

We differentiate the delta composition (3.17) described in (3.13),

$$\mathbf{J}^{\Delta_{ik}}_{\Delta_{ij}} = \begin{bmatrix} \mathbf{I} & \mathbf{I}\delta t & -\Delta\mathbf{R}_{ij}\left[\delta\mathbf{p}_{jk}\right]_\times \\ \mathbf{0} & \mathbf{I} & -\Delta\mathbf{R}_{ij}\left[\delta\mathbf{v}_{jk}\right]_\times \\ \mathbf{0} & \mathbf{0} & \delta\mathbf{R}_{jk}^\top \end{bmatrix} \in \mathbb{R}^{9\times 9} \tag{3.20a}$$

$$\mathbf{J}^{\Delta_{ik}}_{\delta_{jk}} = \begin{bmatrix} \Delta\mathbf{R}_{ij} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \Delta\mathbf{R}_{ij} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \in \mathbb{R}^{9\times 9} \tag{3.20b}$$

where $\Delta\mathbf{R}_{ij}$ and $\delta\mathbf{R}_{jk}$ are the rotation matrix deltas corresponding to the respective quaternion deltas $\Delta\mathbf{q}_{ij}$ and $\delta\mathbf{q}_{jk}$. We develop all the non-trivial blocks as in Section C.1.5.2 and Section C.1.5.3.

3.3.5 Incremental delta covariance integration

Let $\mathbf{Q}_\Delta$ be the covariance of the pre-integrated delta, and $\mathbf{Q}_n$ the one of the measurement noise. For convenience, we first compute the covariance of the current delta,

$$\mathbf{Q}_\delta = \mathbf{J}^\delta_{\mathbf{b}_n} \mathbf{Q}_n \, {\mathbf{J}^\delta_{\mathbf{b}_n}}^\top , \tag{3.21}$$

where $\mathbf{J}^\delta_{\mathbf{b}_n} = \mathbf{J}^\delta_{\mathbf{b}} \cdot \mathbf{J}^{\mathbf{b}}_{\mathbf{b}_n}$ is the noise Jacobian, obtained with (3.18–3.19) and the chain rule. The delta covariance is then integrated with

$$\mathbf{Q}_{\Delta_{ik}} = \mathbf{J}^{\Delta_{ik}}_{\Delta_{ij}} \mathbf{Q}_{\Delta_{ij}} {\mathbf{J}^{\Delta_{ik}}_{\Delta_{ij}}}^\top + \mathbf{J}^{\Delta_{ik}}_{\delta_{jk}} \mathbf{Q}_\delta {\mathbf{J}^{\Delta_{ik}}_{\delta_{jk}}}^\top , \tag{3.22}$$

using Jacobians (3.20), and starting at $\mathbf{Q}_{\Delta_{ii}} = \mathbf{0}_{9\times 9}$.

3.3.6 Delta correction with new bias

Let $\overline{\Delta}$ and $\overline{\mathbf{b}_b}$ be respectively the pre-integrated delta and the bias values used during pre-integration. Since the bias estimates change at each iteration of the optimiser, we need to update the delta according to the new bias values $\mathbf{b}_b$. We do

so with the linearised update,

$$\Delta = \overline{\Delta} + \mathbf{J}^{\Delta}_{\mathbf{b}_b}(\mathbf{b}_b - \overline{\mathbf{b}_b}) \ , \tag{3.23}$$

where $\mathbf{J}^{\Delta}_{\mathbf{b}_b}$ is the pre-integrated bias Jacobian, computed incrementally using the chain rule,

$$\mathbf{J}^{\Delta_{ik}}_{\mathbf{b}_b} = \mathbf{J}^{\Delta_{ik}}_{\Delta_{ij}}\mathbf{J}^{\Delta_{ij}}_{\mathbf{b}_b} - \mathbf{J}^{\Delta_{ik}}_{\delta_{jk}}\mathbf{J}^{\delta_{jk}}_{\mathbf{b}_b} \ . \tag{3.24}$$

with $\mathbf{J}^{\delta_{jk}}_{\mathbf{b}_b} = \mathbf{J}^{\delta_{jk}}_{\mathbf{b}}\mathbf{J}^{\mathbf{b}}_{\mathbf{b}_b}$. This Jacobian starts at $\mathbf{J}^{\Delta_{ii}}_{\mathbf{b}_b} = \mathbf{0}_{9\times 9}$.

3.3.7 Residuals

The computation of the residuals for the IMU delta factors (see Fig. 3.2, green) requires: the state estimates $\mathbf{x}_i$ and $\mathbf{x}_j$; the current bias estimates $\mathbf{b}_{b,i}$; the pre-integrated delta $\overline{\Delta_{ij}}$; the bias used during pre-integration $\overline{\mathbf{b}_{b,i}}$; and the pre-integrated bias Jacobian $\mathbf{J}^{\Delta_{ij}}_{\mathbf{b}_b}$. The process is best understood if split into smaller steps: we first compute a corrected delta Δ_{ij} using (3.23); then we compute a predicted delta $\widehat{\Delta}_{ij}$ using (3.12); and finally we compute the residual with

$$\mathbf{r}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{b}_{b,i}) = \begin{bmatrix} \Delta\mathbf{p}_{ij} - \widehat{\Delta\mathbf{p}_{ij}} \\ \Delta\mathbf{v}_{ij} - \widehat{\Delta\mathbf{v}_{ij}} \\ \mathrm{Log}(\widehat{\Delta\mathbf{q}}^{*}_{ij} \otimes \Delta\mathbf{q}_{ij}) \end{bmatrix} \in \mathbb{R}^9 \ . \tag{3.25}$$

Its information matrix is given by $\mathbf{\Omega} = \mathbf{Q}^{-1}_{\Delta ik}$.

3.4 Pedestrian Localisation

3.4.1 Context

Localisation of pedestrian in indoor environment remains an open problem. A cheap and reliable sensor in this context is the inertial measurement units (IMU), carried by the pedestrian while he/she is walking. However, due to the bias of both the accelerometer and the gyroscope, integrating the inertial measurements directly leads to tremendous drifts, as the state of the system (position, orientation, velocity, bias) is not fully observable. We consider the specific case where an IMU is attached to one of the pedestrian's feet. We exploit specific prior knowledge (*i.e.* the fact that the foot lands at zero velocity during full-contact phase (Fig. 3.5)) in order to make the full state of the IMU observable. The inertial measurements and these prior knowledge are gathered in a graphical model (a factor graph), and are exploited to build a maximum-likelihood estimator. We validate these concepts on several long-range trajectories captured with human subjects from a dataset provided by the literature [Angermann 2010] and compare the results with ground-truth measurements (coming from a motion capture system) and previous results of the state of the art. This study verifies the validity of our method while not

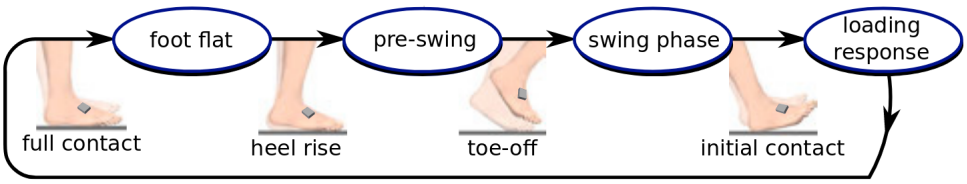


Figure 3.5: Cyclic phases of foot during walking (source: [Schauer 2016]).

relying on any new hardware configuration. Indeed, running tests directly on a humanoid robot may introduce some undesirable effects with respect to the IMU or specificities of generated motions.

3.4.2 Related Work

Pedestrian localisation using a foot-mounted IMU was first introduced in [Hutchings 1998]. Since then, many developments have been made to find alternative ways to accurately localise people. Pedestrian Dead Reckoning (PDR) methods (also known as Personal Navigation Devices), make use of one or more IMU installed on the body of the subject. The main idea of PDR techniques is to integrate inertial measurements with Zero Velocity Update (ZUPT) constraints to reduce errors [Ojeda 2007]. This work is extensively used in IMU-based human localisation works and various fields [Kwanmuang 2015] analyses the gait of a walking person with PDR method to estimate the direction of the shoe, thus the walking direction, and measure stride length.

Shoe-mounted IMU is still considered as a possible way to accurately localise persons in an indoor environment due to lower drift errors when compared to body-mounted solutions [Groves 2007] . One way to understand why foot-mounted IMU is preferred to other alternatives may be given by [Groves 2007] comparing body-mounted and foot-mounted based PDR methods. Both systems had similar performances considering the position error, being lower than 10 m for 60 seconds experiments. However, the foot-mounted shown drifts as results were compared to GPS ground truth. We should note that the body-mounted method proved to be usable in not only walking cases but also when the human agent was jogging or running but with a decrease in terms of accuracy.

Various strategies can be considered to improve the localisation results of foot-mounted IMU navigation. To achieve this goal, one way to consider is the use of one or more IMU and define some special constraints. The main advantage of this choice is that the solution would be easily wearable and thus usable. As shown in [Kourogi 2010] and [Panahandeh 2012], PDR can be used to recognise the action being carried out by the pedestrian through classification methods, but adding this contextual information is also a way to reduce PDR localisation errors ([Kourogi 2010]). [Wagstaff 2017] is not only using this contextual information thanks to the training of a support vector machine (SVM) classifier using IMU data, but also making efforts on finding optimal zero-velocity detection parameters taking into

account a specific user and motion types. Prior information can be exploited when merging the measurements of several IMUs, for example relative to the maximum step length the pedestrian could do when using two foot-mounted IMUs [Skog 2012]. Thus the inequality constraints limit the distance between both IMUs to give better position estimation results when compared to single foot-mounted IMU solution.

Fusion strategies are also used to overcome the drift observed in methods using IMU and to obtain positioning errors of approximately 1.5 m [Ruiz 2012]. In [Chdid 2011], a foot-mounted IMU is fused with a waist-mounted visual odometry system to update the state of the system composed of its position, velocity and acceleration.

More information can be structured into a map following a SLAM approach [Angermann 2012], leading to a bounded error growth to 1 metre. This FootSLAM uses dynamic Bayesian network and loop-closure strategies. The idea exploited here is to use the normal human behaviour consisting in relying on visual information to guide the motion and avoid obstacles. FootSLAM's idea gave birth to several variants for use in different conditions or slightly different purposes ([Puyol 2012, Bruno 2011]). Hardegger et al. use contextual information to body-mounted IMUs with a FastSLAM-base implementation in ActionSLAM, landmarks being location-related actions [Hardegger 2012]. A foot-mounted IMU is then used not only for inertial navigation purposes but also as a landmark observation system through action recognition strategies by applying machine learning techniques for motion classification. The main idea behind ActionSLAM is to consider that some specific actions are done only at some specific locations on the map. Results tend to show that using both foot-mounted and wrist-mounted IMUs is giving results that are robust enough for indoor applications.

Previously cited examples tend to show how important it is for pedestrian inertial navigation system to be able to deal with the localisation drifts due to the integration of IMU's data. Two different methods have been aforementioned to reach this goal: fusion strategies using different complementary sensors and IMU only based methods using specificities of human behaviour (walking patterns, action recognition, contextual information).

From this analysis, we see that two important aspects have been investigated to solve pedestrian localisation: i) exploiting some specificities of human behaviour with the inertial measurements as prior knowledge and ii) fusing IMU with additional complementary sensors. We show that using a graphical model is a sane and efficient way to encode prior knowledge about the human behaviour (horizontal foot during zero-velocity phases). An additional feature of our approach is that it is easy to extend the graphical model, either with additional prior knowledge, or with measurements coming from additional sensors. For example, fusing absolute but noisy measurements like GPS, RFID or RSSI would be straight forward.

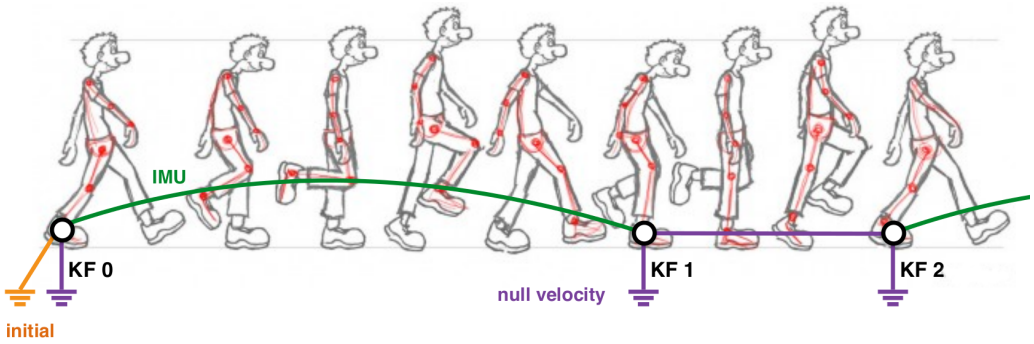


Figure 3.6: Representation of the method used for pedestrian tracking.

3.4.3 Experiments

Two experimental setups are proposed. First, we use the dataset made available in [Angermann 2010] to validate our method on low-drift IMU and compare our results to those attached to the dataset and provided with a state-of-the-art Kalman filter. Our method is applied on several scenarios from this dataset and we present typical results obtained with two representative cases. The first sets of data correspond to a human walking back and forth while the second example is related to a walking pattern describing a eight-shape. Both IMU and motion capture data are provided at 100 Hz.

Then, we will investigate the use of additional sensors to demonstrate the feasibility of fusion strategies using a low-cost IMU running at 1 kHz (Invensense's MPU6050 [InvenSense]).

3.4.3.1 Method

Keyframes are created at the beginning and ending of each support phase of the selected foot according to zero-velocity instants detection provided by the dataset (Fig. 3.6). The zero-velocity instants are often called ZUPT standing for 'Zero velocity UPTdate'. Factors active in the graph (see Fig. 3.2) are: initial position, yaw and velocity set to 0, minimal bias drift; and IMU pre-integration. The only constraints applied on the first key-frame are related to the bias magnitude of the IMU and zero velocity constraint factors. All the graph is optimised after each keyframe creation so that these estimates can be used for future estimations with new keyframes. Here key-frames are added at the beginning and the end of each contact phases. Zero velocity constraints are applied as an estimated prior for the optimiser by setting the velocity to 0, meaning that corresponding variables will still be considered as parameters during the optimisation process. In the following experiments, the variance of the velocity is arbitrarily fixed to a non-zero small value. Indeed, if we use a constraint with 0 variance (*i.e.* the corresponding quantity is not optimised by the solver as a decision variable but is an invariant of the problem) then we must be super confident with this information. Otherwise we

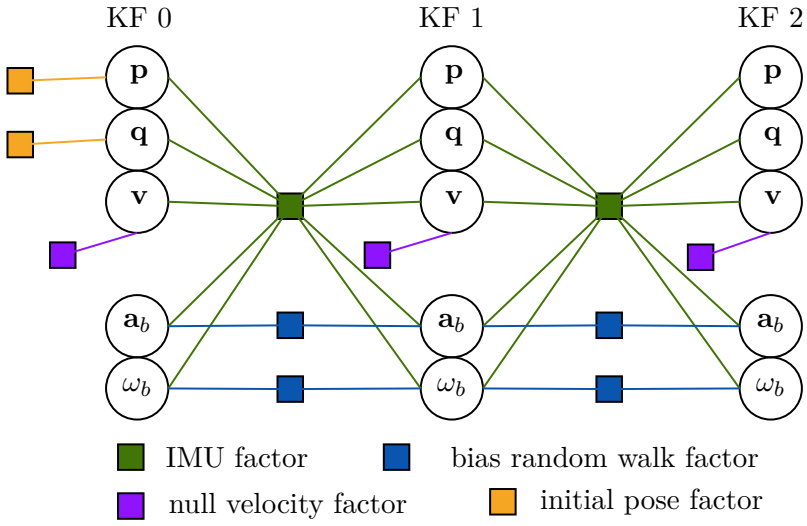


Figure 3.7: Visualisation of the graph in the pedestrian tracking case using only zero-velocity detection. The same graph can be used by adding other factors due to information provided by other possible sensors.

would be imposing a constraint inducing a wrong estimate of the trajectory.

Finally, we define motion factor. In a first set of results, we used the basic motion prior (minimal velocity and biases variation). In a second time we show the interest of data fusion by emulating a fake sensor that would measure additional data during the flying phase, allowing the creation of an additional key frame between contacts. We use motion capture recording to create this additional key-frame. Such information might be provided using biological evidence on humans (*e.g.* using the trajectory of the foot during human walking). But it is mostly a proof of concept of similar experience with the humanoid robot, where kinematic information (measured by the encoders) would provide the same information. As in the usual ZUPT-aided inertial navigation, zero-velocity constraints are imposed only on keyframes created during contact phases. Kinematic odometry is also added between all consecutive keyframes for this experiment (see Fig. 3.2). The optimisation part is currently handled by Google Ceres optimiser [Agarwal] using the sparse structure of the problem.

3.4.3.2 Results

State estimation using ZUPT only

With only IMU measurements, the system is not provided with enough information to be able to precisely estimate the vertical orientation of the IMU due to its non-observability from the IMU measurements directly. As a consequence, the estimation of the state of the foot is not able to converge to its real value. In other words, the foot trajectory can be recovered up to a rotation around the gravity axis.

In the following, we manually fixed initial translation and rotation to align the results with the motion-capture reference in order to compare our results against the ground-truth measurements. Our system is able to estimate the state of the system to precision close to the state-of-the-art 9-state Kalman filter [Foxlin 2005] as shown in Fig. 3.8. Furthermore, we can notice that the level of accuracy is similar in both cases. The initial error jump of the kalman estimate is likely to be caused by the initialisation procedure. Indeed, the experiments starts and ends by hitting the floor with the foot the IMU is attached on, thus resulting in high dynamic variations in the IMU and quick change for foot's pose as given by the motion-capture system. This allows to manually shift the data and work with coherent timestamps. However, the data resulting from this synchronisation procedure is not removed for the processing part although it introduces outliers. We performed a very simple outlier rejection by deleting these synchronisation data. Results of bias estimation using another dataset provided by the DLR are given in Fig. 3.9.

Besides, the experiment shows satisfying results in terms of computational times. Indeed, 3.1 seconds are enough to integrate all the data corresponding to a 50 seconds experiment, that is more than 5000 IMU data, to build and to optimise the graph each time a keyframe is created with our framework, with a total of 80 keyframes. It would be hard, if not currently impossible, to get such results using the IMU without a proper pre-integration method. All in all, it takes $98\mu s$ in average to read and integrate a single IMU measurement including the computation of jacobians.

State estimation using ZUPT and sensor fusion

The strength of the optimal estimation can also be found in the fusion strategies. Fig. 3.10 shows the reconstructed foot trajectories for the two cases: without and with the flying keyframe information. Adding kinematic information between keyframes enhances the observability of the system and allows to a better estimation. A major advantage of IMU pre-integration theory is the ability to use past pre-integrations corrected with current estimates of keyframes and biases as simply as it takes to integrate a single IMU data. This removes the need of linearisation of intermediate IMU data between previous and current estimates of the state on which the integration must be performed. We also notice that the use of a flying keyframe makes the bias estimation more stable.

3.4.4 Conclusion

We have revised the IMU pre-integration theory [Forster 2017], and proposed an implementation in the quaternions manifold, with simpler derivations than previous works, and with physical interpretations, which we believe go in the direction of improving the clarity of the method. This method has been applied to pedestrian navigation with an IMU attached to the foot and by exploiting available knowledge extracted from the gait phases, such as zero velocity and IMU bias dynamics. As

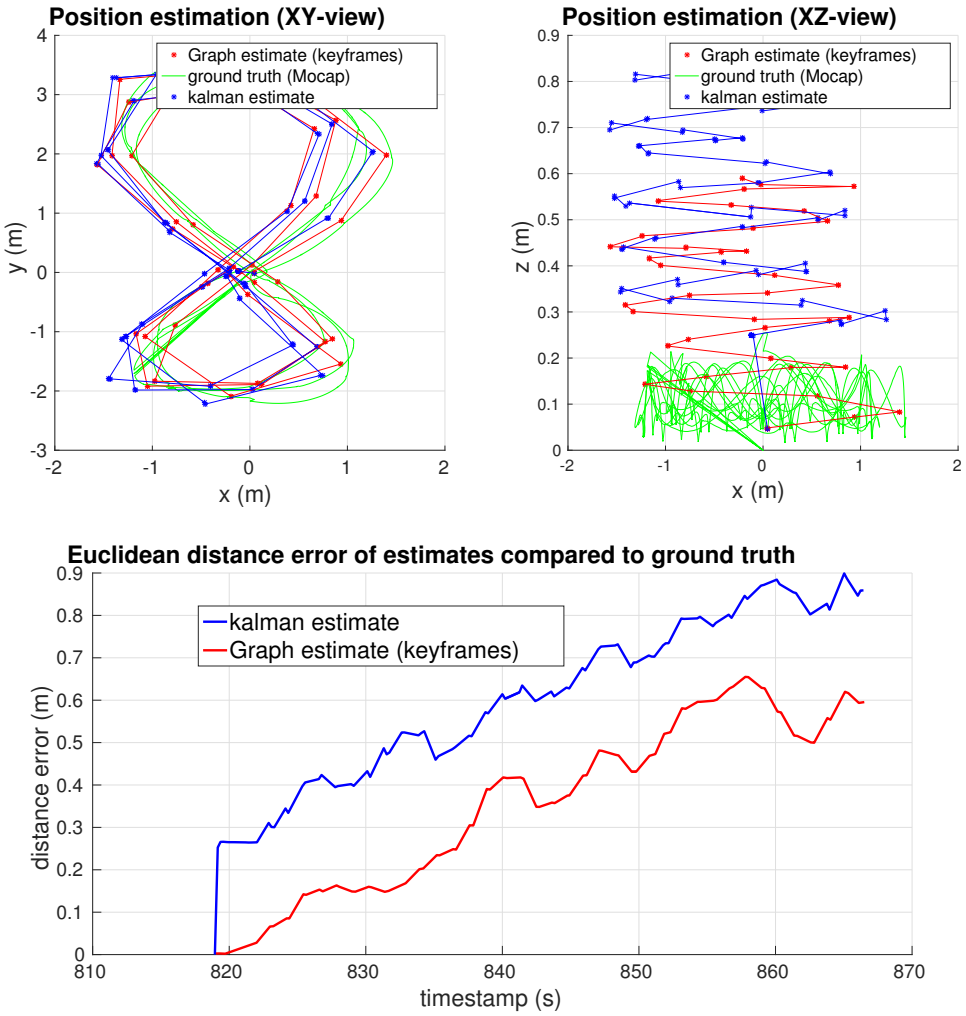


Figure 3.8: **Top**: trajectory estimation during human 8-shaped walking phases with an IMU attached to a foot. Stars in blue and red colours are respectively the reference estimated states provided in the dataset and the estimation with our approach. The continuous green line is the ground truth as given by the motion capture system. **Bottom**: Evolution of Euclidean distance errors between estimates and the ground truth for this 8-shaped walking trajectory.

opposed to usual methods using filtering strategies, we take advantage of the power of nonlinear optimisation techniques based on factor graphs while estimating the pose of the IMU in real-time. Results showed that this estimation method is able to properly estimate the bias, then leading to an accurate odometry where the drift remains reasonable, even after minutes of integration. The method easily extends to additional prior knowledge or additional sensors. Further work is needed to make the system less critical to wrong ZUPT detections. To achieve this goal we can change the ZUPT implementation strategy from a zero prior and arbitrarily low variance to a computation of the variance that depends on the information provided

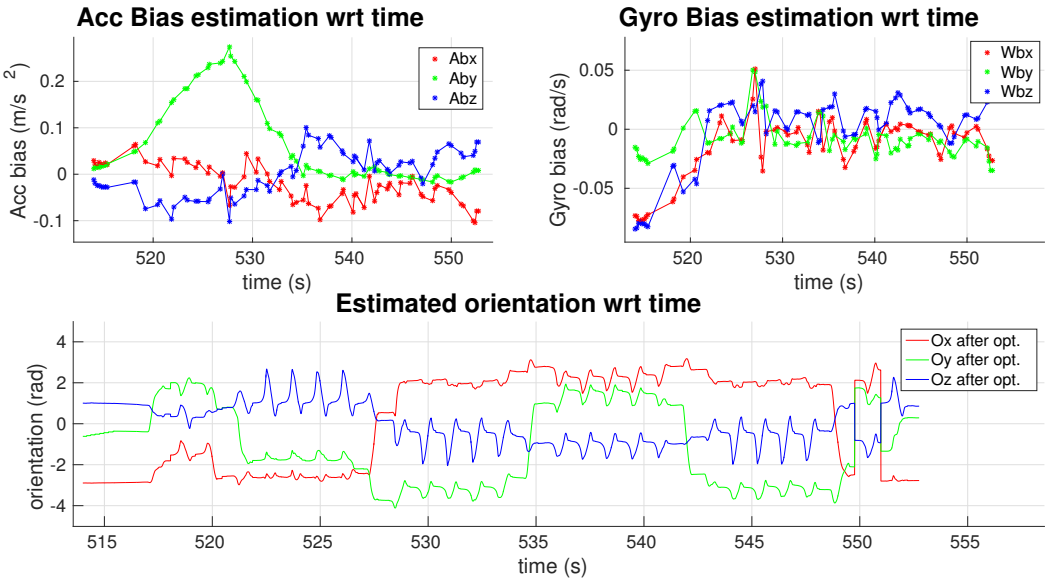


Figure 3.9: Evolution of estimated IMU biases and orientation of the device during a curved walk. Biases are estimated only for keyframes and considered as constant between them.

by available sensors.

3.5 Experiments on a humanoid robot

3.5.1 Introduction

While the feet of many of our legged robots are equipped with force sensors, we often only have a rough idea of their exact location in space, neither of the dynamic effects acting on them (*e.g.* flexibility or inertia). Inaccuracy of the foot location, trajectory or friction often results in rapid and unavoidable falls. We consider the problem of accurately estimating the placement of the foot, either during the contact phase (to assert friction and no sliding) or during the flying phase (to compensate for flexibility in the actuation chain). We propose to attach an inertial measurement unit (IMU) directly on the foot in order to reconstruct its position. The main difficulty is then to recover the observability of the IMU location by integrating all previous sensor measurements along with additional constraints coming from the contact and the kinematic chain. A graphical representation approach is used to integrate IMU measurements, kinematic and contact information on a humanoid robot. Contrary to many state estimators recently proposed in the context of legged locomotion, we propose to rely on an information graph representation to handle observability consistency and other constraints. Since an efficient use of the IMU data is of major importance to prevent the graph from growing too much and to use the available information to estimate the biases online, we use the pre-integration methods on manifolds. We validate these concepts by estimating the trajectory of

the foot of the HRP-2 humanoid robot during various gaits, and show that we are able to accurately reconstruct some subtle effects, such as sliding effects during the contact and flexibility of the kinematic chain.

There are significant differences between this application and the estimation for pedestrian navigation purposes. The first difference lies in the IMU measurements. Indeed, the textile of the shoe the IMU can be attached to can act like a low-pass filter eliminating possible high frequency vibrations. In contrast, the IMU measurements from a sensor attached to the foot of a robot measures the vibrations of the robot. These vibrations are distributed from the different motors to the entire structure of the humanoid robot. These vibrations can turn out to be problematic when considering especially high frequency estimation at up to 1 kHz. Such problems need to be investigated when considering an online experiments working as a standalone application.

3.5.2 Experimental setup

We acquire data on the humanoid robot HRP-2 (Fig. 3.1) during different scenarii with a low-cost IMU placed on its right foot. Before describing the results, we quickly review the properties of the hardware and specify the precise graphical model.

3.5.2.1 Hardware

The humanoid robot HRP-2

The HRP-2 humanoid robot (Fig. 3.1) is embedded with various sensors including a camera (in the head), an IMU (in the chest) and force and torque sensors in end-effectors. Due to its mechanical design, the robot has flexibilities between each foot and the ankle it is linked to (Fig. 3.11). These flexibilities are used to absorb some impacts during walking phases for example [Kanehira 2002]. The presence of such flexibilities are changing the kinematics of the robot's free flyer defined as the body from which the robot's pose in its environment is described. As a result, the estimation of the foot of the robot using the kinematic chain will not take into account these flexibilities possibly inducing wrong orientation estimates for example.

Besides, humanoid robots are usually considered to be rigid and their interactions with the environment are modelled as rigid contacts. Such a model is convenient on a numerical point of view and enables one to simplify the dynamics so that the control itself is made easier. However, the complete consequences of the introduction of flexibilities on such a complex machine have yet to be understood. It is a possibility that such mechanical specificities, despite the fact that they can be useful, introduce undesired effects. If sensor measurements are one possible source of possible errors, the interactions with the environment can also induce some undesired behaviours such as slippage. Finally, the complexity of the machine itself makes its a control a non-trivial task and a lot of effort is put on

improving the robustness and the accuracy of the control to have all the encoders of the robot in a specific configuration. These encoders are used to measure the joint position of each degree of freedom of the robot. Depending on the sensors, the measurements can be either given in an absolute frame or the information can be relative in incremental encoders. The differences have specific implications on the use of the sensors and their accuracy.

Force and torque sensors

The sense of touch is provided to humans through a complex system of nerves under the skin. This information is extremely important in our everyday life to detect contact through forces applied on the skin and thus interact with the environment. The development of a robotic skin is an active field of research but these sensors are not equipped on robots yet. Instead, 6-axis force and torque sensors (FT sensors) are usually embedded in humanoid robots at each contact points that are the feet and the hands. The use of these sensors is important to correct the balance of the robot either during walking or in static phases and more generally to give feedback to the robot about the end effector itself. A 6-axis FT sensor can measure both force and torque along 3 axis. The main technology used in these sensors is based on a deformable material with strain gauges sensitive to deformations. The strain gauges allow to transform the deformation into a variation of electrical resistance. The output of these gauges are only of a few millivolts and the quality of the sensor is strongly impacted by the quality of the conditioning and the amplifier electronics. FT sensors are usually characterised by the range of the measurements they can provide and their resistance to impacts. On the HRP-2 robot, the force sensors at the feet are protected by an elastic element interposed between the sole and the sensors. Finally, FT sensors are also subject to measurement errors. Some of them can also incorporate temperature measurements to compensate for its effect on biases and scaling factors. The relationship between the forces and strain gauges is determined through a calibration processes inducing higher prices.

IMU

We use a MPU6050 6-axis IMU for data acquisition. The IMU is set to provide raw measurements at 1 kHz while mounted on the robot's right foot. IMU data acquisition is triggered by a STM32 microcontroller through I2C connection at 400 kHz. Finally, the microcontroller sends timestamped data to the computer via a USB connection (Fig. 3.12).

In order to investigate a simple and non-invasive estimation method, we simply attach the MPU6050 IMU on the top of the right foot of the robot. Some consequences arise from this choice. First of all, the transformation between the point in the robot considered as the foot end effector in the kinematic chain and the pose of the IMU is unknown. However, we know that this transformation is fixed and we can neglect this point in our experiments since we are more interested in comparing the pose estimations as given by the proposed method and through forward

kinematics estimation.

3.5.2.2 Method

Using the IMU and the force sensor

Since further effort would be required to run the estimation method online directly on the robot in an integration point-of-view into the robot's system, the estimation parts are realised offline. Timestamps of IMU and force sensors data are synchronised manually by hitting slightly the foot several times before the experiments. It is then possible to visualise the chocs on the IMU data and in the force and torque measurements to manually shift a set of measurements.

The forward kinematics estimates are provided using Pinnochio [Carpentier 2018] which is a C++ library for efficient rigid multi-body dynamics computations developed by the GEPETTO Team at LAAS-CNRS. The use of a motion capture (MOCAP) system is considered to provide ground truth displacement measurements and from one point to another.

Forward Kinematics on HRP-2 Robot

Since the experiments are conducted offline, we deduce the kinematic constraint to be added in the factor graph from robot odometry. This odometry can be built using forward kinematics. Standard forward kinematic methods propagate the joint angle information through the kinematic chain of the robot from the free flyer to estimate the position of any joint of the robot in space. To this end we use Pinocchio software implementing these methods explained with more details in [Featherstone 2014]. However, in our case, the free-flyer of the robot is not estimated although the robot moves. Hence if we simply use this method through time to reconstruct the position of the foot then the results would be the same as if the robot walked in the air while its base was fixed. As a consequence, we estimate the position of the robot's right foot with respect to to the left foot and versa while fixing the position of the support foot. The modification of the support foot is realised when a foot was in the air then finally hits the ground. The determination of the new support phase is realised through a hysteresis filter based on the values measured by force sensors in both feet.

Key-frame creation policy

The IMU is used along with the force sensors and the foot pose estimation through the kinematic chain to determine instants with zero velocity. This approximation is possible due to the fixed transformation between the IMU and the rigid body (*i.e.* the foot in our case). Finally, we can investigate several key-frame creation policies depending on the chosen strategy. Fig. 3.13 and Fig. 3.14 show a possible solution given the hypothesis that the slippage occurs during the contact phase when the foot hits the ground. Thus the hypothesis of reliable kinematic reconstruction before hitting the ground can be used to create an odometry constraint during the foot's flying phase.

3.5.2.3 Scenarii under investigation

We describe hereafter the different scenarii the proposed method is evaluated in.

Walking in place

The first motion investigated is a pattern test that is used for calibration purposes. The input control sequence is here supposed to make the robot walk in place while the position and orientation of the feet are not supposed to change. In practice, the robot is not able to follow precisely the desired trajectory and slight changes in both position and orientation can be observed (Fig. 3.15). Although the cause for this error is not completely identified, we suspect that the inner stabiliser running in the robot indirectly changes the trajectory of the robot. This effect may be intensified by small differences between the kinematic model of the robot and its actual design.

Fig. 3.16 and Fig. 3.18 provide estimated states for the IMU with respect to time during the experiment. These figures are to be interpreted along with Fig. 3.15 to understand the results. Indeed, Fig. 3.16 highlights possible drifts for P_x and P_y but a P_z estimate that remains close to the initial pose of the IMU. This behaviour is coherent with the experiment itself as it can be seen in Fig. 3.15. The final pose of the foot is slightly different of the initial one (for both position and orientation). The difference between both poses can be interpreted as a translation in X and Y axis and a rotation around the tilt, *i.e.* Oz axis. The change in Oz orientation is also observed through the estimation of the IMU states in Fig. 3.18 while both Ox and Oy orientations ultimately remain unchanged when comparing the initial and final states. It is also relevant to consider the change in the estimated orientation as it is particularly visible on the Oy estimate of Fig. 3.18 during full contact phases considering that the foot is supposed to be still. One hypothesis that may explain this change is to consider wrong bias estimates. There are several properties that could strengthen this hypothesis. First of all we should note that the foot gait cycle does not imply any rotation along this Oy axis. This specificity may render the bias estimation hard due to non-excited axis in the IMU and hence no full-observation of the dynamics of the sensor. Another property may be the use of the zero-velocity constraints on the states of the IMU. The variance of this constraint needs to be small enough to actually add some relevant information in the graph and use it to characterise the biases. However, setting the variance wrongly to a too small value may cause undesired behaviours on the system (including the estimation of the biases in our specific case). The zero-velocity constraint as it is usually defined is a constraint on the instantaneous velocity of the IMU at timestamp t. The use of this specific zero-velocity constraint on the state of the IMU is problematic due to mechanical vibrations measured by the IMU and propagated through the structure of the robot itself. These vibrations should not be ignored and are visible in Fig. 3.16.

Walking forward

The second experiment that is considered is a walking forward motion on a plane

floor. Again, observations show differences between the desired and actual final positions and thus in the motion generated by HRP-2. Currently used methods tend to use visual perception in order to relocalise the robot in its environment and correct this drift due to the motion [Fallon 2014, Hartley 2018]. However, a simpler high rate estimation method would be beneficial to close the control loop of the robot locally. It could also be used to observe motions that cannot be estimated by the camera in the head of the robot or by the kinematic chain relying on the model of the robot and where all intermediate joints between the end-effector and the base of the robot are possible sources of errors.

Given results in Fig. 3.19 and Fig. 3.20, we can note that it is feasible to estimate the motion of the IMU. P_x and P_y are repectively slightly under 1 and above 0 metre. This is due to rotations of the robot during walking around the yaw axis. However, P_z should be closer to 0. Since our motion capture system is not able to correclty track the pose of the foot or of the IMU itself, it is difficult to compare those results to a ground truth. In Fig. 3.20 one can see again that the IMU is affected by vibrations due to the motion (hitting the floor).

From previous experiments (*i.e.* walking in place and forward), one can already raise some concerns about installing the IMU on the end-effector of the robot and estimate its pose. Although the problems may be due to the use of the zero-velocity constraint, the pose estimation is giving coherent results. It would be interesting to test the same approach used until there on advanced experiments. However, some strategies to bypass the use the zero-velocity constraints need to be investigated.

Due to time constraints and mechanical failure in the HRP-2 robot, these two following experiments (*i.e.* walking alonf the quarter of a circle and climbing stairs) could not be tested. However, they still remain interesting and raise specific challenges in practice.

Walking along the quarter of a circle

In an attempt to visualise the effect of slipping on the robot's motion, we intended to evaluate the proposed method while the robot walks describing the quarter of a circle. This specific motion induces specific constraints on the robot due to the rotation part in the movement. Preliminary experiments have thus shown important slipping phases. Whereas these phases cannot be observed with methods using odometry and forward kinematics only, the use of IMUs may be an interesting alternative to overcome the estimation problem in this context. This experiment is thus a way to characterise how interesting the proposed solution would be to face problems that are due to the dynamics of the motion and the rigidity of the robot. Furthermore, this experiment is a first step towards the design of a better slippage detection method.

Climbing stairs

Finally, we intended to investigate the foot pose estimation while HRP-2 is set to

climb stairs using a handrail. This is a complex motion with slippage phases and possible undesired contacts with the environment. These contacts can be due to not only the initial pose of the robot but also to its ability to follow the desired trajectory (*e.g.* the foot of the robot hitting the stair). The previous experiment was to test the slippage detection using data from the IMU, force sensors and the kinematic chain. This last scenario is more challenging since it implies not only slippage on the stairs (Fig. 3.1), but also collisions that should be detected by the three main sensors we use.

These last experiments may conduct to further studies amongst which one example is given hereafter. Indeed, the time window corresponding to the slippage is very short and it will be challenging to run the detection in realtime with sensors providing data at very different rates. Thus, it might be necessary to run the detection on past data and review the keyframe creation policy. This study would be a way to isolate the data corresponding to the slippage and still benefit from the velocity estimation and kinematic odometry information in the other parts of the trajectory for a more accurate estimation.

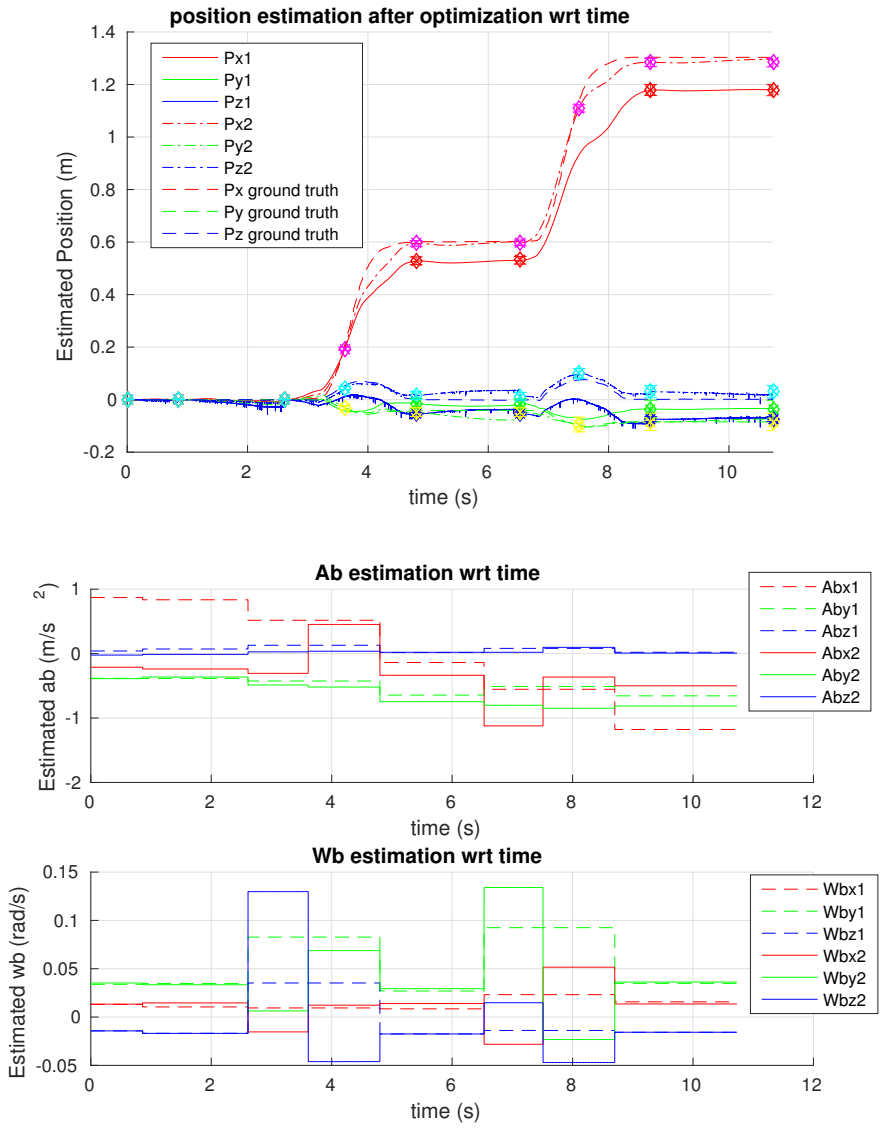


Figure 3.10: **Top**: trajectory estimation during human walking with an IMU attached to a foot. Continuous, dashed and dashed-dot lines are respectively: ground truth, estimation with zero velocity constraints only, estimation using zero velocity constraints and 1 odometry measurement during foot's flying phase. Odometry was here built from motion capture data. **Bottom**: Evolution of corresponding estimated IMU biases

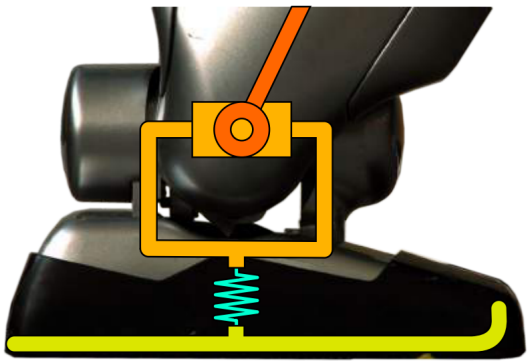


Figure 3.11: Representation of the flexibility between the robot's foot and ankle in HRP-2 [Mifsud 2017].

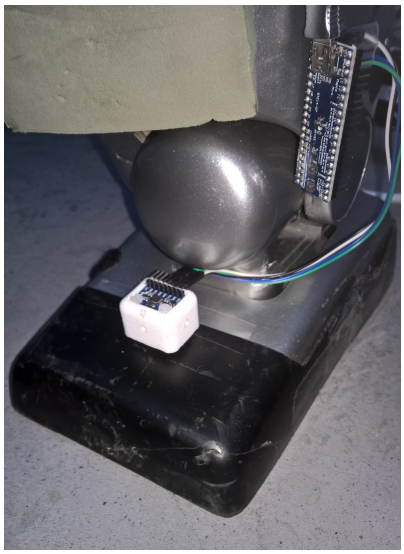


Figure 3.12: The IMU (in a white 3D-printed case) is fixed to the foot and connected to the STM32 microcontroller by I2C.

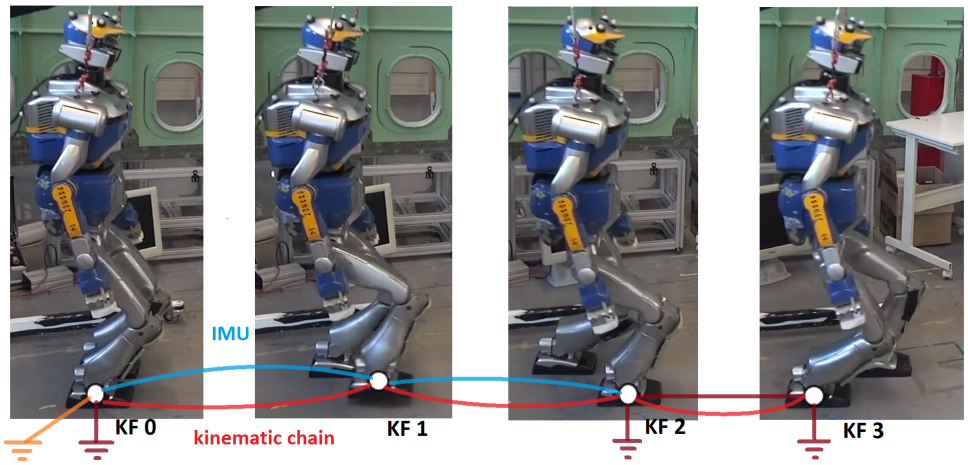


Figure 3.13: A quick insight to tests currently being conducted on the robot. **From left to right**: The experiment starts, the foot is on the ground with zero-velocity. Then right foot moves and we can create a key-frame while during the swing phase by using the robot's kinematic chain. Then the foot of the robot hits the ground, slippage may take place, thus, given the information from the sensors, we can decide whether to add a kinematic constraint between key-frames 1 and 2 or not. While the left foot moves, the right foot is supposed not to move.

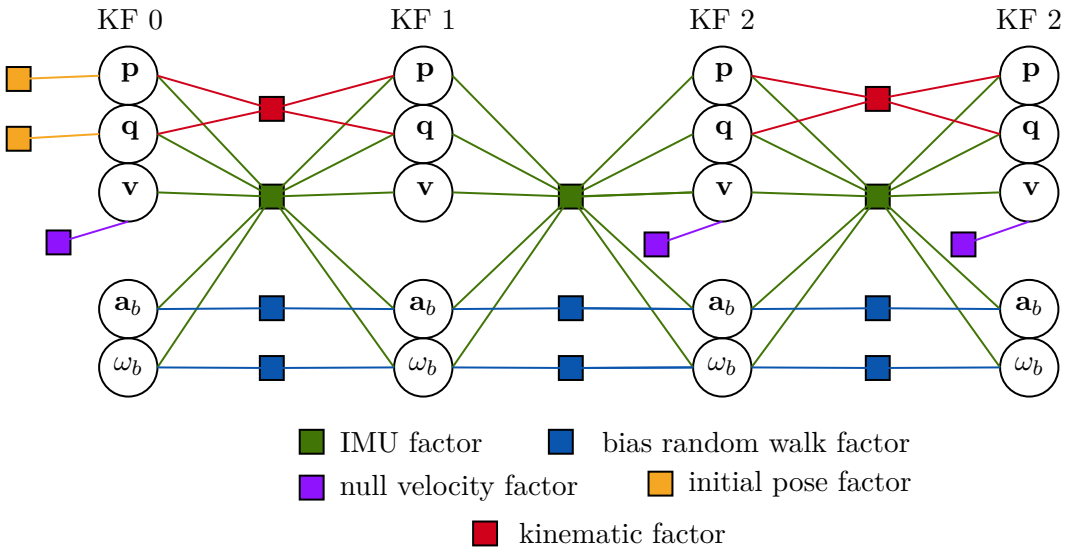


Figure 3.14: Possible factor graph from an experiment conducted with the robot. This case depicts the decision not to create a kinematic factor between key-frames 1 and 2. The key-frame created while the foot was in the air does not have any constraint on the velocity.

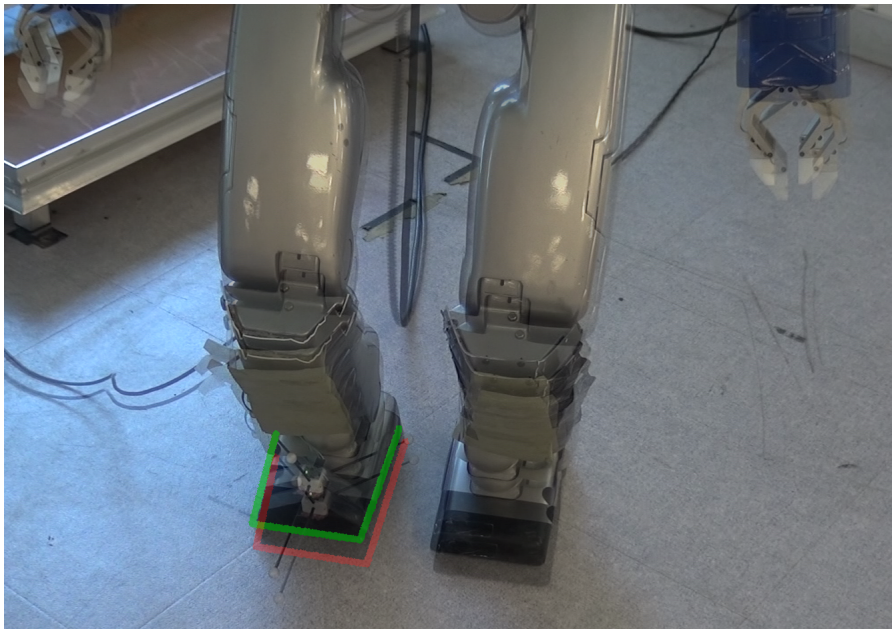


Figure 3.15: HRP2 drifts during the 'walk in place' test. We can notice the difference in both position and orientation between the initial and final poses after performing 5 steps with both right and left foots. The position drift is higher than the orientation one.

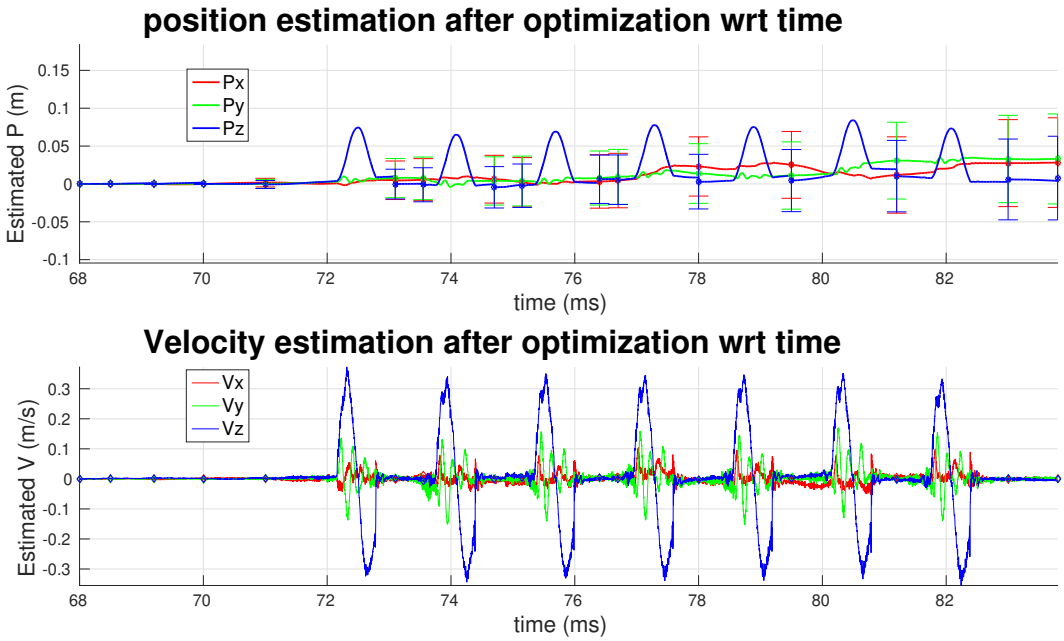


Figure 3.16: Position (**Top**) and velocity (**Bottom**) estimation of the foot of the HRP-2 robot while walking in place. Each diamond on the graphs represents a key-frame while bars under and over provide a way to visualise the $2\boldsymbol{\sigma}$ bounds around the estimates. We can notice that P_x and P_y drift away from 0. The final estimate along P_z axis is close to 0, which is coherent taking into account that the robot is walking on a flat floor. The variance of the zero-velocity constraint being fixed manually to a small value, its evolution is not represented here.

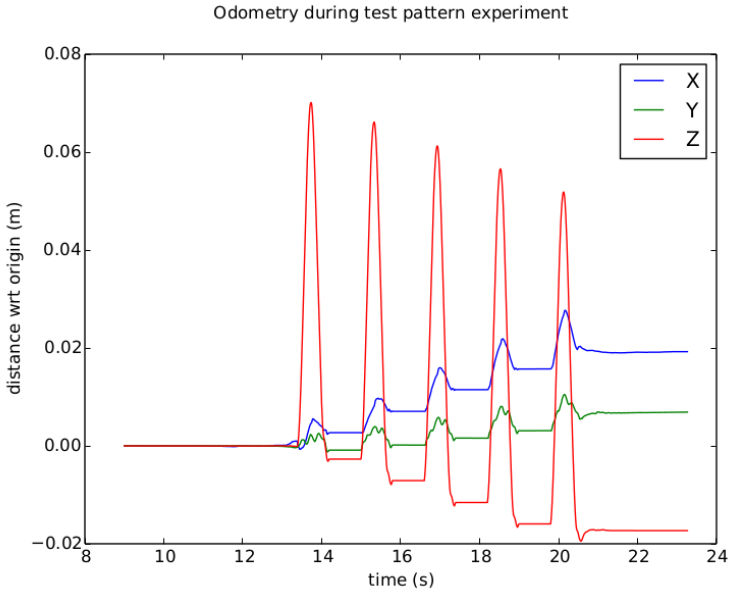


Figure 3.17: Position estimation of the foot of the HRP-2 robot while walking in place with kinematic odometry.

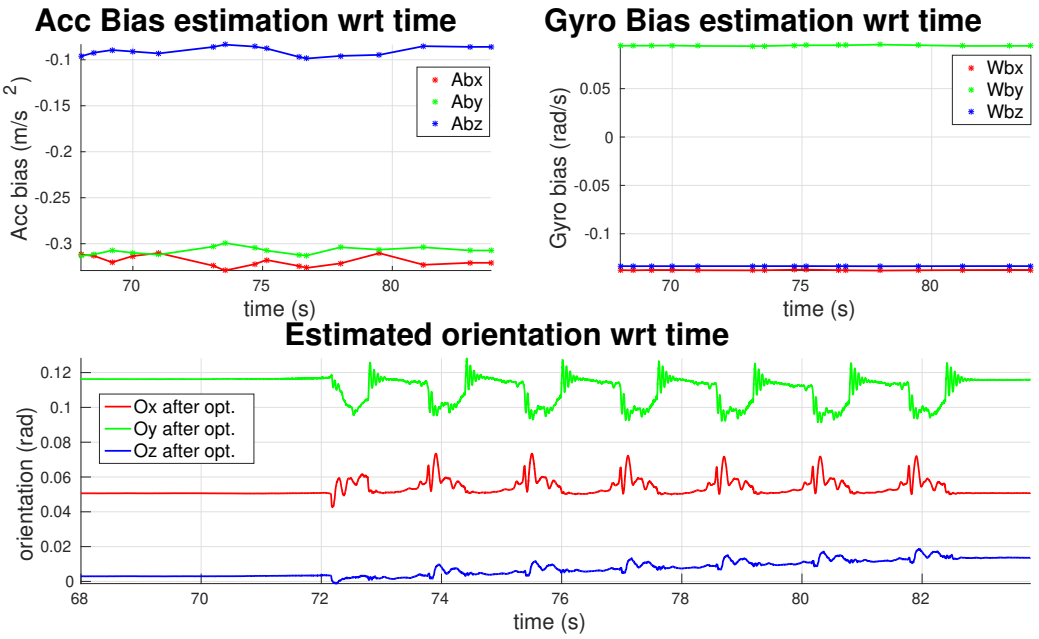


Figure 3.18: Estimation of accelerometer bias $\mathbf{a}_b$ (**Top Left**), gyrometer bias $\boldsymbol{\omega}_b$ (**Top Right**) and orientation (**Bottom**) of the IMU with respect to time while walking in place. Each diamond on the graphs represents a key-frame while bars under and over provide a way to visualise the $2\boldsymbol{\sigma}$ bounds around the estimates. Although it is difficult to evaluate the estimation of the biases, the stability of both accelerometer and gyrometer biases may tend to give the impression of a reliable estimate. However, this stability is due also to constraints imposed over the evolution of the biases through time. Furthermore, we note that the Oy orientation is slowly drifting during contact phases despite the fact that the foot is supposed to be still.

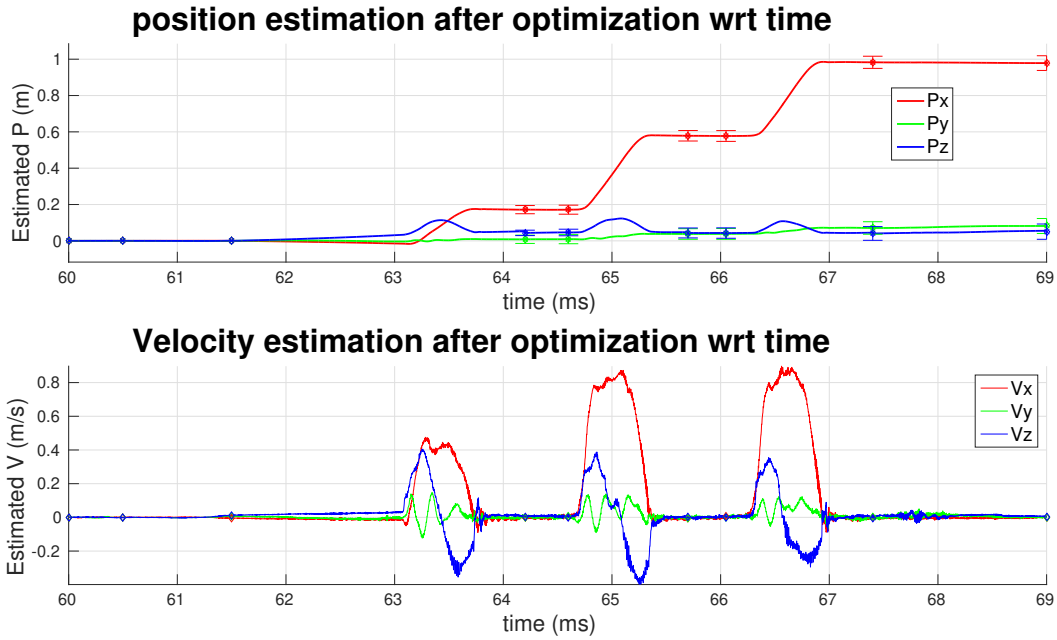


Figure 3.19: Position (**Top**) and velocity (**Bottom**) estimation of the foot of the HRP-2 robot while walking forward for 1 metre. Each diamond on the graphs represents a key-frame while bars under and over provide a way to visualise the $2\boldsymbol{\sigma}$ bounds around the estimates. We can notice that P_y and P_z drift slightly away from 0. The variance of the zero-velocity constraint being fixed manually to a small value, its evolution is not represented here. The results are coherent with the motion described by the robot for both P_x and P_y axis. However, Since we suppose the floor to be flat, P_z should be 0.

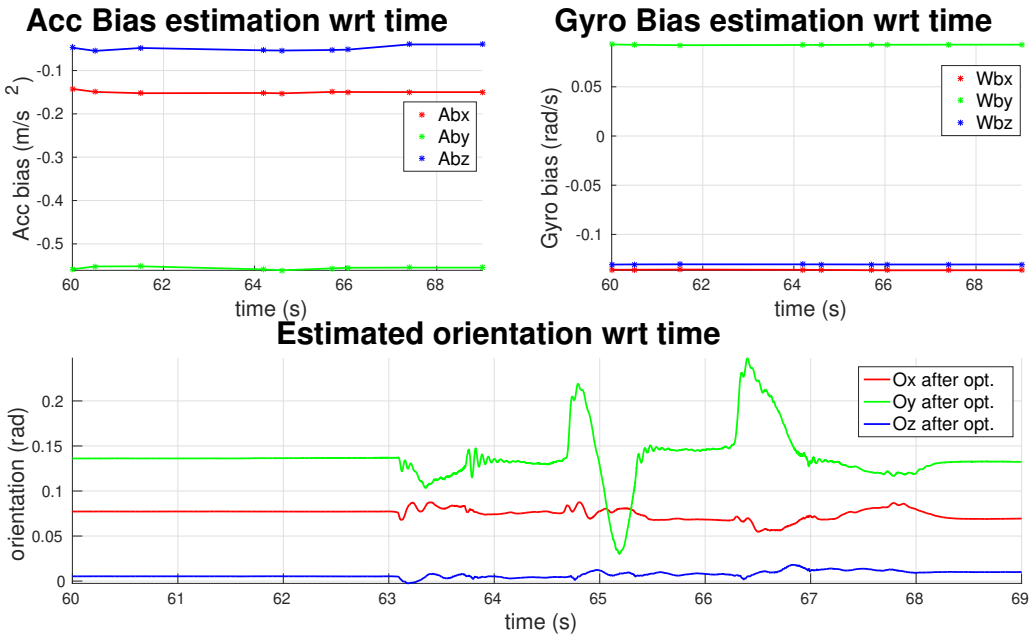


Figure 3.20: Estimation of accelerometer bias $\mathbf{a}_b$ (**Top Left**), gyrometer bias $\boldsymbol{\omega}_b$ (**Top Right**) and orientation (**Bottom**) of the IMU with respect to time while walking 1 metre forward. Each diamond on the graphs represents a key-frame while bars under and over provide a way to visualise the $2\boldsymbol{\sigma}$ bounds around the estimates. The stability of both accelerometer and gyrometer biases may tend to give the impression of a reliable estimate. However, this stability is due also to constraints imposed over the evolution of the biases through time. Furthermore, we note that the Oy orientation is slowly drifting during contact phases despite the fact that the foot is supposed to be still. We can also clearly see the effect of the vibrations on the IMU on this graph. The estimated value for $\mathbf{a}_{by}$ $(-0.55m/s^2)$ is also too big. Hence some doubts remain on the estimation of the biases. We have here one possible explanation for the estimation of the biases given the motion described by the robot and the dynamics applied on the IMU.

Chapter 4

Generalized motion estimation in manifolds, with application to self-calibration

Contents

4.1 Introduction

There has been a remarkable effort in the last years in the robotics community to formulate estimation problems properly. This is motivated by an increasing demand for precision, consistency and stability of the solutions. Indeed, a proper modelling of the states and measurements, the functions relating them, and their uncertainties, is crucial to achieve these goals. This has led to designs involving what has been known as 'manifolds', which are no less than the topologic surfaces of the Lie groups where the state representations evolve. Relying on the Lie group theory we are able to construct a rigorous calculus corpus to handle uncertainties, derivatives and integrals with precision and ease. Typically, these works have focused on the well-known manifolds of rotation $SO(3)$ and rigid motion $SE(3)$. Higher-dimension manifolds such as the IMU states have been treated as composites, though this has not been stated this way explicitly.

When being introduced to Lie groups for the first time, it is important to try to regard them from different points of view. The topological viewpoint, see Fig. 4.1, involves the shape of the manifold and conveys powerful intuitions of its relation to the tangent space and the exponential map. The algebraic viewpoint involves the group operations and their concrete realization, allowing the exploitation of algebraic properties to develop closed-form formulas or to simplify them. The geometrical viewpoint, particularly useful in robotics, associates group elements to the position, velocity, orientation, and/or other modifications of bodies or reference frames. The origin frame may be identified with the group's identity, and any other point on the manifold represents a certain 'local' frame. By resorting to these analogies, many mathematical abstractions of the Lie Theory can be brought closer to intuitive notions in vector spaces, geometry, kinematics and other more classical fields.

In this work, we explore the lessons learned from the development of the IMU preintegration method presented in Section 3.3 and extend some solutions to the usage of any kind of manifold. We purposely take an approach that escapes from the mathematical language of the Group Theory and approaches concepts, terminologies and notations of more widespread use in robotics. The reader interested in more mathematically precise concepts may find this unfortunate, perhaps even inconvenient, but we believe that a good way to familiarise roboticists with such theoretical concepts is through a shared and familiar language. Excellent references one can consult for more detailed explanations are [Chirikjian 2012] and [Eade 2013]. We believe this presentation is novel and interesting. It presents the different Lie Group elements in a way similar to classical linearisation through Jacobians paradigms. This alone we consider a relevant contribution. This work is closely related to Chapter 3 since the bias estimation of the IMU sensor can now be handled as a self-calibration function, even for dynamic parameters. Furthermore, through this generalisation, we present a way to compute Jacobians using the chain rule exposed in 4.2.3.2. It should also be noted that the presented algorithm as well as all the algebra related to deltas remains the same. For new sensors cases, only the

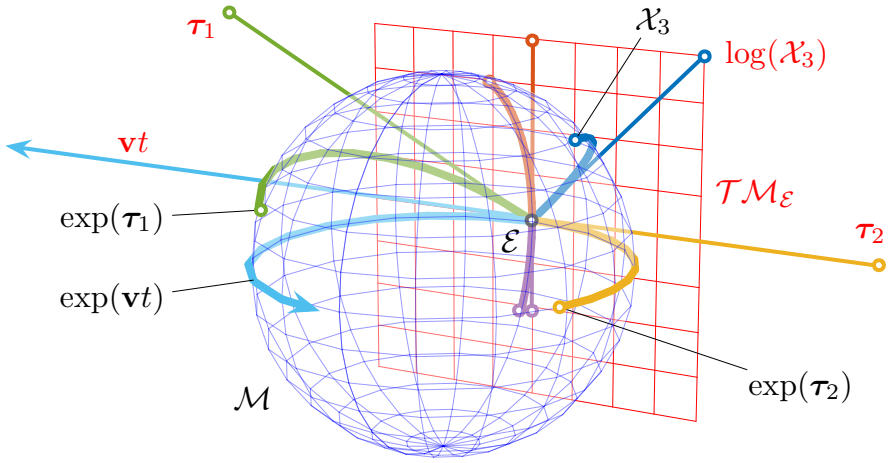


Figure 4.1: Representation of the relation between the Lie group and the Lie algebra. The Lie algebra $\mathcal{T}\mathcal{M}_\mathcal{E}$ (red plane) is the tangent space to the Lie group's manifold $\mathcal{M}$ (here represented as a blue sphere) at the identity $\mathcal{E}$. Through the exponential map, each straight path $\mathbf{v}t$ through the origin on the Lie algebra produces a path $\exp(\mathbf{v}t)$ over the manifold which runs along the respective geodesic. Conversely, each element of the group has an equivalent in the Lie algebra. This relation is so profound that (nearly) all operations in the group, which is curved and nonlinear, have an exact equivalent in the Lie algebra, which is a linear vector space. Though the sphere in $\mathbb{R}^3$ is not a Lie group (we just use it as a representation that can be drawn on paper), that in $\mathbb{R}^4$ is, and describes the group of unit quaternions —see Fig. 4.3.

sensor model equations need to be rewritten. Finally, we introduce the idea of composite manifold for convenience since the IMU state is expressed in such manifold, though this has not been stated this way explicitly.

The main contributor of this work is Joan Solà with the help of Jérémy Deray and I. My personal contribution here lies in the development of the preintegration method explained in Section 3.3 and help provided for the integration of the methods in WOLF C++ library as well as an experimental validation using an IMU.

4.2 Manifold tools for robotics

4.2.1 Definition of manifold and some properties

4.2.1.1 'Manifold' (or 'Lie Group')

In robotics we speak of 'manifold' to mean a Lie group. In mathematics, a group $(\mathcal{G}, \circ)$ is a set, $\mathcal{G}$, with a composition operation, $\circ$, that satisfies the four group

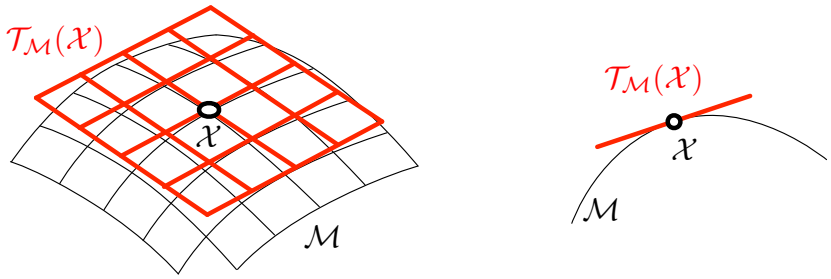


Figure 4.2: A manifold $\mathcal{M}$ and a linear vector space $\mathcal{T}_\mathcal{M} \simeq \mathbb{R}^2$ tangent at the point $\mathcal{X}$, and a convenient side-cut.

axioms:

$$\text{Closure under '}\circ\text{'} : \quad \mathcal{X} \circ \mathcal{Y} \in \mathcal{G} \tag{4.1}$$
$$\text{Associativity} : \quad (\mathcal{X} \circ \mathcal{Y}) \circ \mathcal{Z} = \mathcal{X} \circ (\mathcal{Y} \circ \mathcal{Z}) \tag{4.2}$$
$$\text{Identity } \mathcal{E} : \quad \mathcal{E} \circ \mathcal{X} = \mathcal{X} \circ \mathcal{E} = \mathcal{X} \tag{4.3}$$
$$\text{Inverse } \mathcal{X}^{-1} : \quad \mathcal{X}^{-1} \circ \mathcal{X} = \mathcal{X} \circ \mathcal{X}^{-1} = \mathcal{E} \ . \tag{4.4}$$

for $\mathcal{X}, \mathcal{Y}, \mathcal{Z} \in \mathcal{G}$.

A Lie group is a group that is also a differentiable (*i.e.* smooth) manifold. The reader should be able to visualise the idea of manifold (Fig. 4.2): it is like a bent, smooth (hyper)-surface within a space of higher dimension, like *e.g.* the manifold of unit quaternions, which is a spherical 3-manifold in a 4-dimensional space (Fig. 4.3). In robotics, we say that our state vector evolves in this surface, that is, the manifold describes or is defined by the constraints imposed on the state.

The smoothness of the manifold implies the existence of a unique linear space tangent to any point of it: given $\mathcal{X}$ a point in a manifold $\mathcal{M}$, we name $\mathcal{T}_\mathcal{M}(\mathcal{X})$ the vector space tangent to $\mathcal{M}$ at $\mathcal{X}$ (Fig. 4.2). This is known as the Lie algebra of $\mathcal{M}$. The dimension m of $\mathcal{M}$ is the dimension of $\mathcal{T}_\mathcal{M}$. The elements of $\mathcal{T}_\mathcal{M}$ are non-trivial (skew-symmetric matrices, imaginary numbers, pure quaternions) but the key aspect for us is that they can be expressed as linear combinations of some base elements (called generators). For example, for $[\boldsymbol{\omega}]_\times \in \mathfrak{so}(3)$ we have $[\boldsymbol{\omega}]_\times = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} = \omega_x \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} + \omega_y \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix} + \omega_z \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \simeq \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \in \mathbb{R}^3$. Therefore $\mathcal{T}_\mathcal{M}$ is isomorphic to the Cartesian space $\mathbb{R}^m$ — one writes $\mathcal{T}_\mathcal{M} \simeq \mathbb{R}^m$. A few examples of manifold are detailed in Table 4.1 on page 114.

4.2.1.2 The group actions

Importantly, Lie groups come with the power to transform elements of other sets, producing *e.g.* rotations, translations, scalings, and combinations of them. These are extensively used in robotics, both in 2D and 3D.

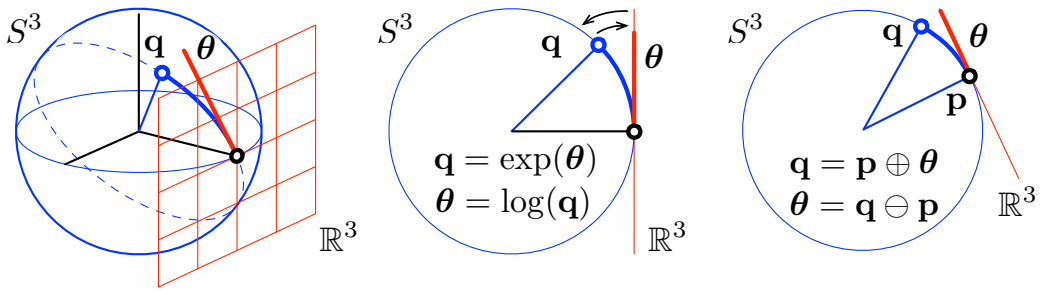


Figure 4.3: The S^3 manifold is a unit 3-sphere in $\mathbb{R}^4$ (blue) where the unit quaternions $\|\mathbf{q}\| = 1$ live. The tangent space $\mathcal{T}_\mathcal{M}$ is isomorphic to the hyperplane $\mathbb{R}^3$ (red grid). The centre and right figures show a side-cut through the plane (dashed) defined by the tangent vector $\boldsymbol{\theta}$ (red segment) and the sphere centre. Mappings exp and log (arrows) map elements of $\mathbb{R}^3$ to/from elements of S^3 (blue arc). The $\oplus, \ominus$ operators express increments between quaternions.

Given a Lie group $\mathcal{M}$ and a set $\mathcal{V}$, we note $\mathcal{X} \cdot v$ the *action* of $\mathcal{X} \in \mathcal{M}$ on $v \in \mathcal{V}$,

$$\cdot \; : \; \mathcal{M} \times \mathcal{V} \to \mathcal{V} \; ; \; (\mathcal{X}, v) \mapsto \mathcal{X} \cdot v \; . \tag{4.5}$$

For $\cdot$ to be a group action, it must satisfy the axioms,

$$\text{Identity :} \qquad \mathcal{E} \cdot v = v \tag{4.6}$$

$$\text{Compatibility :} \qquad (\mathcal{X} \circ \mathcal{Y}) \cdot v = \mathcal{X} \cdot (\mathcal{Y} \cdot v) \; . \tag{4.7}$$

Common examples are the groups of rotation matrices $SO(n)$, the group of unit quaternions, and the groups of rigid motion $SE(n)$. Their respective actions on vectors satisfy

$SO(n)$: rotation matrix	$\mathbf{R} \cdot \mathbf{x} \triangleq \mathbf{R}\mathbf{x}$
$SE(n)$: Euclidean matrix	$\mathbf{H} \cdot \mathbf{x} \triangleq \mathbf{R}\mathbf{x} + \mathbf{t}$
S^1 : unit complex	$\mathbf{z} \cdot \mathbf{x} \triangleq \mathbf{z}\,\mathbf{x}$
S^3 : unit quaternion	$\mathbf{q} \cdot \mathbf{x} \triangleq \mathbf{q}\,\mathbf{x}\,\mathbf{q}^*$

The group composition (4.1) may be viewed as an action of the group on itself, $\circ : \mathcal{M} \times \mathcal{M} \to \mathcal{M}$. Another interesting action is the *adjoint action*, which we will see in 4.2.2.3.

4.2.1.3 The tangent spaces and the Lie algebra

Given $\mathcal{X}(t)$ a point moving on a Lie group's manifold $\mathcal{M}$, its velocity $\dot{\mathcal{X}} = \partial\mathcal{X}/\partial t$ belongs to the space tangent to $\mathcal{M}$ at $\mathcal{X}$ (Fig. 4.2), which we note $\mathcal{T}\mathcal{M}_\mathcal{X}$. The smoothness of the manifold, *i.e.*, the absence of edges or spikes, implies the existence of a unique tangent space at each point. The structure of such tangent spaces is

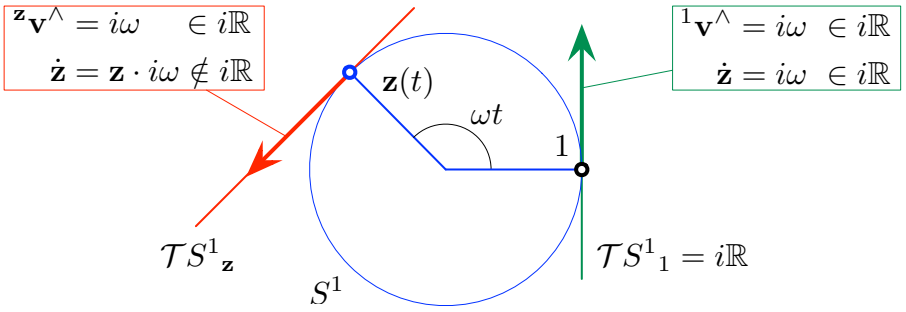


Figure 4.4: Let a point $\mathbf{z} \in S^1$ move at constant rotation rate ω, $\mathbf{z}(t) = \cos\omega t + i\sin\omega t$. Its velocities when passing through 1 and $\mathbf{z}$ are in the respective tangent spaces, $\mathcal{T}S^1{}_1$ and $\mathcal{T}S^1{}_{\mathbf{z}}$. In the case of $\mathcal{T}S^1{}_{\mathbf{z}}$, the velocity is $\dot{\mathbf{z}} = \mathbf{z}\, i\omega = -\omega\sin\omega t + i\omega\cos\omega t$ when expressed in the global coordinates, and $^{\mathbf{z}}\mathbf{v}^{\wedge} = i\omega$ when expressed locally. Their relation is given by $^{\mathbf{z}}\mathbf{v}^{\wedge} = \mathbf{z}^{-1}\dot{\mathbf{z}} = \mathbf{z}^{*}\dot{\mathbf{z}}$. In the case of $\mathcal{T}S^1{}_1$, this relation is the identity $^{1}\mathbf{v}^{\wedge} = \dot{\mathbf{z}} = i\omega$. Clearly, the structure of all tangent spaces is $i\mathbb{R}$, which is the Lie algebra. This is also the structure of $\dot{\mathbf{z}}$ at the identity, and this is why the Lie algebra is defined as the tangent space at the identity.

the same everywhere.

The Lie algebra $\mathfrak{m}$

The tangent space at the identity, $\mathcal{T}\mathcal{M}_{\mathcal{E}}$, is called the *Lie algebra* of $\mathcal{M}$, and noted $\mathfrak{m}$,

$$\text{Lie algebra}: \qquad \mathfrak{m} \triangleq \mathcal{T}\mathcal{M}_{\mathcal{E}} \ . \tag{4.8}$$

Every Lie group has an associated Lie algebra. We relate the Lie group with its Lie algebra through the following facts [Eade] (see Fig. 4.1):

- The Lie algebra $\mathfrak{m}$ is a vector space.[1] As such, its elements can be *identified* with vectors in $\mathbb{R}^m$, whose dimension m is the number of degrees of freedom of $\mathcal{M}$.
- The *exponential map* (also called *retract* in the following), $\exp : \mathfrak{m} \to \mathcal{M}$, exactly converts elements of the Lie algebra into elements of the group. The *log map* (also called *lift* in the following) is the inverse operation.
- Vectors of the tangent space at $\mathcal{X}$ can be transformed to the tangent space at the identity $\mathcal{E}$ through a linear transform. This transform is called the *adjoint*.

Lie algebras can be defined locally to a tangent point $\mathcal{X}$, establishing local coordinates for $\mathcal{T}\mathcal{M}_{\mathcal{X}}$ (Fig. 4.4). We shall denote elements of the Lie algebras

[1] In a Lie algebra, the vector space is endowed with a non-associative product called the Lie bracket. In this work, we will not make use of it.

with a 'hat' decorator, such as $\mathbf{v}^\wedge$ for velocities or $\boldsymbol{\tau}^\wedge = (\mathbf{v}t)^\wedge = \mathbf{v}^\wedge t$ for general elements. A left superscript may also be added to specify the precise tangent space, *e.g.*, ${}^{\mathcal{X}}\mathbf{v}^\wedge \in \mathcal{TM}_\mathcal{X}$ and ${}^{\mathcal{E}}\mathbf{v}^\wedge \in \mathcal{TM}_\mathcal{E}$.

The structure of the Lie algebra can be found by time-differentiating the group constraint (4.3). For multiplicative groups this yields the new constraint $\mathcal{X}^{-1}\dot{\mathcal{X}} + \dot{\mathcal{X}^{-1}}\mathcal{X} = 0$, which applies to the elements tangent at $\mathcal{X}$ (the term $\dot{\mathcal{X}^{-1}}$ is the derivative of the inverse). The elements of the Lie algebra are therefore of the form,[2]

$$\mathbf{v}^\wedge = \mathcal{X}^{-1}\dot{\mathcal{X}} = -\dot{\mathcal{X}^{-1}}\mathcal{X} \ . \tag{4.9}$$

The Cartesian vector space $\mathbb{R}^m$ The elements $\boldsymbol{\tau}^\wedge$ of the Lie algebra have non-trivial structures (skew-symmetric matrices, imaginary numbers, pure quaternions, see Table 4.1) but the key aspect for us is that they can be expressed as linear combinations of some base elements E_i, where E_i are called the *generators* of $\mathfrak{m}$ (they are the derivatives of $\mathcal{X}$ around the origin in the i-th direction). It is then handy to manipulate just the coordinates as vectors in $\mathbb{R}^m$, which we shall note simply $\boldsymbol{\tau}$. We may pass from $\mathfrak{m}$ to $\mathbb{R}^m$ and vice versa through two mutually inverse linear maps or isomorphisms, commonly called *hat* and *vee*,

$$\text{Hat}: \qquad \mathbb{R}^m \to \mathfrak{m}\,; \qquad \boldsymbol{\tau} \mapsto \boldsymbol{\tau}^\wedge = \sum_{i=1}^{m} \tau_i E_i \tag{4.10}$$

$$\text{Vee}: \qquad \mathfrak{m} \to \mathbb{R}^m\,; \qquad \boldsymbol{\tau}^\wedge \mapsto (\boldsymbol{\tau}^\wedge)^\vee = \boldsymbol{\tau} = \sum_{i=1}^{m} \tau_i \mathbf{e}_i \ , \tag{4.11}$$

with $\mathbf{e}_i$ the vectors of the base of $\mathbb{R}^m$ (we have $\mathbf{e}_i^\wedge = E_i$). This means that $\mathfrak{m}$ is isomorphic to the vector space $\mathbb{R}^m$ — one writes $\mathfrak{m} \simeq \mathbb{R}^m$, or $\boldsymbol{\tau}^\wedge \simeq \boldsymbol{\tau}$. Vectors $\boldsymbol{\tau} \in \mathbb{R}^m$ are handier for our purposes than their isomorphics $\boldsymbol{\tau}^\wedge \in \mathfrak{m}$, since they can be stacked in larger state vectors, and more importantly, manipulated with linear algebra using matrix operators. In this work, we enforce this preference of $\mathbb{R}^m$ over $\mathfrak{m}$, to the point that most of the operators and objects that we define (specifically: the adjoint, the Jacobians, the perturbations and their covariances matrices, as we will see soon) are on $\mathbb{R}^m$.

4.2.2 Manifold maps and operations

We have the following maps relating $\mathbb{R}^m$, $\mathcal{T}_\mathcal{M}$ and $\mathcal{M}$,

$$\mathbb{R}^m \underset{\simeq}{\overset{\simeq}{\rightleftharpoons}} \mathcal{T}_\mathcal{M}(\mathcal{E}) \underset{\text{lift}(\cdot)}{\overset{\text{retr}(\cdot)}{\rightleftharpoons}} \mathcal{M} \ , \tag{4.12}$$

where $\simeq$ are linear isomorphisms, and $\text{lift}(\cdot)$ and $\text{retr}(\cdot)$ map the manifold elements to/from the tangent space (see *lift and retract* below). For simplicity, and since we

[2] For additive Lie groups the constraint $\mathcal{X} - \mathcal{X} = 0$ differentiates to $\dot{\mathcal{X}} = \dot{\mathcal{X}}$, that is, no constraint affects the tangent space. This means that the tangent space is the same as the group space.

do not require here the concepts and tools proper of the Lie algebra, we refer to $\mathcal{T}_{\mathcal{M}}$ as the tangent space but express its elements in its isomorphic Cartesian space $\mathbb{R}^m$, so that the maps $\mathcal{T}_{\mathcal{M}} \rightleftharpoons \mathcal{M}$ are implemented algebraically as $\mathbb{R}^m \rightleftharpoons \mathcal{M}$.

4.2.2.1 Lift and retract

Lift and retract map elements $\mathcal{X} \in \mathcal{M}$ with elements $\tau \in \mathcal{T}_{\mathcal{M}}(\mathcal{E})$. *Retract* is the operation of wrapping the tangent space onto the manifold (as when wrapping a ball with a paper), while *lift* is the unwrapping operation. For $\mathcal{X} \in \mathcal{M}$ and $\tau \in \mathcal{T}_{\mathcal{M}}(\mathcal{E})$, we say that

$$\text{lift}: \quad \mathcal{M} \to \mathcal{T}_{\mathcal{M}}(\mathcal{E}) \; ; \quad \mathcal{X} \mapsto \tau = \text{lift}(\mathcal{X}) \tag{4.13}$$

$$\text{retr}: \quad \mathcal{T}_{\mathcal{M}}(\mathcal{E}) \to \mathcal{M} \; ; \quad \tau \mapsto \mathcal{X} = \text{retr}(\tau) \; . \tag{4.14}$$

In the rigid motion groups that we consider (see Table 4.1 on page 114 and Fig. 4.3), lift() and retr() correspond respectively to the log() and exp() maps.

4.2.2.2 Plus and minus

Plus and minus allow us to introduce differences between elements of a (nonlinear) manifold, and express them in its (linear) tangent space. Denoted by $\oplus$ and $\ominus$, they combine one lift/retract operation with one composition. Because of the non-commutativity, they are defined in right- and left- versions (see Fig. 4.3-right for the right version),

$$\text{right-plus:} \quad \tilde{\mathcal{X}} = \mathcal{X} \oplus \tau_r \triangleq \mathcal{X} \circ \text{retr}(\tau_r) \; \in \mathcal{M} \tag{4.15}$$

$$\text{right-minus:} \quad \tau_r = \tilde{\mathcal{X}} \ominus \mathcal{X} \triangleq \text{lift}(\mathcal{X}^{-1} \circ \tilde{\mathcal{X}}) \in \mathcal{T}_{\mathcal{M}}(\mathcal{X}) \tag{4.16}$$

$$\text{left-plus:} \quad \tilde{\mathcal{X}} = \tau_l \oplus \mathcal{X} \triangleq \text{retr}(\tau_l) \circ \mathcal{X} \; \in \mathcal{M} \tag{4.17}$$

$$\text{left-minus:} \quad \tau_l = \tilde{\mathcal{X}} \ominus \mathcal{X} \triangleq \text{lift}(\tilde{\mathcal{X}} \circ \mathcal{X}^{-1}) \in \mathcal{T}_{\mathcal{M}}(\mathcal{E}) \tag{4.18}$$

Because in (4.15) $\text{retr}(\tau_r)$ appears at the right-hand side of the composition, τ_r belongs to the tangent space at $\mathcal{X}$ (see (4.16)): we say that τ_r is expressed in the local frame at $\mathcal{X}$. Conversely, in (4.17) $\text{retr}(\tau_l)$ is on the left and we have $\tau_l \in \mathcal{T}_{\mathcal{M}}(\mathcal{E})$: we say that τ_l is expressed in the global frame. Notice that while left- and right-$\oplus$ are distinguished by the operands order, the $\ominus$ notation in (4.16) and (4.18) is ambiguous. It is typical to express perturbations locally and therefore we use the right- forms of $\oplus$ and $\ominus$ by default.

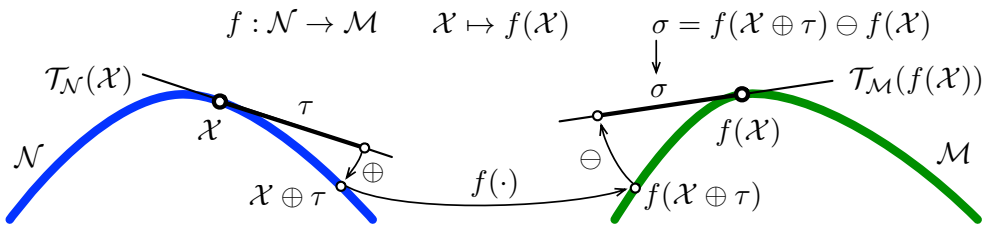


Figure 4.5: Derivative of functions acting between two manifolds $\mathcal{N}$ and $\mathcal{M}$. The perturbation $\tau \in \mathcal{T}_{\mathcal{N}}$ around $\mathcal{X} \in \mathcal{N}$ is propagated to a perturbation $\sigma \in \mathcal{T}_{\mathcal{M}}$ around $f(\mathcal{X}) \in \mathcal{M}$ through the processes of plus, apply $f()$, and minus (arrows). The derivative is simply $\lim_{\tau \to 0} \sigma/\tau$.

4.2.2.3 Adjoint

The adjoint linearly relates vectors from the tangent space at a point $\mathcal{X}$ to vectors of the tangent space at the origin. This may be written with these equivalent forms,

$$\mathcal{X} \oplus \tau = (\mathrm{Adj}_{\mathcal{X}}\, \tau) \oplus \mathcal{X} \tag{4.19}$$

$$\mathcal{X} \circ \mathrm{retr}(\tau) = \mathrm{retr}(\mathrm{Adj}_{\mathcal{X}}\, \tau) \circ \mathcal{X} \tag{4.20}$$

$$\mathrm{Adj}_{\mathcal{X}}\, \tau = \mathrm{lift}(\mathcal{X} \circ \mathrm{retr}(\tau) \circ \mathcal{X}^{-1}) \tag{4.21}$$

Observe that from Eq. (4.15, 4.17, 4.19) we have $\tau_l = \mathrm{Adj}_{\mathcal{X}} \tau_r$.

4.2.2.4 Action

We note with $\mathcal{X} \cdot v$ the *action* of elements $\mathcal{X} \in \mathcal{M}$ on elements v of another set $\mathcal{V}$, having

$$(\mathcal{X} \circ \mathcal{Y}) \cdot v = \mathcal{X} \cdot (\mathcal{Y} \cdot v)\ . \tag{4.22}$$

For example, for vector rotation we have $(\mathbf{QR})\mathbf{v} = \mathbf{Q}(\mathbf{Rv})$.

4.2.3 Derivatives on manifolds

4.2.3.1 Definition of derivatives of functions

Departing from the standard derivative definition, which applies to vector spaces,[3]

$$f : \mathbb{R}^n \to \mathbb{R}^m,\ \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} \triangleq \lim_{\mathbf{h} \to 0} \frac{f(\mathbf{x}+\mathbf{h}) - f(\mathbf{x})}{\mathbf{h}} \in \mathbb{R}^{m \times n}\ ,$$

[3] We use a compact notation in order to handle the n-dimensional infinitesimal $\mathbf{h}$. The notation establishes each column i of the Jacobian matrix of derivatives as $\mathbf{J}_i = \lim_{h \to 0} \frac{f(\mathbf{x}+h\mathbf{e}_i) - f(\mathbf{x})}{h}$, where h is a scalar and $\mathbf{e}_i$ is the i-th vector of the canonical basis of $\mathbb{R}^n$.

we can now use our plus and minus operators to define derivatives of functions acting on manifolds (see Fig. 4.5),[4]

$$f : \mathcal{N} \to \mathcal{M}, \ \frac{\partial f(\mathcal{X})}{\partial \mathcal{X}} \triangleq \lim_{\tau \to 0} \frac{f(\mathcal{X} \oplus \tau) \ominus f(\mathcal{X})}{\tau} \in \mathbb{R}^{m \times n} \ . \tag{4.23}$$

Notice that $\tau \in \mathcal{T}_{\mathcal{N}} \simeq \mathbb{R}^n$, and the numerator of the derivative belongs to $\mathcal{T}_{\mathcal{M}} \simeq \mathbb{R}^m$. This derivative is then a proper Jacobian matrix $\mathbb{R}^{m \times n}$ linearly mapping the spaces $\mathcal{T}_{\mathcal{N}}(\mathcal{X}) \to \mathcal{T}_{\mathcal{M}}(f(\mathcal{X}))$. Remark that whenever the function f passes from one manifold to another, the plus and minus operators must be selected appropriately: plus for the domain $\mathcal{N}$, and minus for the codomain or image $\mathcal{M}$. For small values of τ, the following linear approximation holds,

$$f(\mathcal{X} \oplus \tau) \xrightarrow[\tau \to 0]{} f(\mathcal{X}) \oplus \frac{\partial f(\mathcal{X})}{\partial \mathcal{X}} \tau \in \mathcal{M} \ . \tag{4.24}$$

With (4.23), we can easily compute any derivative from the partial derivative blocks of *inversion*, *composition*, *retraction* and *action* defined hereafter.

The *inverse* derivative block, *i.e.* for $\tau = \mathcal{X}^{-1}$, is defined with (4.23) as

$$\mathbf{J}_{\mathcal{X}}^{\mathcal{X}^{-1}} \triangleq \frac{\partial \mathcal{X}^{-1}}{\partial \mathcal{X}} \in \mathbb{R}^{m \times m} \ . \tag{4.25}$$

while the *composition* derivative block is defined as

$$\mathbf{J}_{\mathcal{X}}^{(\mathcal{X} \circ \mathcal{Y})} \triangleq \frac{\partial \mathcal{X} \circ \mathcal{Y}}{\partial \mathcal{X}} \in \mathbb{R}^{m \times m} \tag{4.26}$$

$$\mathbf{J}_{\mathcal{Y}}^{(\mathcal{X} \circ \mathcal{Y})} \triangleq \frac{\partial \mathcal{X} \circ \mathcal{Y}}{\partial \mathcal{Y}} \in \mathbb{R}^{m \times m} \ . \tag{4.27}$$

We define the right Jacobian of $\mathcal{M}$ as the derivative of retr(), *i.e.*, for $\tau \in \mathcal{T}_{\mathcal{M}}(\mathcal{E}) \simeq \mathbb{R}^m$,

$$\mathbf{J}_r(\tau) \triangleq \frac{\partial \operatorname{retr}(\tau)}{\partial \tau} \in \mathbb{R}^{m \times m} \ , \tag{4.28}$$

which is developed with (4.23) using right-$\ominus$ in $\mathcal{M}$ and '+' in $\mathbb{R}^m$. From (4.23) it is easy to prove that, for small $\delta\tau$, the following approximations hold,

$$\operatorname{retr}(\tau + \delta\tau) \approx \operatorname{retr}(\tau) \operatorname{retr}(\mathbf{J}_r(\tau)\delta\tau) \tag{4.29}$$

$$\operatorname{retr}(\tau) \operatorname{retr}(\delta\tau) \approx \operatorname{retr}(\tau + \mathbf{J}_r^{-1}(\tau)\, \delta\tau) \tag{4.30}$$

$$\operatorname{lift}(\operatorname{retr}(\tau) \operatorname{retr}(\delta\tau)) \approx \tau + \mathbf{J}_r^{-1}(\tau)\, \delta\tau \ . \tag{4.31}$$

Thus, with (4.13), (4.14), (4.23) and approximations above, the Jacobian of

[4]The notation $\frac{\partial f(\mathcal{X})}{\partial \mathcal{X}}$ is chosen in front of other alternatives in order to make the chain rule readable, *i.e.*, $\frac{\partial \mathcal{Z}}{\partial \mathcal{X}} = \frac{\partial \mathcal{Z}}{\partial \mathcal{Y}} \frac{\partial \mathcal{Y}}{\partial \mathcal{X}}$. Also, Note 3 applies.

lift, *i.e.* for $\tau = \text{lift}(\mathcal{X})$, is defined as

$$\begin{aligned}\mathbf{J}_{\mathcal{X}}^{\text{lift}(\mathcal{X})} &\triangleq \lim_{\delta\tau\to 0} \frac{\text{lift}(\mathcal{X}\oplus\delta\tau) - \text{lift}(\mathcal{X})}{\delta\tau} \\ &= \lim_{\delta\tau\to 0} \frac{\text{lift}(\text{retr}(\tau)\,\text{retr}(\delta\tau)) - \tau}{\delta\tau} \\ &= \lim_{\delta\tau\to 0} \frac{\tau + \mathbf{J}_r^{-1}(\tau)\,\delta\tau - \tau}{\delta\tau} \\ &= \mathbf{J}_r^{-1}(\tau) = \mathbf{J}_r^{-1}(\text{lift}(\mathcal{X}))\ . \end{aligned} \tag{4.32}$$

Finally, noting $\mathcal{X}\cdot v$ the action of $\mathcal{X}\in\mathcal{M}$ on $v\in\mathcal{V}$, we define with the jacobian of the group action (4.23) as

$$\mathbf{J}_{\mathcal{X}}^{\mathcal{X}\cdot v} \triangleq \frac{\partial \mathcal{X}\cdot v}{\partial \mathcal{X}}\ , \qquad\qquad \mathbf{J}_{v}^{\mathcal{X}\cdot v} \triangleq \frac{\partial \mathcal{X}\cdot v}{\partial v} \tag{4.33}$$

and all these derivatives can be used with the chain rule detailed hereafter.

4.2.3.2 Using the chain rule to compute other derivatives

The derivatives such as those defined above fulfil the chain rule according to which, the composition of functions allows to compute derivatives by multiplying derivatives in a certain way.

Lemma 4.2.1. *Let's define functions $\mathcal{Y} = f(\mathcal{X})$ and $\mathcal{Z} = g(\mathcal{Y})$ so that $\mathcal{Z} = g(f(\mathcal{X}))$. These functions have the following derivatives respectively: $\mathbf{J}_{\mathcal{X}}^{\mathcal{Y}}$, $\mathbf{J}_{\mathcal{Y}}^{\mathcal{Z}}$ and $\mathbf{J}_{\mathcal{X}}^{\mathcal{Z}}$. Then, according to the* chain rule*, $\mathbf{J}_{\mathcal{X}}^{\mathcal{Z}} = \mathbf{J}_{\mathcal{Y}}^{\mathcal{Z}}\mathbf{J}_{\mathcal{X}}^{\mathcal{Y}}$.*

Proof. The validity of this chain rule can be proven in the case $\mathbf{J}_{\mathcal{X}}^{\mathcal{Z}} = \mathbf{J}_{\mathcal{Y}}^{\mathcal{Z}}\mathbf{J}_{\mathcal{X}}^{\mathcal{Y}}$ using (4.24) twice and the identity $\mathcal{X}\oplus\tau\ominus\mathcal{X} = \tau$, which leads to

$$\begin{aligned}\mathbf{J}_{\mathcal{X}}^{\mathcal{Z}} \triangleq \mathbf{J}_{\mathcal{X}}^{g(f(\mathcal{X}))} &\triangleq \lim_{\tau\to 0} \frac{g(f(\mathcal{X}\oplus\tau))\ominus g(f(\mathcal{X}))}{\tau} \\ [(4.24)\text{ on } f(\mathcal{X})] &= \lim_{\tau\to 0} \frac{g(f(\mathcal{X})\oplus\mathbf{J}_{\mathcal{X}}^{\mathcal{Y}}\tau)\ominus g(f(\mathcal{X}))}{\tau} \\ [(4.24)\text{ on } g(\mathcal{Y})] &= \lim_{\tau\to 0} \frac{g(f(\mathcal{X}))\oplus\mathbf{J}_{\mathcal{Y}}^{\mathcal{Z}}\mathbf{J}_{\mathcal{X}}^{\mathcal{Y}}\tau\ominus g(f(\mathcal{X}))}{\tau} \\ &= \lim_{\tau\to 0} \frac{\text{lift}((g(f(\mathcal{X})))^{-1}\circ g(f(\mathcal{X}))\circ\text{retr}(\mathbf{J}_{\mathcal{Y}}^{\mathcal{Z}}\mathbf{J}_{\mathcal{X}}^{\mathcal{Y}}\tau))}{\tau} \\ &= \lim_{\tau\to 0} \frac{\text{lift}(\text{retr}(\mathbf{J}_{\mathcal{Y}}^{\mathcal{Z}}\mathbf{J}_{\mathcal{X}}^{\mathcal{Y}}\tau))}{\tau} \\ &= \lim_{\tau\to 0} \frac{\mathbf{J}_{\mathcal{Y}}^{\mathcal{Z}}\,\mathbf{J}_{\mathcal{X}}^{\mathcal{Y}}\,\tau}{\tau} = \mathbf{J}_{\mathcal{Y}}^{\mathcal{Z}}\,\mathbf{J}_{\mathcal{X}}^{\mathcal{Y}}\ . \end{aligned} \tag{4.34}$$

□

The chain rule can be seen as a main tool making the expression of Jacobians much clearer and their computation easier. As an illustrative example, the derivatives of *e.g.* $\mathcal{Z} = \mathcal{X} \circ f(\mathcal{Y} \oplus \tau)$ can be given through simple application of the chain rule as,

$$\begin{aligned}
\mathbf{J}^{\mathcal{Z}}_{\mathcal{X}} &= \mathbf{J}^{(\mathcal{X}\circ f(\mathcal{Y}\oplus\tau))}_{\mathcal{X}} \\
\mathbf{J}^{\mathcal{Z}}_{\mathcal{Y}} &= \mathbf{J}^{(\mathcal{X}\circ f(\mathcal{Y}\oplus\tau))}_{f(\mathcal{Y}\oplus\tau)} \mathbf{J}^{f(\mathcal{Y}\oplus\tau)}_{\mathcal{Y}\oplus\tau} \mathbf{J}^{\mathcal{Y}\oplus\tau}_{\mathcal{Y}} \\
\mathbf{J}^{\mathcal{Z}}_{\tau} &= \mathbf{J}^{(\mathcal{X}\circ f(\mathcal{Y}\oplus\tau))}_{f(\mathcal{Y}\oplus\tau)} \mathbf{J}^{f(\mathcal{Y}\oplus\tau)}_{\mathcal{Y}\oplus\tau} \mathbf{J}^{\mathcal{Y}\oplus\tau}_{\tau} \ ,
\end{aligned}$$

where $\mathbf{J}^{f(\mathcal{X})}_{\mathcal{X}}$ must be derived from (4.23), and all other partial derivative blocks are defined in 4.2.3.1. Derivative forms for $\oplus$ and $\ominus$ can be easily derived from the partial derivative blocks and are defined respectively as

$$\mathbf{J}^{\mathcal{X}\oplus\tau}_{\mathcal{X}} = \mathbf{J}^{\mathcal{X}\circ(\mathrm{retr}(\tau))}_{\mathcal{X}} \tag{4.35}$$

$$\mathbf{J}^{\mathcal{X}\oplus\tau}_{\tau} = \mathbf{J}^{\mathcal{X}\circ(\mathrm{retr}(\tau))}_{\mathrm{retr}(\tau)} \mathbf{J}^{\mathrm{retr}(\tau)}_{\tau} = \mathbf{J}^{\mathcal{X}\circ(\mathrm{retr}(\tau))}_{\mathrm{retr}(\tau)} \mathbf{J}_r(\tau) \tag{4.36}$$

for $\oplus$ and

$$\mathbf{J}^{\mathcal{Y}\ominus\mathcal{X}}_{\mathcal{X}} = \mathbf{J}^{\mathrm{lift}(\mathcal{X}^{-1}\circ\mathcal{Y})}_{(\mathcal{X}^{-1}\circ\mathcal{Y})} \mathbf{J}^{(\mathcal{X}^{-1}\circ\mathcal{Y})}_{\mathcal{X}^{-1}} \mathbf{J}^{\mathcal{X}^{-1}}_{\mathcal{X}} \tag{4.37}$$

$$\mathbf{J}^{\mathcal{Y}\ominus\mathcal{X}}_{\mathcal{Y}} = \mathbf{J}^{\mathrm{lift}(\mathcal{X}^{-1}\circ\mathcal{Y})}_{(\mathcal{X}^{-1}\circ\mathcal{Y})} \mathbf{J}^{(\mathcal{X}^{-1}\circ\mathcal{Y})}_{\mathcal{Y}} \ , \tag{4.38}$$

for $\ominus$. Once these forms or 'blocks' are found, all other derivatives follow by the chain rule. Furthermore, the Jacobians are also needed to take into account the effect of uncertainties in manifold through covariance propagation.

4.2.4 Uncertainty in manifolds, covariance propagation

Perturbations around a point $\bar{\mathcal{X}} \in \mathcal{N}$ are defined in the linear space $\mathcal{T}_{\mathcal{N}}(\bar{\mathcal{X}})$, that is, for a local perturbation τ,

$$\mathcal{X} = \bar{\mathcal{X}} \oplus \tau \ , \qquad\qquad \tau = \mathcal{X} \ominus \bar{\mathcal{X}} \ \in \mathcal{T}_{\mathcal{N}}(\bar{\mathcal{X}}) \ . \tag{4.39}$$

Covariances matrices can be properly defined on this tangent space at $\bar{\mathcal{X}}$ through the standard expectation operator $\mathbb{E}[\cdot]$,

$$cov(\mathcal{X}) \triangleq \mathbb{E}[(\mathcal{X} \ominus \bar{\mathcal{X}})(\mathcal{X} \ominus \bar{\mathcal{X}})^\top] \ \in \mathbb{R}^{n\times n} \ . \tag{4.40}$$

Notice that since the dimension n of $\mathcal{T}_{\mathcal{N}}$ matches the degrees of freedom of $\mathcal{N}$, these covariances are well defined.[5] Covariance propagation through a function $f : \mathcal{N} \to \mathcal{M}; \mathcal{X} \mapsto f(\mathcal{X})$ just requires the manifold Jacobian matrices (4.23) and

[5] A plain definition $cov(\mathcal{X}) \triangleq \mathbb{E}[(\mathcal{X} - \bar{\mathcal{X}})(\mathcal{X} - \bar{\mathcal{X}})^\top]$ is always ill-defined if $\mathrm{size}(\mathcal{X}) > \dim(\mathcal{N})$, which is the case for most non-trivial manifolds.

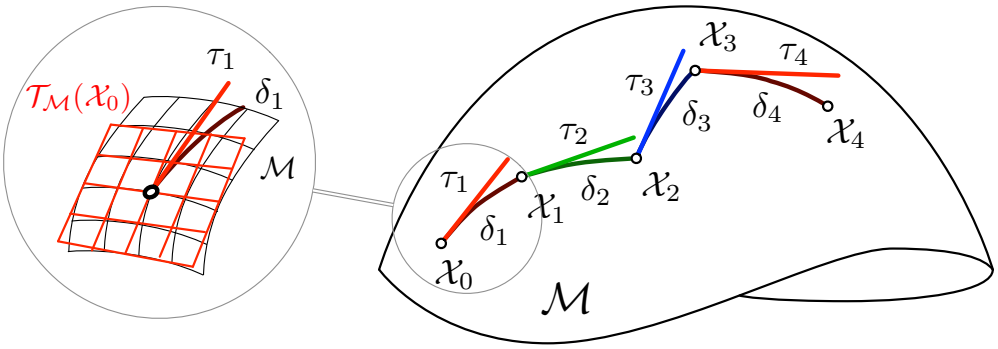


Figure 4.6: Motion integration on a manifold. Each motion data produces a step $\tau_k \in \mathcal{T}_{\mathcal{M}}(\mathcal{X}_{k-1})$, which is retracted to a local motion increment or 'delta' $\delta_k \in \mathcal{M}$, and then composed with $\mathcal{X}_{k-1}$ to yield $\mathcal{X}_k = \mathcal{X}_{k-1} \circ \text{retr}(\tau_k) = \mathcal{X}_{k-1} \circ \delta_k \in \mathcal{M}$.

regular linear covariance propagation,

$$cov(f(\mathcal{X})) \approx \frac{\partial f}{\partial \mathcal{X}}\, cov(\mathcal{X})\, \frac{\partial f}{\partial \mathcal{X}}^{\top} \in \mathbb{R}^{m\times m} . \tag{4.41}$$

Perturbations can also be expressed in the global reference, that is, in the tangent space at the origin $\mathcal{T}_{\mathcal{N}}(\mathcal{E})$. Using left- plus and minus,

$$\mathcal{X} = \tau \oplus \bar{\mathcal{X}} , \qquad \tau = \mathcal{X} \ominus \bar{\mathcal{X}} \in \mathcal{T}_{\mathcal{N}}(\mathcal{E}) . \tag{4.42}$$

This allows global specification of covariance matrices using left-minus in (4.40). For example, a 3D orientation that is known up to rotations in the horizontal plane can be associated to a covariance $\mathbf{Q} = \text{Diag}(\sigma_\phi^2, \sigma_\theta^2, \infty)$ only if $\mathbf{Q}$ is specified in the global reference.

4.2.5 Discrete integration on manifolds

We can construct a sound integration of a sequence of (small) steps $\tau_1, \cdots, \tau_k \in \mathcal{T}_{\mathcal{M}}$ onto the manifold (Fig. 4.6), *i.e.*, $\mathcal{X}_k \triangleq \mathcal{X}_0 \oplus \tau_1 \oplus \cdots \oplus \tau_k$, which we write in recursive form,

$$\mathcal{X}_k = \mathcal{X}_{k-1} \oplus \tau_k = \mathcal{X}_{k-1} \circ \text{retr}(\tau_k) . \tag{4.43}$$

Notice here that the time-derivative of a point moving on a manifold lies precisely on the tangent space, so this space is sometimes called the velocity space. Thus, if we are to integrate discrete velocity data, say v_k, then $\tau_k = v_k \delta t$ is a proper step in the tangent space. Common examples are the integration of 3D angular rates $\boldsymbol{\omega}$ into the rotation matrix, $\mathbf{R}_k = \mathbf{R}_{k-1} \exp([\boldsymbol{\omega}_k \delta t]_\times)$, or into the quaternion, $\mathbf{q}_k = \mathbf{q}_{k-1} \otimes \exp(\boldsymbol{\omega}_k \delta t/2)$.

4.2.6 Composite manifolds

One can consider more complex manifolds as composites. Take as example the space of translations and rotations. We have for this the well-known $SE(n)$ manifold of rigid motions, but we can also construct the $\mathbb{R}^n \times SO(n)$ manifold, which we call a *composite* of the (trivial) $\mathbb{R}^n$ and the $SO(n)$ manifolds. These are very similar, but not equivalent.

Composite manifolds may employ operators retract, lift, plus and minus that act either on the whole manifold, or on each of its components. Take $\mathbb{R}^3 \times SO(3)$ as an example, with manifold elements $\mathcal{S} = (\mathbf{p}, \mathbf{R})$ and tangents $\tau = (\delta\mathbf{p}, \delta\boldsymbol{\theta})$. For motion integration, it is beneficiary to use the whole manifold, as the different parts get conveniently coupled,

$$\mathcal{S} \oplus \tau \triangleq \begin{bmatrix} \mathbf{p} + \mathbf{R}\delta\mathbf{p} \\ \mathbf{R}\exp(\delta\boldsymbol{\theta}) \end{bmatrix} \tag{4.44}$$

$$\mathcal{S}_2 \ominus \mathcal{S}_1 \triangleq \begin{bmatrix} \mathbf{R}_1^\top(\mathbf{p}_2 - \mathbf{p}_1) \\ \log(\mathbf{R}_1^\top \mathbf{R}_2) \end{bmatrix} \tag{4.45}$$

On the contrary, for differentiation, error evaluation and uncertainty management, it is usual to take each manifold block separately, *i.e.*,

$$\mathcal{S} \boxplus \tau \triangleq \begin{bmatrix} \mathbf{p} \oplus \delta\mathbf{p} \\ \mathbf{R} \oplus \delta\boldsymbol{\theta} \end{bmatrix} = \begin{bmatrix} \mathbf{p} + \delta\mathbf{p} \\ \mathbf{R}\exp(\delta\boldsymbol{\theta}) \end{bmatrix} \tag{4.46}$$

$$\mathcal{S}_2 \boxminus \mathcal{S}_1 \triangleq \begin{bmatrix} \mathbf{p}_2 \ominus \mathbf{p}_1 \\ \mathbf{R}_2 \ominus \mathbf{R}_1 \end{bmatrix} = \begin{bmatrix} \mathbf{p}_2 - \mathbf{p}_1 \\ \log(\mathbf{R}_1^\top \mathbf{R}_2) \end{bmatrix} , \tag{4.47}$$

where we mark the block-operation of plus and minus with $\boxplus$, $\boxminus$. It is convenient to realise that these operators are indeed simpler than the regular plus and minus (4.44, 4.45).

The consequence of these considerations is that new derivatives can be defined,

$$\frac{\partial f(\mathcal{X})}{\partial \mathcal{X}} \triangleq \lim_{\tau \to 0} \frac{f(\mathcal{X} \oplus \tau) \boxminus f(\mathcal{X})}{\tau} . \tag{4.48}$$

With this derivative, Jacobian matrices of functions acting on composite manifolds can be determined in a per-block basis, which yields simpler expressions,

$$\frac{\partial f}{\partial \mathcal{X}} = \begin{bmatrix} \frac{\partial f_1}{\partial \mathcal{X}_1} & \cdots & \frac{\partial f_1}{\partial \mathcal{X}_M} \\ \vdots & & \\ \frac{\partial f_1}{\partial \mathcal{X}_N} & \cdots & \frac{\partial f_N}{\partial \mathcal{X}_M} \end{bmatrix} \tag{4.49}$$

where M, N are respectively the number of composite blocks of $\mathcal{X} \in \mathcal{M}$ and $f(\mathcal{X}) \in \mathcal{N}$. Although never explicitly stated, this is the approach taken in many IMU works on manifolds, for example [Forster 2017]. We present this concept of composite manifold for convenience, since this makes it possible to unify the methods presented

into a generalised methodology.

4.3 Graph-based motion estimation on manifolds

In graph-based motion estimation (as in *e.g.* odometry and SLAM, Fig. 4.7) we often have very different rates of motion sensor data and keyframe (KF) creation. Therefore, hundreds of motion measurements need to be integrated to generate a motion factor linking two consecutive KFs. It is convenient to integrate a relative motion or 'delta' Δ_{ij} with respect to the last KF with state $\mathbf{x}_i$ (Fig. 4.8), so as to have, given some composition operator $\boxplus$,

$$\mathbf{x}_j = \mathbf{x}_i \boxplus \Delta_{ij} \ , \tag{4.50}$$

and use it to generate a relative motion factor linking $\mathbf{x}_i$ to the next KF, *i.e.*, through an error $\mathbf{e}_{ij}$ so that

$$\mathbf{e}_{ij} = \Delta_{ij} \ominus (\mathbf{x}_j \boxminus \mathbf{x}_i) \ . \tag{4.51}$$

The motion increment Δ_{ij} is found by integrating small motion contributions at the sensor data rate, *i.e.*, at each arrival of the motion data $\mathbf{y}_k$, we do

$$\Delta_{ik} = \Delta_{ij} \boxplus \delta_k \qquad \text{with} \quad \delta_k = f(\mathbf{y}_k) \ . \tag{4.52}$$

This appears rather straightforward. But only in the most basic cases these operations are trivial. In many cases, the motion data comes from uncalibrated sensors, either in its extrinsic or intrinsic parameters. Some of these parameters may additionally drift with time, thus requiring continuous tracking. Moreover, in some complicated cases (*e.g.* IMU) the integrated deltas strongly depend on the initial conditions of orientation and velocity. So the general situation becomes,

$$\Delta_{ik}(\mathbf{x}_i, \mathbf{c}_i) = \Delta_{ij}(\mathbf{x}_i, \mathbf{c}_i) \boxplus f(\mathbf{y}_k, \mathbf{x}_i, \mathbf{c}_i) \ , \tag{4.53}$$

where $\mathbf{c}_i$ are (possibly time-varying) calibration parameters to be jointly estimated alongside the states $\mathbf{x}_i$.

In all these cases, the changes in the estimates of $\mathbf{x}_i$ and $\mathbf{c}_i$ (inherent to the iterative nature of the optimisation) affect the whole motion integral. To avoid the need of re-integrating all data at each iteration of the optimisation algorithm, the delta pre-integration theory was developed for the IMU sensor [Lupton 2009, Forster 2017]. On the one hand, this theory defines the deltas independently of the initial conditions of orientation and velocity, thus depending *only* on sensor data $\mathbf{y}$ and calibration parameters $\mathbf{c}$. These are pre-integrated once,

$$\overline{\Delta}_{ik}(\overline{\mathbf{c}}_i) = \overline{\Delta}_{ij}(\overline{\mathbf{c}}_i) \boxplus f(\mathbf{y}_k, \overline{\mathbf{c}}_i) \ . \tag{4.54}$$

On the other hand, the effect of the changes in the estimates of the calibration

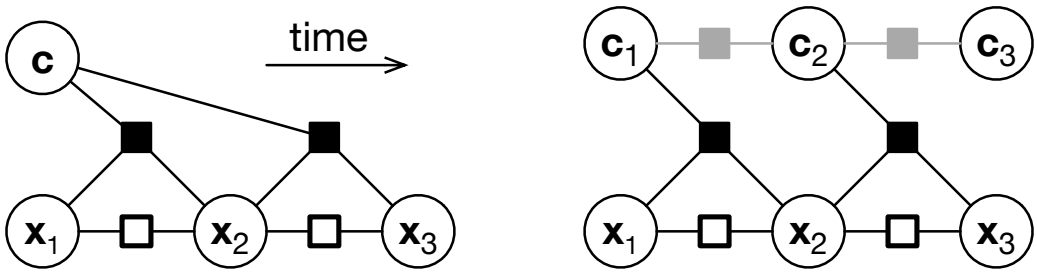


Figure 4.7: Graph-based motion estimation with self-calibration. Pre-integrated factors (black squares) link consecutive KF states and the calibration parameters (circles). Observability is provided by other sensors observing motion at a lower rate (wireframe squares), such as vision or LIDAR. *Left:* fixed calibration parameters. *Right:* time-varying calibration parameters. Additional factors limiting parameter drifts are shown in grey. See Fig. 3.2 for a real implementation.

parameters is linearised so that the deltas can be corrected a posteriori, *i.e.*, when computing the residual, using pre-integrated Jacobians and (4.24),

$$\Delta_{ij} = \overline{\Delta}_{ij}(\overline{\mathbf{c}}_i) \oplus \mathbf{J}_{\mathbf{c}_i}^{\Delta_{ij}}(\mathbf{c}_i - \overline{\mathbf{c}}_i) \ . \tag{4.55}$$

We generalise the pre-integration theory to any kind of motion sensor. We provide several contributions:

1. a segmentation of the computation pipeline (from measurements, to body magnitudes, to the current delta, and to the integrated delta);

2. a physical interpretation of the delta magnitudes, defined in a manifold;

3. a simple yet rigorous algebraic approach, valid for any type of manifold, which takes profit of the pipeline segmentation and the chain rule to compute the otherwise cumbersome [Forster 2017] Jacobians. Important complements are provided in Subsection 4.2.2 and Section C.1 for the sake of background and completeness.

4.3.1 Notation

We note the deltas manifold as $\mathcal{D}$, with dimension d. We define the integrated delta $\Delta_{ij} \in \mathcal{D}$ and the current delta $\delta_k \in \mathcal{D}$, with time indices defined as in Fig. 4.8. We define the additive and subtractive compositions on $\mathcal{D}$ as

$$\Delta_{ij} \boxplus \Delta_{jk} \triangleq \Delta_{ij} \circ \Delta_{jk} \tag{4.56}$$

$$\Delta_{ik} \boxminus \Delta_{ij} \triangleq \Delta_{ij}^{-1} \circ \Delta_{ik} \ . \tag{4.57}$$

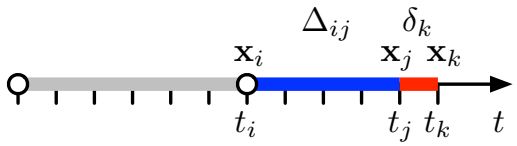


Figure 4.8: The pre-integrated delta $\Delta_{ij} \in \mathcal{D}$ contains all motion from the last KF at time i, up to time j, so that $\mathbf{x}_j = \mathbf{x}_i \boxplus \Delta_{ij}$. The current delta $\delta_k \in \mathcal{D}$ contains the motion from times j to k, computed from the last motion measurement at time k, so that $\mathbf{x}_k = \mathbf{x}_j \boxplus \delta_k = \mathbf{x}_i \boxplus \Delta_{ij} \boxplus \delta_k = \mathbf{x}_i \boxplus \Delta_{ik}$.

Similarly, we use the same symbols to define compositions between states $\mathbf{x}_i$ and deltas Δ_{ij} so that

$$\mathbf{x}_j = \mathbf{x}_i \boxplus \Delta_{ij} \tag{4.58}$$

$$\Delta_{ij} = \mathbf{x}_j \boxminus \mathbf{x}_i \ , \tag{4.59}$$

and notice that only in cases where the states are in the same manifold $\mathcal{D}$ these compositions are the same as above. We mark pre-integrated magnitudes with a bar, *e.g.* $\overline{\Delta}_{ij}, \overline{\mathbf{c}}$. We name $\mathbf{Q}_{ij}$, $\mathbf{Q}_k$ and $\mathbf{Q}_n$ the covariances of respectively the pre-integrated delta, the current delta, and the measurement noise. For a function $\mathcal{Y} = f(\mathcal{X})$, we note the Jacobian $\mathbf{J}^{\mathcal{Y}}_{\mathcal{X}} \triangleq \frac{\partial \mathcal{Y}}{\partial \mathcal{X}} = \frac{\partial f(\mathcal{X})}{\partial \mathcal{X}}$ in the manifold sense, *i.e.*, (4.23). The chain rule becomes $\mathbf{J}^{\mathcal{Z}}_{\mathcal{X}} = \mathbf{J}^{\mathcal{Z}}_{\mathcal{Y}} \cdot \mathbf{J}^{\mathcal{Y}}_{\mathcal{X}}$.

4.3.2 Pipeline

The pipeline has two distinct phases: pre-integration to create a factor, and factor evaluation to optimise the system. During the pre-integration phase we proceed as follows:

1. Compensate measured data with available calibration parameters. This typically results in an element of $\mathcal{T}_{\mathcal{D}}$.
2. Compute the current motion delta $\delta_k \in \mathcal{D}$.
3. Pre-integrate $\overline{\Delta}_{ik} \in \mathcal{D}$, $\mathbf{Q}_{ik}$, and the Jacobian $\mathbf{J}^{\Delta_{ik}}_{\mathbf{c}}$. Upon creation of a new KF, a motion factor is built with the result of these integrals. Then, at each solver iteration, we evaluate each of the factors as follows.
4. Compute the residual with the pre-integrated covariance.

These are detailed in Algs. 5-7, and described hereafter.

4.3.2.1 Delta pre-integration (Algorithm 5)

At the arrival of each motion measurement $\mathbf{y}_k$, we use the currently available calibration parameters $\overline{\mathbf{c}}$ to pre-compensate the measurements. Depending on the nature of the measurements, we may need to integrate the motion over the sampling time

Algorithm 5: Pre-integration on the deltas manifold $\mathcal{D}$

Input: i; $\overline{\mathbf{c}}$; $\mathcal{Y} = \{\mathbf{y}_k\}$; $\mathbf{Q}_n$

$\overline{\Delta}_{ii} = \mathcal{E}_{\mathcal{D}}$, $\mathbf{Q}_{ii} = 0$, $\mathbf{J}_{\mathbf{c}}^{\Delta_{ii}} = 0$, $j = i$

while *Not KF* **do**

$k = j + 1$

Compensate motion data

$\mathbf{b}_k = f_c(\overline{\mathbf{c}}, \mathbf{y}_k, \mathbf{n}, \delta t) \in \mathcal{T}_{\mathcal{D}} \;\rightarrow\; \mathbf{J}_{\mathbf{c}}^{\mathbf{b}_k}, \mathbf{J}_{\mathbf{n}}^{\mathbf{b}_k}$

Compute current delta and covariance

$\delta_k = \mathrm{retr}(\mathbf{b}_k) \in \mathcal{D} \quad \rightarrow\; \mathbf{J}_{\mathbf{b}_k}^{\delta_k}$

$\mathbf{Q}_k = \mathbf{J}_{\mathbf{n}}^{\delta_k}\, \mathbf{Q}_n\, \mathbf{J}_{\mathbf{n}}^{\delta_k}{}^{\top}, \quad$ with $\;\mathbf{J}_{\mathbf{n}}^{\delta_k} = \mathbf{J}_{\mathbf{b}_k}^{\delta_k}\, \mathbf{J}_{\mathbf{n}}^{\mathbf{b}_k}$

Pre-integrate delta, covariance, and Jacobian

$\overline{\Delta}_{ik} = \overline{\Delta}_{ij} \boxplus \delta_k \in \mathcal{D} \quad \rightarrow\; \mathbf{J}_{\Delta_{ij}}^{\Delta_{ik}}, \mathbf{J}_{\delta_k}^{\Delta_{ik}}$

$\mathbf{Q}_{ik} = \mathbf{J}_{\Delta_{ij}}^{\Delta_{ik}}\, \mathbf{Q}_{ij}\, \mathbf{J}_{\Delta_{ij}}^{\Delta_{ik}}{}^{\top} + \mathbf{J}_{\delta_k}^{\Delta_{ik}}\, \mathbf{Q}_k\, \mathbf{J}_{\delta_k}^{\Delta_{ik}}{}^{\top}$

$\mathbf{J}_{\mathbf{c}}^{\Delta_{ik}} = \mathbf{J}_{\Delta_{ij}}^{\Delta_{ik}}\, \mathbf{J}_{\mathbf{c}}^{\Delta_{ij}} + \mathbf{J}_{\delta_k}^{\Delta_{ik}}\, \mathbf{J}_{\mathbf{b}_k}^{\delta_k}\, \mathbf{J}_{\mathbf{c}}^{\mathbf{b}_k}$

$j = k$

Compute upper square root of information matrix $\mathbf{\Omega}_{ij}^{\top/2} = \mathbf{L}^{\top}$,
with $\mathbf{Q}_{ij}^{-1} = \mathbf{L}\mathbf{L}^{\top}$ via Cholesky

Output: $\overline{\Delta}_{ij}, \mathbf{\Omega}_{ij}^{\top/2}, \mathbf{J}_{\mathbf{c}}^{\Delta_{ij}}$

δt. We also compute the Jacobians with respect to the calibration parameters $\mathbf{c}$ and the measurements' noise $\mathbf{n} \sim \mathcal{N}\{0, \mathbf{Q}_n\}$,

$$\mathbf{b}_k = f_b(\overline{\mathbf{c}}, \mathbf{y}_k, \mathbf{n}, \delta t) \tag{4.60a}$$

$$\mathbf{J}_{\mathbf{c}}^{\mathbf{b}_k} = \frac{\partial \mathbf{b}_k}{\partial \mathbf{c}} \qquad \mathbf{J}_{\mathbf{n}}^{\mathbf{b}_k} = \frac{\partial \mathbf{b}_k}{\partial \mathbf{n}} \tag{4.60b}$$

It is common, though not mandatory, that $\mathbf{b}_k \in \mathcal{T}_{\mathcal{D}}$. The operations in $f_b()$ are completely dependent on each particular system and cannot be generalised.

We then wrap the compensated step $\mathbf{b}_k$ onto a delta on the manifold (Fig. 4.6),

$$\delta_k = f_\delta(\mathbf{b}_k) \qquad \in \mathcal{D} \tag{4.61a}$$

$$\mathbf{J}_{\mathbf{b}_k}^{\delta_k} = \frac{\partial \delta_k}{\partial \mathbf{b}_k} \qquad \in \mathbb{R}^{d \times d} \tag{4.61b}$$

Notice that if $\mathbf{b}_k \in \mathcal{T}_{\mathcal{D}}$, then $\delta_k = f_\delta(\mathbf{b}_k) = \mathrm{retr}(\mathbf{b}_k)$. We then compute the delta covariances matrix,

$$\mathbf{Q}_k = \mathbf{J}_{\mathbf{n}}^{\delta_k} \mathbf{Q}_n \mathbf{J}_{\mathbf{n}}^{\delta_k}{}^{\top} \qquad \in \mathbb{R}^{d \times d} \tag{4.62}$$

Algorithm 6: Residual evaluation, Lupton and Forster

Input: $\overline{\Delta}_{ij}$, $\boldsymbol{\Omega}_{ij}^{\top/2}$, $\mathbf{J}_{\mathbf{c}}^{\Delta_{ij}}$, $\overline{\mathbf{c}}$, $\mathbf{x}_i$, $\mathbf{x}_j$, $\mathbf{c}$

Predict delta from KFs $\mathbf{x}_i$ and $\mathbf{x}_j$

$$\widehat{\Delta}_{ij} = \mathbf{x}_j \boxminus \mathbf{x}_i \in \mathcal{D} \qquad \rightarrow \mathbf{J}_{\mathbf{x}_i}^{\widehat{\Delta}_{ij}}, \mathbf{J}_{\mathbf{x}_j}^{\widehat{\Delta}_{ij}}$$

Correct pre-integrated delta with new calibration $\mathbf{c}$

$$\Delta_{ij} = \overline{\Delta}_{ij} \oplus \mathbf{J}_{\mathbf{c}}^{\Delta_{ij}}(\mathbf{c} - \overline{\mathbf{c}}) \in \mathcal{D} \qquad \rightarrow \mathbf{J}_{\mathbf{c}}^{\Delta_{ij}}$$

Compute delta error and residual

$$\mathbf{e}_{ij} = \widehat{\Delta}_{ij} \ominus \Delta_{ij} \in \mathcal{T}_{\mathcal{D}}(\Delta_{ij}) \simeq \mathbb{R}^d \qquad \rightarrow \mathbf{J}_{\widehat{\Delta}_{ij}}^{\mathbf{e}_{ij}}, \mathbf{J}_{\Delta_{ij}}^{\mathbf{e}_{ij}}$$

$$\mathbf{r}_{ij} = \boldsymbol{\Omega}_{ij}^{\top/2}\mathbf{e}_{ij} \in \mathbb{R}^d$$

Compute residual Jacobians

$$\mathbf{J}_{\mathbf{x}_i}^{\mathbf{e}_{ij}} = \mathbf{J}_{\widehat{\Delta}_{ij}}^{\mathbf{e}_{ij}}\mathbf{J}_{\mathbf{x}_i}^{\widehat{\Delta}_{ij}},\ \mathbf{J}_{\mathbf{x}_j}^{\mathbf{e}_{ij}} = \mathbf{J}_{\widehat{\Delta}_{ij}}^{\mathbf{e}_{ij}}\mathbf{J}_{\mathbf{x}_j}^{\widehat{\Delta}_{ij}},\ \mathbf{J}_{\mathbf{c}}^{\mathbf{e}_{ij}} = \mathbf{J}_{\Delta_{ij}}^{\mathbf{e}_{ij}}\mathbf{J}_{\mathbf{c}}^{\Delta_{ij}}$$

$$\mathbf{J}_{*}^{\mathbf{r}_{ij}} = \boldsymbol{\Omega}_{ij}^{\top/2}\mathbf{J}_{*}^{\mathbf{e}_{ij}}$$

Output: $\mathbf{r}_{ij}$, $\mathbf{J}_{\mathbf{x}_i}^{\mathbf{r}_{ij}}$, $\mathbf{J}_{\mathbf{x}_j}^{\mathbf{r}_{ij}}$, $\mathbf{J}_{\mathbf{c}}^{\mathbf{r}_{ij}}$

where $\mathbf{J}_{\mathbf{n}}^{\delta_k} = \mathbf{J}_{\mathbf{b}_k}^{\delta_k}\mathbf{J}_{\mathbf{n}}^{\mathbf{b}_k}$ is obtained via the chain rule.

We finally incorporate the current delta onto the pre-integrated delta (Fig. 4.6), and compute the Jacobians,

$$\overline{\Delta}_{ik} = \overline{\Delta}_{ij} \boxplus \delta_k \qquad \in \mathcal{D} \tag{4.63a}$$

$$\mathbf{J}_{\Delta_{ij}}^{\Delta_{ik}} = \frac{\partial \overline{\Delta}_{ik}}{\partial \Delta_{ij}}\ ,\ \mathbf{J}_{\delta_k}^{\Delta_{ik}} = \frac{\partial \overline{\Delta}_{ik}}{\partial \delta_k} \qquad \in \mathbb{R}^{d\times d}\ , \tag{4.63b}$$

we integrate the covariances matrix on $\mathcal{T}_{\mathcal{D}}(\overline{\Delta}_{ik})$,

$$\mathbf{Q}_{ik} = \mathbf{J}_{\Delta_{ij}}^{\Delta_{ik}}\mathbf{Q}_{ij}{\mathbf{J}_{\Delta_{ij}}^{\Delta_{ik}}}^{\top} + \mathbf{J}_{\delta_k}^{\Delta_{ik}}\mathbf{Q}_k{\mathbf{J}_{\delta_k}^{\Delta_{ik}}}^{\top} \qquad \in \mathbb{R}^{d\times d}\ , \tag{4.64}$$

and integrate the Jacobian with respect to the calibration parameters, using the chain rule,

$$\mathbf{J}_{\mathbf{c}}^{\Delta_{ik}} = \mathbf{J}_{\Delta_{ij}}^{\Delta_{ik}}\mathbf{J}_{\mathbf{c}}^{\Delta_{ij}} + \mathbf{J}_{\delta_k}^{\Delta_{ik}}\mathbf{J}_{\mathbf{b}_k}^{\delta_k}\mathbf{J}_{\mathbf{c}}^{\mathbf{b}_k} \qquad \in \mathbb{R}^{d\times c}\ . \tag{4.65}$$

Since $\mathbf{c} \in \mathbb{R}^c$, this Jacobian maps $\mathbb{R}^c \rightarrow \mathcal{T}_{\mathcal{D}}(\overline{\Delta}_{ik})$.

4.3.2.2 Residual evaluation (Algs. 6 and 7)

At keyframe creation, we need to produce a factor storing $\overline{\Delta}_{ij}$, $\mathbf{Q}_{ij}$, $\mathbf{J}_{\mathbf{c}}^{\Delta_{ij}}$, and $\overline{\mathbf{c}}$. These are used to evaluate the residual, which depends on the states $\mathbf{x}_i, \mathbf{x}_j$ of the two KFs and the state $\mathbf{c}$ of the calibration parameters.

We propose two alternative algorithms. The first one (Algorithm 6) corresponds to the original algorithm [Lupton 2009] for the IMU, adapted to the $SO(3)$ manifold

Algorithm 7: Residual evaluation, our proposal

Input: $\overline{\Delta}_{ij}$, $\boldsymbol{\Omega}_{ij}^{\top/2}$, $\mathbf{J}_{\mathbf{c}}^{\Delta_{ij}}$, $\overline{\mathbf{c}}$, $\mathbf{x}_i$, $\mathbf{x}_j$, $\mathbf{c}$

Predict delta from KFs $\mathbf{x}_i$ and $\mathbf{x}_j$

$$\widehat{\Delta}_{ij} = \mathbf{x}_j \boxminus \mathbf{x}_i \in \mathcal{D} \qquad \rightarrow \mathbf{J}_{\mathbf{x}_i}^{\widehat{\Delta}_{ij}}, \mathbf{J}_{\mathbf{x}_j}^{\widehat{\Delta}_{ij}}$$

Predict delta correction step

$$\hat{\mathbf{s}} \triangleq \widehat{\Delta}_{ij} \ominus \overline{\Delta}_{ij} \in \mathcal{T}_{\mathcal{D}}(\overline{\Delta}_{ij}) \qquad \rightarrow \mathbf{J}_{\widehat{\Delta}_{ij}}^{\hat{\mathbf{s}}}, \ \mathbf{J}_{\overline{\Delta}_{ij}}^{\hat{\mathbf{s}}}$$

Compute delta correction step from $\mathbf{c}$ and $\overline{\mathbf{c}}$

$$\mathbf{s} \approx \mathbf{J}_{\mathbf{c}}^{\Delta_{ij}}(\mathbf{c} - \overline{\mathbf{c}}) \in \mathcal{T}_{\mathcal{D}}(\overline{\Delta}_{ij}) \qquad \rightarrow \mathbf{J}_{\mathbf{c}}^{\mathbf{s}} = \mathbf{J}_{\mathbf{c}}^{\Delta_{ij}}$$

Compute correction step error and residual

$$\mathbf{e}_{ij} = \hat{\mathbf{s}} - \mathbf{s} \in \mathcal{T}_{\mathcal{D}}(\overline{\Delta}_{ij}) \simeq \mathbb{R}^d$$

$$\mathbf{r}_{ij} = \boldsymbol{\Omega}_{ij}^{\top/2}\, \mathbf{e}_{ij} \in \mathbb{R}^d$$

Compute residual Jacobians

$$\mathbf{J}_{\mathbf{x}_i}^{\mathbf{e}_{ij}} = \mathbf{J}_{\widehat{\Delta}_{ij}}^{\hat{\mathbf{s}}} \mathbf{J}_{\mathbf{x}_i}^{\widehat{\Delta}_{ij}}, \ \mathbf{J}_{\mathbf{x}_j}^{\mathbf{e}_{ij}} = \mathbf{J}_{\widehat{\Delta}_{ij}}^{\hat{\mathbf{s}}} \mathbf{J}_{\mathbf{x}_j}^{\widehat{\Delta}_{ij}}, \ \mathbf{J}_{\mathbf{c}}^{\mathbf{e}_{ij}} = -\mathbf{J}_{\mathbf{c}}^{\mathbf{s}}$$

$$\mathbf{J}_{*}^{\mathbf{r}_{ij}} = \boldsymbol{\Omega}_{ij}^{\top/2}\, \mathbf{J}_{*}^{\mathbf{e}_{ij}}$$

Output: $\mathbf{r}_{ij}$, $\mathbf{J}_{\mathbf{x}_i}^{\mathbf{r}_{ij}}$, $\mathbf{J}_{\mathbf{x}_j}^{\mathbf{r}_{ij}}$, $\mathbf{J}_{\mathbf{c}}^{\mathbf{r}_{ij}}$

in [Forster 2017], and generalised here for any kind of motion integration. We start by predicting a motion delta given KF's $\mathbf{x}_i$ and $\mathbf{x}_j$,

$$\widehat{\Delta}_{ij} = \mathbf{x}_j \boxminus \mathbf{x}_i \qquad \in \mathcal{D} \tag{4.66a}$$

Second, we correct the pre-integrated delta with the new calibration values $\mathbf{c}$,

$$\Delta_{ij} = \overline{\Delta}_{ij} \oplus \mathbf{J}_{\mathbf{c}}^{\Delta_{ij}}(\mathbf{c} - \overline{\mathbf{c}}) \qquad \in \mathcal{D} \tag{4.66b}$$

Third, we compute the tangent delta error,

$$\mathbf{e}_{ij} = \widehat{\Delta}_{ij} \ominus \Delta_{ij} \qquad \in \mathcal{T}_{\mathcal{D}}(\Delta_{ij}) \tag{4.66c}$$

And finally we weight it to produce the residual

$$\mathbf{r}_{ij} = \boldsymbol{\Omega}_{ij}^{\top/2}\, \mathbf{e}_{ij} \qquad \in \mathcal{T}_{\mathcal{D}}(\Delta_{ij}) \simeq \mathbb{R}^d \tag{4.66d}$$

where $\boldsymbol{\Omega}_{ij} = \mathbf{Q}_{ij}^{-1}$ is the information matrix of the pre-integrated delta, defined in $\mathcal{T}_{\mathcal{D}}(\overline{\Delta}_{ij})$, and $\boldsymbol{\Omega}_{ij}^{\top/2}$ is its upper triangular square root, obtained by Cholesky factorisation.

This first algorithm has the (small) inconvenient that the pre-computed covariance $\mathbf{Q}_{ij}$ belongs to $\mathcal{T}_{\mathcal{D}}(\overline{\Delta}_{ij})$, while the computed error $\mathbf{e}_{ij}$ belongs to $\mathcal{T}_{\mathcal{D}}(\Delta_{ij})$, so the weighting in (4.66d) is not proper. This can be corrected by propagating $\mathbf{Q}_{ij}$ to $\mathcal{T}_{\mathcal{D}}(\Delta_{ij})$ using the Jacobian of (4.66b), but we prefer instead to evaluate the error

directly on $\mathcal{T}_{\mathcal{D}}(\overline{\Delta}_{ij})$. In our proposed algorithm (Algorithm 7), we predict a correction step $\hat{\mathbf{s}}$ given the predicted $\widehat{\Delta}_{ij}$ and pre-integrated $\overline{\Delta}_{ij}$ deltas, and compare it against the actual correction step $\mathbf{s}$ computed through the pre-integrated Jacobian,

$$\widehat{\Delta}_{ij} = \mathbf{x}_j \boxminus \mathbf{x}_i \qquad \in \mathcal{D} \tag{4.67a}$$

$$\hat{\mathbf{s}} \triangleq \widehat{\Delta}_{ij} \ominus \overline{\Delta}_{ij} \qquad \in \mathcal{T}_{\mathcal{D}}(\overline{\Delta}_{ij}) \tag{4.67b}$$

$$\mathbf{s} = \mathbf{J}_{\mathbf{c}}^{\Delta_{ij}}(\mathbf{c} - \overline{\mathbf{c}}) \qquad \in \mathcal{T}_{\mathcal{D}}(\overline{\Delta}_{ij}) \tag{4.67c}$$

$$\mathbf{e}_{ij} = \hat{\mathbf{s}} - \mathbf{s} \qquad \in \mathcal{T}_{\mathcal{D}}(\overline{\Delta}_{ij}) \tag{4.67d}$$

$$\mathbf{r}_{ij} = \boldsymbol{\Omega}_{ij}^{\top/2}\, \mathbf{e}_{ij} \qquad \in \mathcal{T}_{\mathcal{D}}(\overline{\Delta}_{ij}) \simeq \mathbb{R}^d \ . \tag{4.67e}$$

Notice that all these results belong to the same tangent space. Moreover, this method is cheaper in that it has one fewer $\oplus$ (retract and compose) operation.

4.3.3 State reconstruction

Optimised states can be retrieved at any time t_j, even if this t_j does not correspond to any KF,

$$\mathbf{x}_j = \mathbf{x}_i \boxplus (\overline{\Delta}_{ij} \oplus \mathbf{J}_{\mathbf{c}}^{\Delta_{ij}}(\mathbf{c}_i - \overline{\mathbf{c}}_i)) \ , \tag{4.68}$$

where $\mathbf{x}_i$ is the last optimised KF before t_j, and $\mathbf{c}_i$ is the optimised calibration vector associated to $\mathbf{x}_i$. To do so we need to store all the history of $\overline{\Delta}_{ij}$ and $\mathbf{J}_{\mathbf{c}}^{\Delta_{ij}}$. The same is true for $\mathbf{Q}_{ij}$ if we aim also at reconstructing the covariance trajectory.

In the next section, we present an example of application of the concepts presented until here. Thus the generalised preintegration method is used in $SE(2)$ in the case of the self-calibration using a differential drive model.

4.4 The generalised preintegration method for the IMU

The preintegration method initially presented in Section 3.3 uses the pipeline exposed in Algs. 5 and 6. Here we show to what extent the initial preintegration method adheres to the general one.

4.4.1 The Delta preintegration is specific to the sensor

The first step of the preintegration method is to compute the current body magnitude with pre-compensation of the measurements given the current calibration parameters $\overline{\mathbf{c}}$ following (3.11). In the IMU case, $\overline{\mathbf{c}}$ are the bias parameters. As it was stated sooner, this operation and the computation of the related derivatives (Eq. (3.10)) cannot be generalised because of dependencies on the system. Finally, the computed body magnitude is used to define the current δ_k as detailed in Eq. (3.16) as well as the jacobians of the current δ. These are therefore the only

equations that need to be coded for each new sensor providing a different set of measurements.

All remaining operations, that are the composition in the manifold, as well as the correction and residuals evaluation steps have a common generalised formulation. In the specific case of the IMU, these operations are processed in a composite manifold (see 4.2.6) of dimension 10 due to the dimension of δ itself. The corresponding composite manifold is defined by $\mathbb{R}^3 \times S(3) \times \mathbb{R}^3$ with manifold elements $\mathcal{S} = (\mathbf{p}, \mathbf{q}, \mathbf{v})$ and tangents $\tau = (\delta\mathbf{p}, \delta\mathbf{q}, \delta\mathbf{v})$. This delta preintegration step, represented by (4.54) in the general case, is manifold dependent and thus needs to be defined for each different preintegration method. We recall hereafter the incremental delta preintegration for IMUs.

$$\begin{aligned}
\Delta\mathbf{p}_{ik} &= \Delta\mathbf{p}_{ij} + \Delta\mathbf{v}_{ij}\delta t + \frac{1}{2}\Delta\mathbf{q}_{ij} \odot \delta\mathbf{p}_{jk} \\
\Delta\mathbf{v}_{ik} &= \Delta\mathbf{v}_{ij} + \Delta\mathbf{q}_{ij} \odot \delta\mathbf{v}_{jk} \\
\Delta\mathbf{q}_{ik} &= \Delta\mathbf{q}_{ij} \otimes \delta\mathbf{q}_{jk}
\end{aligned}$$

Similarly, the delta composition as well as the state composition or differentiation equations implying the use of operators $\boxplus$ or $\boxminus$ are specific to each sensor.

4.4.2 Using the derivatives

4.4.2.1 Derivatives computation

The preintegration method implies the uses of several derivatives (Jacobians of the body magnitude, of the current delta and Jacobians of the delta composition). These derivatives are related to sensor specific operations that need to be redefined for each new sensor. As a consequence, these specific derivatives need to be coded again to use the preintegration method with a new type of sensor. The chain rule makes it possible to generalise the computation of other Jacobians and Covariances using the derivatives that are specific to each type of sensor. The first consequence is that the computation of covariances is made easier and does not need to be re-coded. Furthermore, we can notice that the jacobian with respect to the calibration parameters *i.e.*, the bias parameters in the IMU case, can be computed using the chain rule (see (4.65)) and thus their computation can be generalised. In the IMU case developed in Section 3.3, this Jacobian is the pre-integrated bias Jacobian $\mathbf{J}^{\Delta}_{\mathbf{b}_b}$ which computation we recall hereafter.

$$\mathbf{J}^{\Delta_{ik}}_{\mathbf{b}_b} = \mathbf{J}^{\Delta_{ik}}_{\Delta_{ij}}\mathbf{J}^{\Delta_{ij}}_{\mathbf{b}_b} - \mathbf{J}^{\Delta_{ik}}_{\delta_{jk}}\mathbf{J}^{\delta_{jk}}_{\mathbf{b}_b} \ .$$

This formula matches exactly (4.65) with $\mathbf{J}^{\delta_{jk}}_{\mathbf{b}_b} = \mathbf{J}^{\delta_{jk}}_{\mathbf{b}}\mathbf{J}^{\mathbf{b}}_{\mathbf{b}_b}$ and $\mathbf{J}^{\mathbf{b}_k}_{\mathbf{c}} = \mathbf{J}^{\mathbf{b}_k}_{\mathbf{b}_b} = -\mathbf{I}_6$ (see (3.18)).

4.4.2.2 A generalised sensor self-calibration

Thus another consequence of the generalisation of the preintegration method to any manifold is that the self-calibration method itself is generalised. Indeed, the delta correction with new bias given as $\Delta = \overline{\Delta} + \mathbf{J}^{\Delta}_{\mathbf{b}_b}(\mathbf{b}_b - \overline{\mathbf{b}_b})$ is a formulation specific to the IMU and yet matching the more general preintegrated delta correction with new calibration values $\Delta_{ij} = \overline{\Delta}_{ij} \oplus \mathbf{J}^{\Delta_{ij}}_{\mathbf{c}}(\mathbf{c} - \overline{\mathbf{c}})$. This correction does not depend on the sensor and is realised in the manifold formed by the preintegrated values. Thus it does not need to be coded again for each new type of sensor.

Only the residual evaluation of the former IMU preintegration method is different to the proposal as formulated in Algorithm 7. Indeed, in the first case, the residual is defined using the computed error $\mathbf{e}_{ij}$ defined in $\mathcal{T}_{\mathcal{D}}(\Delta_{ij})$ instead of $\mathcal{T}_{\mathcal{D}}(\overline{\Delta}_{ij})$ where the pre-computed covariance $\mathbf{Q}_{ij}$ (and by extension the information matrix $\mathbf{\Omega}_{ij}$) used to weight the error to produce the residual is defined. The new expression of the residual can be deduced from the methodology proposed in Algorithm 7 as

$$\begin{aligned}\mathbf{r}_{ij} = \mathbf{\Omega}_{ij}^{\top/2}\,\mathbf{e}_{ij} &= \mathbf{\Omega}_{ij}^{\top/2}\left((\widehat{\Delta}_{ij} \diamondminus \overline{\Delta}_{ij}) - \mathbf{J}^{\Delta_{ij}}_{\mathbf{c}}(\mathbf{c} - \overline{\mathbf{c}})\right) \\ &= \mathbf{\Omega}_{ij}^{\top/2}\left(\begin{bmatrix} \widehat{\Delta\mathbf{p}}_{ij} - \overline{\Delta\mathbf{p}}_{ij} \\ \widehat{\Delta\mathbf{v}}_{ij} - \overline{\Delta\mathbf{v}}_{ij} \\ \mathrm{Log}(\overline{\Delta\mathbf{q}}^{*}_{ij} \otimes \widehat{\Delta\mathbf{q}}_{ij}) \end{bmatrix} - \mathbf{J}^{\Delta_{ij}}_{\mathbf{c}}(\mathbf{c} - \overline{\mathbf{c}})\right).\end{aligned}$$

This generalised preintegration method is implemented in WOLF, a library presented with more details in Chapter 5.

4.5 Self-calibration example in SE(2) for a differential drive

The self-calibration application exposed hereafter is part of the work developed by Jérémy Deray with the help of Joan Solà.

4.5.1 Overview

The differential drive model consists of two actuated wheels on a single axis, one on each side of the robot base, with its origin frame located at the center of the axis. The robot is parametrised by its wheels radii (r_l, r_r) and the length d of the axis. To each of these parameters is associated a correction factor such that $\mathbf{c} = [c_l\ c_r\ c_d]^\top$ is the intrinsic parameters calibration vector. The motion is usually measured by means of wheel encoders reporting incremental wheel angles $\mathbf{y} = [\delta\psi_l, \delta\psi_r] + \mathbf{n}$ every time step δt, where $\mathbf{n}$ is additive Gaussian noise.

4.5.2 State and delta definitions

We define the states of position and orientation angle $\mathbf{x} = (\mathbf{p}, \theta)$, which act as a compact representation of $SE(2)$. Consequently, state increments or deltas Δ_{ij} and δ_k are also in $SE(2)$.

4.5.3 Incremental delta pre-integration

For convenience, we define the body magnitudes $\mathbf{b} = (\delta l, \delta\theta) = f_b(\mathbf{y}, \mathbf{c}, \mathbf{n})$ as

$$\begin{aligned} \delta l &= \tfrac{1}{2}(r_r c_r \delta\psi_r + r_l c_l \delta\psi_l) \\ \delta\theta &= \tfrac{1}{D}(r_r c_r \delta\psi_r - r_l c_l \delta\psi_l) \ . \end{aligned} \tag{4.69}$$

with $D = d\, c_d$. Its components, respectively, highlight the common (along track length), and differential (turn) components of the wheels reported motions. From the equation above, one can find the jacobians of the motion components. Their expressions are given hereafter.

$$\mathbf{J}^{\mathbf{b}}_{\mathbf{y}} = \mathbf{J}^{\mathbf{b}}_{\mathbf{n}} = \begin{bmatrix} \frac{1}{2} r_l c_l & \frac{1}{2} r_r c_r \\ -r_l c_l & r_r c_r \end{bmatrix} \qquad \in \mathbb{R}^{2\times 2} \tag{4.70a}$$

$$\mathbf{J}^{\mathbf{b}}_{\mathbf{c}} = \begin{bmatrix} \frac{1}{2}\psi_l r_l & \frac{1}{2}\psi_r r_r & 0 \\ -\psi_l r_l & \psi_r r_r & -\frac{\delta\theta}{c_d} \end{bmatrix} \qquad \in \mathbb{R}^{2\times 3} \tag{4.70b}$$

Assuming constant control inputs over the sampling time period $[t_j, t_k]$, the robot moves along an arc of circle of radius

$$R = \frac{\delta l}{\delta\theta} \ . \tag{4.71}$$

The motion increment δ_k can be expressed in $SE(2)$ as

$$\begin{aligned} \delta x &= R \sin(\delta\theta) \\ \delta y &= R(1 - \cos(\delta\theta)) \\ \delta\theta &= \delta\theta \ , \end{aligned} \tag{4.72}$$

which is exactly $\delta = \exp(\tau) \in SE(2)$ with $\tau = [\delta l, 0, \delta\theta] \in \mathfrak{se}(2)$, *i.e.*, assuming no wheel slippage. In case the robot follows a straight trajectory, we have $\delta\theta \to 0$, and $R \to +\infty$. This case is handled by means of the approximation

$$\begin{aligned} \delta x &= \delta l \cos(\delta\theta/2) \\ \delta y &= \delta l \sin(\delta\theta/2) \\ \delta\theta &= \delta\theta \ . \end{aligned} \tag{4.73}$$

As it is done for the IMU case in $SO(3)$ (Section 3.3.4.2), we can define the

Jacobian of δ from (4.72) as

$$\mathbf{J}_{\mathbf{b}}^{\delta_k} = \begin{bmatrix} \frac{\sin(\delta\theta)}{\delta\theta} & R(\cos(\delta\theta) - \frac{\sin(\delta\theta)}{\delta\theta}) \\ \frac{1-cos(\delta\theta)}{\delta\theta} & R(\sin(\delta\theta) - \frac{1-\cos(\delta\theta)}{\delta\theta}) \\ 0 & 1 \end{bmatrix} \in \mathbb{R}^{3\times 2} \tag{4.74}$$

Similarly, an equivalent form can be found from (4.73),

$$\mathbf{J}_{\mathbf{b}}^{\delta_k} = \begin{bmatrix} \cos(\delta\theta/2) & -\frac{1}{2}\delta l \sin(\delta\theta/2) \\ \sin(\delta\theta/2) & \frac{1}{2}\delta l \cos(\delta\theta/2) \\ 0 & 1 \end{bmatrix} \in \mathbb{R}^{3\times 2} \tag{4.75}$$

The delta integration is simply the composition in $SE(2)$,

$$\begin{aligned} \Delta\mathbf{p}_{ik} &= \Delta\mathbf{p}_{ij} + \Delta\mathbf{R}_{ij}\delta\mathbf{p}_k \\ \Delta\theta_{ik} &= \Delta\theta_{ij} + \delta\theta_k \end{aligned} \tag{4.76}$$

where $\Delta\mathbf{R}_{ij} = \exp(\Delta\theta_{ij})$ is the rotation matrix delta corresponding to the rotation angle of $\Delta\theta_{ij}$, see (C.17) in Appendix C.2, and $\delta\mathbf{p}_k$ is the translation vector delta corresponding to δ_k. The jacobians of the delta composition are given by

$$\mathbf{J}_{\Delta_{ij}}^{\Delta_{ik}} = \begin{bmatrix} \mathbf{I} & \Delta\mathbf{R}_{ij}\,[1]_\times\,\delta\mathbf{p}_k \\ \mathbf{0} & 1 \end{bmatrix} \in \mathbb{R}^{3\times 3} \tag{4.77a}$$

$$\mathbf{J}_{\delta_k}^{\Delta_{ik}} = \begin{bmatrix} \Delta\mathbf{R}_{ij} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \in \mathbb{R}^{3\times 3} \tag{4.77b}$$

with $[1]_\times = \left[\begin{smallmatrix} 0 & -1 \\ 1 & 0 \end{smallmatrix}\right]$, see Appendix C.2.3.

4.5.4 Integration of the delta covariance and the Jacobian

We follow the general procedure, obtaining $\mathbf{Q}_k$ from (4.62). Here, the diagonal measurement covariance $\mathbf{Q}_n$ is defined by,

$$\begin{aligned} \mathbf{Q}_n &= \begin{bmatrix} \sigma_{\psi_l}^2 + \alpha^2 & 0 \\ 0 & \sigma_{\psi_r}^2 + \alpha^2 \end{bmatrix}, \\ \sigma_{\psi_l}^2 &= k_l\delta\psi_l, \quad \sigma_{\psi_r}^2 = k_r\delta\psi_r, \quad \alpha = \tfrac{1}{2}(\mu_l + \mu_r)\ , \end{aligned} \tag{4.78}$$

where k_r and k_l are constant parameters, and α acts as an offset equal to half the wheels encoders resolution μ_l and μ_r [Siegwart 2011]. The pre-integrated delta covariance $\mathbf{Q}_{ik}$ is then integrated normally with (4.64), starting at $\mathbf{Q}_{ii} = \mathbf{0}_{3\times 3}$.

For the Jacobian we use (4.65), starting at $\mathbf{J}_{\mathbf{c}}^{\Delta_{ii}} = \mathbf{0}_{3\times 3}$.

4.5.5 Residual

Following Algorithm 6 with $\diamondminus$ for errors, we have

$$\widehat{\Delta}_{ij} = \begin{bmatrix} \mathbf{R}_i^\top(\mathbf{p}_j - \mathbf{p}_i) \\ \theta_j - \theta_i \end{bmatrix} \tag{4.79}$$

$$\Delta_{ij} = \Delta_{ij} \oplus \mathbf{J}_{\mathbf{c}}^{\Delta_{ij}}(\mathbf{c}_i - \overline{\mathbf{c}}_i) \tag{4.80}$$

$$\mathbf{r}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{c}_i) = \mathbf{\Omega}_{ij}^{\top/2} \begin{bmatrix} \widehat{\Delta\mathbf{p}}_{ij} - \Delta\mathbf{p}_{ij} \\ \widehat{\Delta\theta}_{ij} - \Delta\theta_{ij} \end{bmatrix} \in \mathbb{R}^3 \ , \tag{4.81}$$

where the angle differences must be brought to $(-\pi, \pi\,]$. The information matrix is given by $\mathbf{\Omega}_{ij} = \mathbf{Q}_{\Delta ij}^{-1}$. As explained sooner, this residual evaluation is not proper due to the terms belonging to different spaces. Indeed the information matrix belongs to $\mathcal{T}_{\mathcal{D}}(\overline{\Delta}_{ij})$ while the error computed as above is defined in $\mathcal{T}_{\mathcal{D}}(\Delta_{ij})$. For a more rigorous algebraic approach, we could also define the residuals following Algorithm 7 as

$$\mathbf{r}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{c}_i) = \mathbf{\Omega}_{ij}^{\top/2}\, \mathbf{e}_{ij} = \mathbf{\Omega}_{ij}^{\top/2} \left(\begin{bmatrix} \widehat{\Delta\mathbf{p}}_{ij} - \overline{\Delta\mathbf{p}}_{ij} \\ \widehat{\Delta\theta}_{ij} - \overline{\Delta}\theta_{ij} \end{bmatrix} - \mathbf{J}_{\mathbf{c}}^{\Delta_{ij}}(\mathbf{c} - \overline{\mathbf{c}}) \right) \in \mathbb{R}^3 \tag{4.82}$$

where all the terms are defined in $\mathcal{T}_{\mathcal{D}}(\overline{\Delta}_{ij}) \simeq \mathbb{R}^3$.

4.6 Conclusion

In this work, we have proceeded to a selection of materials that avoids abstract mathematical concepts as much as possible. This we hope helps focusing Lie theory to make its tools easier to understand and to use. Third, we have promoted the usage of handy operators, such as the lift() and retr() maps, and the plus and minus operators $\oplus$, $\ominus$, $\boxplus$, $\boxminus$. They allow us to work on the Cartesian representation of the tangent spaces, producing formulas for derivatives and covariance handling that greatly resemble their counterparts in standard vector spaces. Using these tools, we have generalised the correction step of the preintegration theory to its use on any manifold and by extension with any kind of motion sensor. Through the presentation of this generalised method, we provide a computation pipeline for implementation in real systems and that benefits to the computation of Jacobians on which we have made a special emphasis by using the chain rule. Furthermore, the delta magnitude defined in a manifold have a physical understanding. We believe these contributions will help to a better understanding of the preintegration theory, make its use easier for further applications and allow an efficient implementation of the method on real systems. Among possible applications, we can cite the sensor self-calibration as a side-effect of the generalisation of the theory through a general procedure to correct preintegrated deltas taking into account new calibration parameters. We illustrated this application with a differential drive and an IMU. Works are being conducted to provide further sensor calibration examples as well

as experimental results. Again, the interested reader is invited to read [Solà 2018] for further explanations as well as examples for a better understanding.

Table 4.1: Typical manifolds used in 2D and 3D motion, including the trivial $\mathbb{R}^n$ (some details omitted)

Manifold $\mathcal{M}, \circ$		size	dim	$\mathcal{M}$	$\mathcal{T}_\mathcal{M}$	Cartesian	retract	lift
n-D vector space	$\mathbb{R}^n, +$	n	n	$\mathbf{v} \in \mathbb{R}^n$	$\mathbf{v} \in \mathbb{R}^n$	$\mathbf{v} \in \mathbb{R}^n$	identity	identity
circle group	$S^1, \cdot$	2	1	$\mathbf{z} \in \mathbb{C}$	$i\theta \in \mathrm{Im}(\mathbb{C})$	$\theta \in \mathbb{R}$	$\mathbf{z} = \exp(i\theta)$	log()
Rotation group	$SO(2), \cdot$	4	1	$\mathbf{R}$	$[\theta]_\times \in \mathfrak{so}(2)$	$\theta \in \mathbb{R}$	$\mathbf{R} = \exp([\theta]_\times)$	log()
Rigid motion group	$SE(2), \cdot$	9	3	$\begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} [\theta]_\times & \mathbf{v} \\ 0 & 1 \end{bmatrix} \in \mathfrak{se}(2)$	$(\mathbf{v}, \theta) \in \mathbb{R}^3$	exp()	log()
3-sphere group	$S^3, \otimes$	4	3	$\mathbf{q} \in \mathbb{H}$	$\mathbf{u}\theta/2 \in \mathrm{Im}(\mathbb{H})$	$\boldsymbol{\theta} \in \mathbb{R}^3$	$\mathbf{q} = \exp(\mathbf{u}\theta/2)$	log()
Rotation group	$SO(3), \cdot$	9	3	$\mathbf{R}$	$[\boldsymbol{\theta}]_\times \in \mathfrak{so}(3)$	$\boldsymbol{\theta} \in \mathbb{R}^3$	$\mathbf{R} = \exp([\boldsymbol{\theta}]_\times)$	log()
Rigid motion group	$SE(3), \cdot$	16	6	$\begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} [\boldsymbol{\theta}]_\times & \mathbf{v} \\ 0 & 1 \end{bmatrix} \in \mathfrak{se}(3)$	$(\mathbf{v}, \boldsymbol{\theta}) \in \mathbb{R}^6$	exp()	log()
IMU deltas group	$\mathcal{D}, \boxplus$	11	10	$\Delta \in \mathcal{D}$	not specified	$\delta(\mathbf{p}, \boldsymbol{\theta}, \mathbf{v}, t) \in \mathbb{R}^{10}$	retr()	lift()

Chapter 5

WOLF as part of the Loco3D project

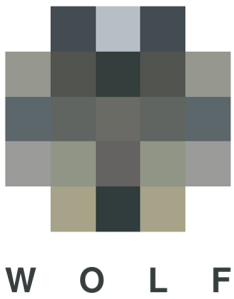


Figure 5.1: WOLF logo

Contents

One of the current research focus in GEPETTO team is to face the multi-contact locomotion problem of legged robots in complex environments through the Loco3D project [Carpentier 2017]. While this is clearly a planning and control problem in order to generate precise and complex motions for the robot to navigate and interact with its environment, it is required to close the control loop through onboard perception. SLAM methods are perfect candidates to enhance the control with perception techniques. In this context, the library WOLF (acronym for Windowed

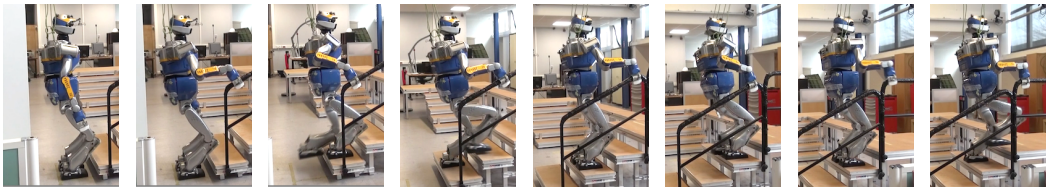


Figure 5.2: Snapshots of the climbing up 15-cm high steps motion with the HRP-2 using the stair railing.

LOcalization Frames) is very useful to provide the perception side which is needed for a complete closed-loop control scheme that does not require any external infrastructure (for robot localisation) or previous knowledge of the environment (thus the need of online mapping).

5.1 The Loco3D Project

5.1.1 Introduction

Planning, adapting and executing multi-contact locomotion movements on legged robots in complex environments remains an open problem. The Loco3D project aims at introducing a complete pipeline to address this issue in the context of humanoid robots inside industrial environments. This pipeline relies on a multi-stage approach in order to simplify the process flow and to exploit at best state-of-the-art techniques both in terms of contact planning, whole-body control and perception. Loco3D stands for Locomotion in 3D, in contrast to the classic locomotion on quasi-flat terrains, where the motion of the centre of mass of the robot is mostly limited to a 2D plane. The main challenges lie in the choice of the different modules composing this pipeline as well as their mutual interactions [Carpentier 2017].

5.1.2 Motivations

Many challenges remain open when it comes to applications involving the use of legged robots. Efforts are already made on the perception of the environment and on the interaction with it [Victorino 2003]. Amongst the challenges, we can also cite the multi-contact locomotion of legged robots in complex environments. Indeed, hardware limitations due to the design and constraints applied to robots may be overcome through the use of the environment so that the robot can finally achieve a task. For example, the use of a handrail by HRP-2 to climb stairs (Fig. 5.2) allows it to achieve the task while reducing the power consumption by 25% [Kudruss 2015]. An example of motion requiring the use of contacts with the environment is given in [Carpentier 2016]. The presented method allows to generate feasible contact sequences for the robot to stand up using the proximal environment.

The multi-contact locomotion problem is hard, and the Loco3D project aims at decoupling it into specific subproblems that are simpler to solve and able to work

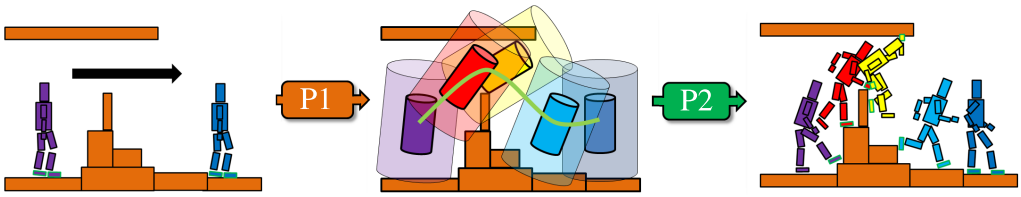


Figure 5.3: Overview of our two-stage framework. Given a path request between start and goal positions (left image), P1 is the problem of computing a guide path in the space of equilibrium feasible root configurations. We achieve this by defining a geometric condition, the reachability condition (abstracted with the transparent cylinders on the middle image). P2 is then the problem of extending the path into a discrete sequence of contact configurations.

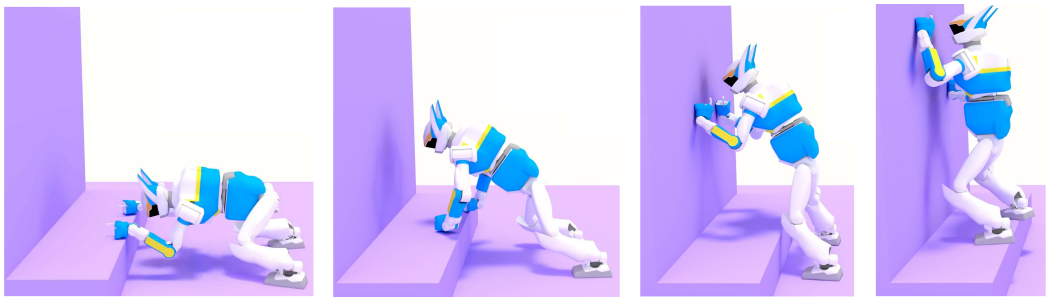


Figure 5.4: Representation of some stages in the HRP-2 standing up simulation. The robot needs to use contacts with its environment to change poses and get to the final goal: standing up.

together in a coherent framework. The ultimate goal of the Loco3D project is then to define the multi-contact locomotion solution through a pipeline (see Fig. 5.3) and apply it to both HRP-2 and the TALOS humanoid platform designed by PAL robotics [Stasse 2017].

5.1.3 The need of a perception method

Since the generation of motions on the robot depends on the environment and the robot's configuration in it, a precise estimation of the surroundings of the robot is necessary as well as an estimate of its location and the direction of the gravity vector, which needs to be taken into account for the dynamics. Furthermore, in the case of multi-contact locomotion, a dense representation of the environment is required to compute feasible contact sequences, *e.g.* in Fig. 5.4. All these perception requirements can be fulfilled by SLAM methods. Another objective the perception method faces is the need to estimate the robot's localisation at control-rate, that is 1 kHz. To meet these challenges, we made the choice to contribute to the WOLF project and design a SLAM method that we could rely on the long-term and augment relating to future needs.

5.2 WOLF

5.2.1 Introduction

WOLF (acronym for Windowed LOcalization Frames) is a C++ library to solve localisation problems in mobile robotics, such as SLAM, map-based localisation, or visual odometry. The approach contemplates the coexistence of multiple sensors of different kinds, be them synchronised or not. WOLF organises the state vectors as a set of key-frames, defining the robot trajectory, plus other states such as landmarks in the environment or sensor parameters for calibration, WOLF computes error vectors given the available measurements in that window. The philosophy developed through WOLF is to have a structure for having the data accessible and organised, called the WOLF Tree, plus some functionalities for managing this data. It requires, on one side, several front-ends (one per sensor, or per sensor type), and a back-end constituted by the solver.

Efforts are made to have WOLF interface able with many kinds of solvers, including filters and nonlinear optimisers. Thus the choice of the solver is left to users so that they can use filters such as EKF, error-state KF, iterative EKF, information filters or others as well as nonlinear or incremental optimisers. The task of interfacing WOLF with these solvers is relegated to wrappers, which are coded out of WOLF. WOLF is currently used with Google Ceres through such a wrapper. However, we also provide an experimental wrapper-less solver for incremental, sparse, nonlinear optimisation based on the QR decomposition (implementing the iSAM algorithm [Kaess 2008b]).

5.2.2 Motivations

SLAM solutions are usually provided with a specific implementation of the solver. When the problem is tackled with a filtering approach such as one of those presented in Section 1.2, the solution can be provided as an application relying on no other external requirement. A good example of such solution is RT-SLAM [Roussillon 2011] specifically designed to provide a practical remedy for online experiments on robots requiring real-time execution. RT-SLAM is an EKF-based solution implemented in C++. However, if one tries to cope with providing a SLAM solution based on nonlinear optimisation methods, it usually comes as an association of a front-end coping with the sensory data processing, a back-end solving for the optimal solution, with a specific philosophy regarding the structure of the problem.

The efforts put in the development of the WOLF library is motivated by the need to propose multiple front-ends that can be used together with state-of-the-art back-ends for optimisation purposes. Several challenges need to be faced in order to construct the front-ends. Processing multiple sensors providing different types of information at different rates is one part of the problem that is of main importance. Sensor fusion is still a main topic of research as uncertainty management [Murphy 1998], fusion schemes [Kaempchen 2003, Spanos 2005], decision-making [Klein 2004] and other problems are still open challenges. Besides, there

is a wide range of possible applications from calibration [Olsson 2016, Hol 2011] and robotics [De Silva 2017, Olofsson 2016, Santoso 2017, Novak 2015] to body pose estimation [Kok 2015] and tracking problems [Chavez-Garcia 2016]. Thus there is an interest for an easy-to-use library with a flexible architecture to take care of sensor fusion tasks. WOLF is also designed to fulfil this demand in the context of localisation solving.

WOLF is a collaborative project from the Institut de Robòtica i Informàtica Industrial (IRI), PAL Robotics and LAAS-CNRS to propose a state-of-the-art SLAM solution we could rely on and update ourselves for long-term projects. But it may also be of interest to other research communities.

5.2.3 Objectives

On the application side, the objective of the WOLF development project is to propose an open-source visual-odometry and SLAM solutions based on factor graphs that can be extended and modified according to users' specific needs. The initial goal includes proposing methods to use most common sensors (cameras, LIDAR, IMUs, ...) with a plug-and-play approach making their use easier. The sensors are set to communicate one with others so that a key-frame creation results in a coherent factor graph containing all the information from one key-frame to another given the data timestamps provided by the sensors. The use of this library can be extended to other purposes such as calibration, inertial navigation, simulation, or even be used to investigate other research purposes such as Active SLAM problems that are defined as a combination of SLAM and path planning problems.

5.2.4 Architecture

The architecture behind WOLF is complex but its global view is made intuitive. The basic Wolf structure is a tree of base classes reproducing the elements of the robotic problem. This is called the Wolf Tree (Fig. 5.5). It has a robot, with sensors, with a trajectory formed by key-frames, and the map with its landmarks. These base classes can be derived to build specialised classes with specific behaviours based on the base classes functionalities. The Wolf Tree connectivity is augmented with the constraints linking different parts of it, becoming a real network of relations. This network is equivalent to the factor graph that would be solved by graphical models and nonlinear optimisation. Wrappers are the ones transferring the Wolf structure into a factor graph that can be provided to the solver.

Sensors produce *Captures* that are entities designed to contain the data provided by first ones. The use of captures is critical for state estimation process and calibration methods. Indeed, whereas sensors contain static intrinsic and extrinsic parameters, the captures can be given dynamic sensor parameters that are to be estimated and taken into account all along the trajectory. Finally, the captures' data are processed by *Processor* classes providing a common base to handle information provided by captures. Processors can be used to detect specific features in

images or for tracking purposes as well as integrating motion measurements. They also are at the crux of the key-frame creation process and thus also the initiators of the constraints generation in the graph.

The resulting architecture allows one to easily create a factor graph-based representation of the problem using multiple sensors. The classes can be specialised according to the users' will using polymorphism, virtual inheritance and templates. The choice of the solver is let to the user and only needs a wrapper to connect it to WOLF.

5.2.5 Contributions

5.2.5.1 Implementation and team members

The development of WOLF is coordinated by Prof. Joan SOLÀ from IRI who also contributed to the development of almost all the subparts of the library. Joan Vallvé Navarro is a PhD student focusing on the aspects of information metrics for localisation and mapping. His main contributions to the project lie in his research towards a more efficient use of factor graphs [Vallvé 2018, Vallvé 2017, Vallvé 2015] and the expertise he could bring on the development side. He also provided the wrapper that allows one to use WOLF with Google Ceres. Jérémie Deray is also a PhD from PAL robotics with major contributions regarding the loop closure strategy and the implementation of Bag of Words techniques in the library. Angel Santamaria Navarro is a post-doctoral researcher working at IRI. His main research interests are related to unmanned aerial vehicles (UAV). along with estimation methods based on visual information [Santamaria-Navarro 2017b, Santamaria-Navarro 2017a]. He is a main contributor to the visual perception method implementation. Finally, my main personal contributions in WOLF lie in the development of the preintegration method (Section 3.3) and its integration the library as well as the use of IMU sensors and both static and dynamic state estimation methods.

5.2.5.2 Related Projects

WOLF is involved in several projects as a SLAM solution provider. This SLAM solution is to be used on straddle carriers for port scenarios in the context of the Logimatic project [Gonzalez 2016] aiming at finding more efficient and cost-effective ways to handle containers in ports based on innovative technical solutions. It will also be used for autonomous vehicles in warehouses with the ASTI project as well as for the perception side of aerial robotic system with multiple arms and advances manipulation capabilities (AEROARMS [Ollero 2015]) to be applied in industrial inspection and maintenance. Finally, WOLF's SLAM solution will also be used by the GAUSS project [Jimenez-Gonzalez 2018] for robust GNSS localisation of large drone floats.

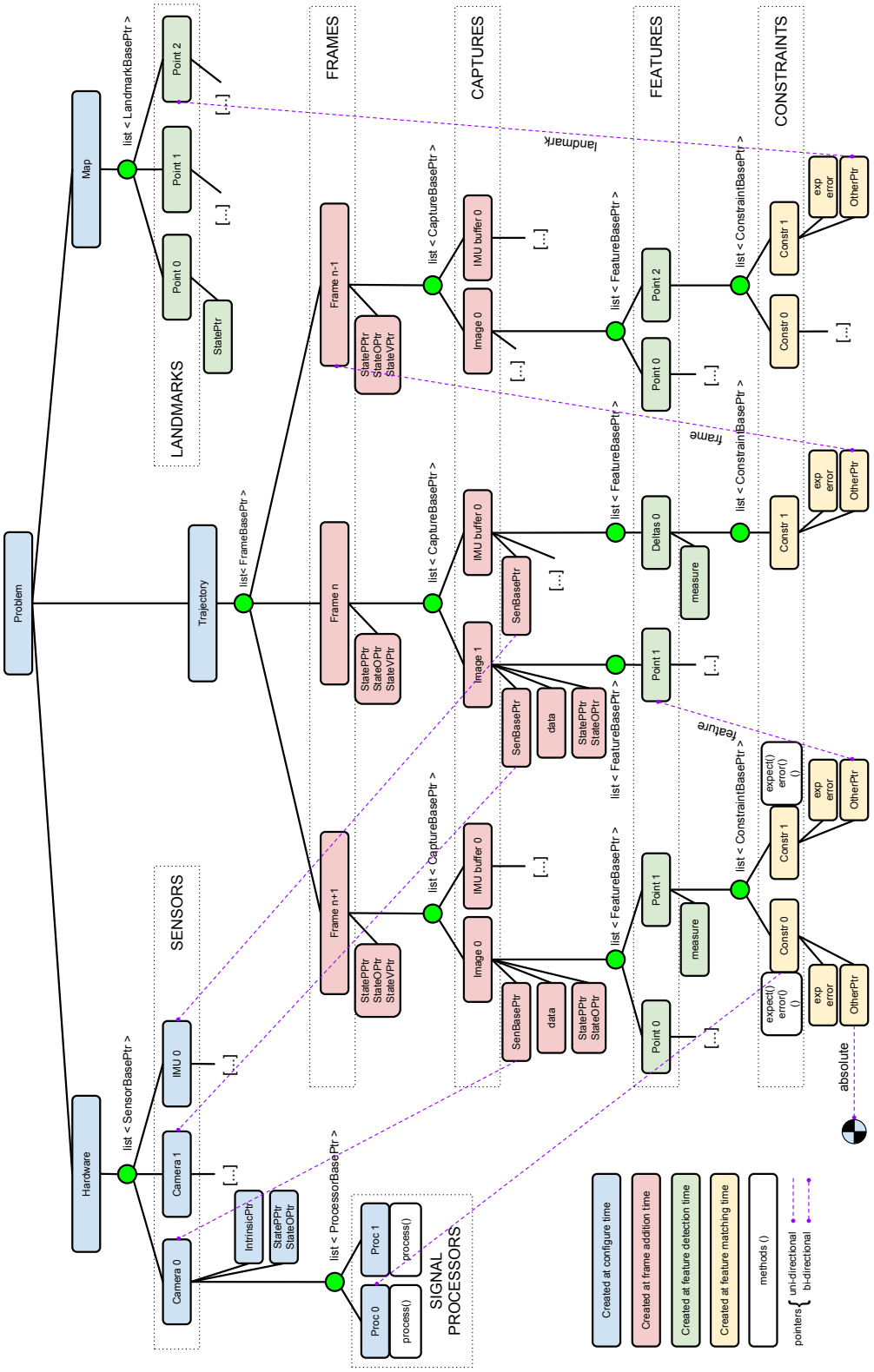


Figure 5.5: Representation of the WOLF Tree.

Conclusion

The work presented in this thesis was motivated by the need of a high frequency state estimation method for closed loop control of the humanoid robot HRP-2. Our objective is to give a first part of the answer to the following question: is it relevant to use low-cost IMUs on different places of a robot in order to get a new sensing capability? The question behind is: can we use this new sensing capability to overcome some estimation problems on legged robots? In order to complete this objective, we investigated the use of inertial estimation techniques applied to pedestrian navigation and robot locomotion using a factor graph representation and a probabilistic formulation. We also investigated the use of IMUs on an end-effector of the robot with the intention to give it a new sensing capability that would be complementary with the kinematic odometry and provide some useful motion-related information. Despite real-time requirements, we showed that batch-optimisation techniques can be applied for estimations with a very high update rate (1 kHz in our case). This can be achieved by using sensors in an efficient manner and by optimising the implementations.

In **Chapter 3**, we described the *quaternion-based preintegration technique* for an efficient use of IMU sensors giving a physical explanation to the preintegrated values. Additional efforts were made in order to present the derivatives clearly in the hope of *helping the community to better understand and to facilitate the implementation of this method requiring analytic derivatives to fully benefit of its efficiency*. The estimation method was presented with a factor graph containing all the kinematic and odometry constraints. The optimisation was performed with a nonlinear least squares solver. The proposed method was applied to pedestrian navigation and tests have been conducted on the humanoid robot HRP-2 with an IMU on its foot to estimate its trajectory. These investigations raised the concern of a better integration of the IMU in the robot itself as opposed to the plug-and-play approach that we used. However, despite the problems introduced by the mechanical vibrations that are propagated through the robot, we were able to have *coherent estimates of the foot's trajectory* on simple scenarii.

In **Chapter 4**, we investigated *self-calibration procedures on manifold using graph-based motion estimation*. This work extends some solutions usually expressed in the mathematical language of the Group Theory to any kind of manifold while presenting the method with a vocabulary that is more familiar to roboticists. The described method can be used for calibration procedures using various sensors.

In **Chapter 5**, we presented the WOLF library to which the GEPETTO group is contributing to in the context of the Loco3D project. Special attention was given to the integration of IMU sensors in WOLF not only for integration and estimation purposes with the preintegration method on manifolds, but also for the estimation of intrinsic and extrinsic parameters.

As a result, the use of IMUs on the end-effectors of legged-robots deserves to be

studied further. Indeed, despite the results highlighted in this thesis, the application on a legged-robot seems to require specific cares. The estimation method explained through this thesis was tested on the pedestrian case before it was successfully applied on the HRP-2 robot with simple study cases. Some perspectives need to be investigate to have a clear understanding in whether this new sensing capability is relevant or not. And with these perspectives, new challenges are likely to be faced.

Perspectives

Short-term perspectives

Several perspectives arise from the work presented in this thesis. In the short-term, the high frequency estimation technique should be fused with SLAM to investigate the use of an auto-calibrating kHz SLAM method with low-cost sensors. The resulting SLAM method could be of interest for applications with humanoid robots and UAVs to close the control loop with a fast perception method. Furthermore, the self-calibration method is currently capable of estimating intrinsic and extrinsic parameters of a sensor. However, the intrinsinc parameters of the IMU are currently limited to the estimation of the biases. Those intrinsic parameters should be extended to the correction of the scale factor and misalignment errors due to the sensor in order to cope with effects introduced by motions with high dynamics. The result would then be an easy IMU calibration method that may be of interest for research communities using UAVs or investigating motion analysis. Furthermore, the application presented through the pedestrian navigation estimation could be extended to use with other sensors such as ultrawidebands [Kok 2015] or magnetometers [Kok 2013].

Mid-term perspectives

With regard to mid-term perspectives it would be interesting to investigate the use of IMUs directly integrated to end-effectors for legged locomotion and interaction with the environment. This requires specific design as low-level integration of IMU sensors in the system which needs to be planned in advance. The investigations detailed by Rotella et al. [Rotella 2016] are a first step towards a stronger integration of inertial measurement units in robots for improved state estimation. The authors managed to improve the quality of the velocity and acceleration estimates for the control feedback. Furthermore, the generalisation of the preintegration method by including the possibility to work on $SE3$ leads to possible investigations on closed-loop control using velocity measurements.

Long-term perspectives

In a long-term, a logical perspective would be to improve the state estimation of a humanoid robot with IMUs attached not only to end-effectors but to each segment providing synchronised data. Although a state estimation providing estimates at

control-rate may seem appealing in this context, the importance of vibrations propagated through the robot and measured by the IMUs may constitute an obstacle if tackled with low-cost sensors. Furthermore, the higher the number of sensors involved in the fusion strategy, the more computationally demanding the estimation process may be. Thus we could investigate the limits of a fusion scheme designed for real-time whole-body control of a humanoid robots. Another interesting investigation could be the design of an estimator that would be able to not only estimate the state of the system on a wide time interval but also to consider only specific states of the system given sensory inputs. Such an estimator would be a first step towards realtime reactions to unexpected events while complying with the whole-body control phylosophy of robots.

Another long term motivational perspective oriented to biomechanical applications would be the use of several IMUs with sensor fusion methods in the context of neuroprosthesis and exoskeletons to estimate the dynamics of the system and close the control-loop. There are interesting works on passive prosthesis [Johnson 2009] and powered exoskeletons [Bortole 2015, Esquenazi 2012]. Furthermore, the use of IMUs in the design of prosthesis has been of growing interest to improve the control of prosthetic devices [Aisen 2007, Lauretti 2016, Krasoulis 2017, Seel 2016b]. Nevertheless these researches usually only cope with standard IMU integration and filter-based methods. Similarly, IMUs are also of interest in the analysis of the human motions [Zhao 2016, Seel 2016a, Seel 2014]. Due to the importance recently given to IMUs, my personal belief is that the methods presented through this manuscript can be of interest for these research areas. Indeed, real-time control is one of the specifications prosthetic devices need to satisfy. Various methods can be used to achieve this objective while minimising the embedded computing power such as the use of filters [Fakoorian 2016, Lenzi 2014] or continuous integration along with stance phase detection [Moreno 2006]. These methods can benefit from an IMU integration method providing real-time results. Although a unified architecture for sensor fusion providing real-time estimates may also be useful, the computation requirements from these applications make the development of such a solution very challenging. The development of motion analysis systems using IMUs along with other sensors as described in [Mueller 2011, Steins 2014, Šlajpah 2014] seems to result in encouraging outcomes that may gain advantages from future customisable sensor fusion solutions. Thus yet another long-term perspective would be to make possible the use of WOLF on systems with limited computing power.

This thesis resulted in the development of a high-rate estimation method based on an IMU and batch-optimisation techniques. Due to the use of low-cost sensors, special attention was required for the estimation of intrinsic parameters and we investigated the use of a common preintegration formulation for the calibration of various sensors. The contributions developped through this thesis come with a sight on the preintegration subject from a roboticist point of view. Thus efforts were made to explain this point of view in a way that roboticists may understand

with more ease so that they can use these methods and bring their contributions to the subject. We believe the preintegration theory and the use of multiple IMUs to be interesting methods to be used together and with other sensors.

Appendix A

Probabilistic formulation

Hereafter are introduced main notions the reader may need to understand the work presented in this thesis. The presented topics are not exclusive.

A.1 Probability

Probability is a mathematical formulation of knowledge, representing the probability distribution of values that are likely to be taken by random variables. The more likely the variable is to fall in the range of a specific value the higher the corresponding probability will be. Probabilities are given by a function encoding the density of probability that the statement "the random variable X takes a value in the range $x \pm dx$" becomes true relatively to the other values that can be taken. Such function is usually called *probability density function (pdf)*. The probability density that X actually takes the value x is thus given by the function $p_X(x)$

$$p_X(x) \triangleq \lim_{dx \to 0} \frac{P(|X - x| < dx)}{dx} \tag{A.1}$$

where $P(A) \in [0, 1]$ is the probability for the event A to be true. For the sake of clarity, $p_X(x)$ will be noted $p(x)$ in the following.

A.2 Expectation

Expectation is a central notion of the probability theory and denotes intuitively the long-run average value one can expect the random variable to take. The expectation of a random variable X is a linear function defined by

$$\mathbb{E}[X] \triangleq \sum_x xp(x), \text{in discrete form} \tag{A.2}$$

$$\mathbb{E}[X] \triangleq \int_{-\infty}^{\infty} xp(x)dx, \text{in continuous form} \tag{A.3}$$

From these definitions, we can introduce some other notions that will be used hereafter

$$\mu_x \triangleq \mathbb{E}[x] \tag{A.4}$$

$$\sigma_x \triangleq \sqrt{\mathbb{E}\left[(x-\mu)^2\right]} \tag{A.5}$$

$$\mathbb{V}[x] \triangleq \sigma_x^2 = \mathbb{E}\left[(x-\mu)^2\right] \tag{A.6}$$

where μ_x, σ_x and σ_x^2 are respectively the mean, the standard deviation and the variance of X. They will be noted respectively μ, σ and σ^2 hereafter. An intuitive understanding of these notions is given in the following introduction related to the Gaussian distribution.

A.3 Gaussian distribution

The Gaussian distribution (also called *Gauss*, *Laplace-Gauss* or *normal* distribution) is a continuous probability distribution. It is an important distribution used in statistics to describe the densities distribution of probability density functions, used not only to model measurement errors but also in natural and social sciences. Describing the real value of a random variable whose distribution is actually unknown with a Gaussian one is a common assumption.

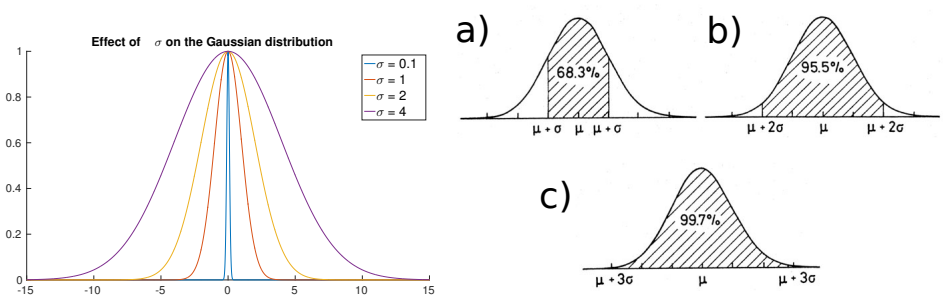
The probability density of a 1D random variable $\mathbf{x}$ following a normal distribution is described by the mean (or expectation) μ of this distribution and its variance σ^2

$$p(\mathbf{x}) \sim \mathcal{N}(\mu,\, \sigma^2)) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}(\frac{x-\mu}{\sigma})^2\right) \tag{A.7}$$

Note that σ is the standard deviation of the distribution. Its effect on the distribution itself and its meaning are shown in Fig. A.1. The generalised normal distribution for multidimensional random variables is usually described by the mean $\boldsymbol{\mu}$ and the covariance $\boldsymbol{\Sigma}$. n being the dimension of the random variable $\mathbf{x}$, we have

$$p(\mathbf{x}) \sim \mathcal{N}(\boldsymbol{\mu},\, \boldsymbol{\Sigma})) = \frac{1}{det(\boldsymbol{\Sigma})^{\frac{1}{2}}(2\pi)^{\frac{n}{2}}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right) \tag{A.8}$$

where Σ is a $n \times n$ matrix. We can notice that if $n = 1$ then (A.8) simplifies to (A.7). Fig. A.2 gives a representation of points sampled with a 2D Gaussian distribution on a plane for visualisation purpose. This figure also helps to figure out the meaning of the standard variation and its representation in the two-dimension case.



(a) Variation of 1D Gaussian with σ (b) Confidence intervals

Figure A.1: Visualisation of a 1D Gaussian distribution. **Left**: We can see on this figure the effect of the standard deviation, and thus the variance, on the shape of the Gaussian distribution. As σ goes towards 0, the Gaussian distribution takes the shape of a Dirac function. **Right**: The area under the Gaussian curve is a specificity to keep in mind when one uses a Gaussian assumption to model errors to interpret the measurements. Presenting an estimation x with a confidence interval of σ means that the true value has a probability of about 0.68 of lying in the interval $[x-\sigma, x+\sigma]$ (see a)). The real value has also a probability of 0.955 and 0.997 to be in the intervals of respectively $[x-1\sigma, x+2\sigma]$ (see b)) and $[x-3\sigma, x+3\sigma]$ (see c)) [Leo 1988]

A.4 Bayes' rule

Bayes' rule (also called *Bayes' theorem* or *Bayes' law*), describes the probability of an event based on prior knowledge. This law is described by a simple mathematical formulation. Let us note A and B events, $P(A)$ and $P(B)$ are respectively the likelihood for events A and B to occur. Bayes' rule allows one to describe the likelihood A (respectively B) to occur given prior knowledge about B (respectively A) and noted $P(A|B)$ (respectively $P(B|A)$).

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{A.9}$$

For specific examples of how this theorem can be used, I would recommend the interested reader to have a look at the first chapter of [Stone 2013].

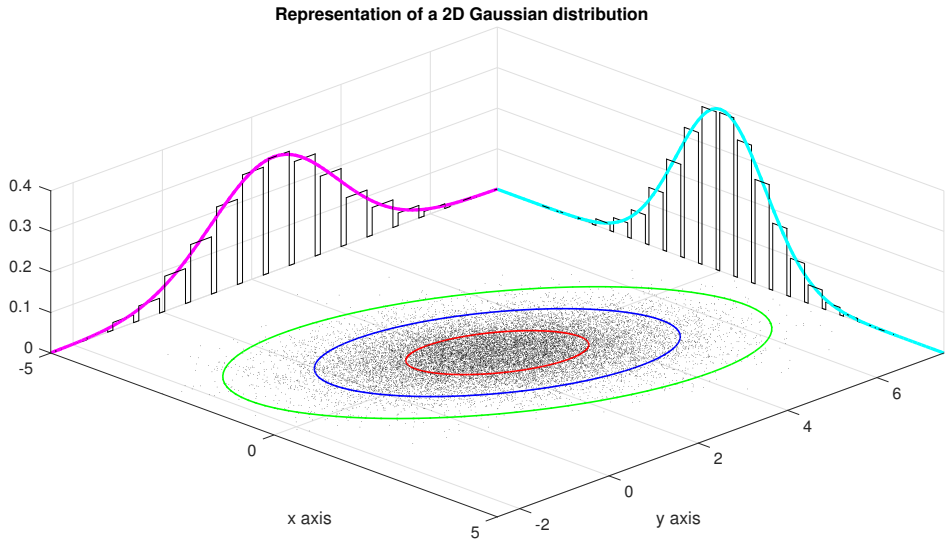


Figure A.2: Representation of a 2D Gaussian distribution with mean $\boldsymbol{\mu} = [0, 2.5]$ and the associated 2×2 covariance matrix with diagonal set to $[1, 2]$ and the two other parameters are set to 0.6. On the {x,y} plane of this figure are distributed 20000 points following the 2D Gaussian. Are also represented on this plane the σ, 2σ and 3σ confidence area respectively in red, blue and green.

Appendix B

Quaternion and rotation representation

The following description of quaternions comes from [Solà Ortega 2016].

B.1 Definition of quaternion

Quaternions are a tool amongst others to describe orientations. Lets us note Q a quaternion defined in the space of quaternions $\mathbb{H}$. Q is defined as

$$Q = a + bi + cj + dk \in \mathbb{H} \tag{B.1}$$

where $\{a, b, c, d\} \in \mathbb{R}$ and $\{i, j, k\}$ are three imaginary unit number so that

$$i^2 = j^2 = k^2 = ijk = -1 \tag{B.2}$$

$$ij = ji = k,\ jk = -kj = i,\ ki = -ik = j. \tag{B.3}$$

We define numbers that we will refer to as *pure quaternions* in the tri-dimensional imaginary subspace of $\mathbb{H}$,

$$Q = bi + cj + dk \ \in \ \mathbb{I}^3 \ \subset \mathbb{H} \tag{B.4}$$

Please note that the definition mentioned above are derived from the specific quaternion definition in Eq. (B.1) where the real part is the first number of the quaternion. An alternative quaternion definition sets this scalar part at the end so that $Q = ai + bj + ck + d$.

For a more convenient use, quaternions can be defined as a sum of a scalar number and an imaginary vector so that

$$Q = q_w + q_x i + q_y j + q_z k \ \Leftrightarrow \ Q = q_w + \mathbf{q}_v \tag{B.5}$$

where q_w is referred to as the *scalar* part, and $\mathbf{q}_v = q_x i + q_y j + q_z k = (q_x, q_y, q_z)$ as the *imaginary* or *vector* part. Finally, a quaternion can be defined using a pair of scalar and vector

$$Q =< q_w, \mathbf{q}_v > \tag{B.6}$$

or also with as a four-dimensional vector $\mathbf{q}$

$$\mathbf{q} \triangleq \begin{bmatrix} q_w \\ \mathbf{q}_v \end{bmatrix} = \begin{bmatrix} q_w \\ q_x \\ q_y \\ q_z \end{bmatrix} \tag{B.7}$$

and for sake of clarity, we may sometimes define the quaternions using only subscript indices in the following so that $\mathbf{q} = (w, x, y, z)$. We may also abuse of the notations so that

$$Q = q_w + \mathbf{q}_v = \begin{bmatrix} q_w \\ \mathbf{q}_v \end{bmatrix} \ , \ q_w = \begin{bmatrix} q_w \\ \mathbf{0}_v \end{bmatrix} \ , \ \mathbf{q}_v = \begin{bmatrix} 0 \\ \mathbf{q}_v \end{bmatrix} \tag{B.8}$$

where q_w and $\mathbf{q}_v$ are used to represent respectively *real quaternions* and *pure quaternions*.

B.2 quaternion properties

B.2.0.3 Sum / difference

The sum and difference of two quaternions are straightforward and defined so that

$$\mathbf{q}_1 \ \pm \ \mathbf{q}_2 = \begin{bmatrix} q_{w1} \\ \mathbf{q}_{v1} \end{bmatrix} \ \pm \ \begin{bmatrix} q_{w2} \\ \mathbf{q}_{v2} \end{bmatrix} = \begin{bmatrix} q_{w1} \ \pm \ q_{w2} \\ \mathbf{q}_{v1} \ \pm \ \mathbf{q}_{v2} \end{bmatrix} \tag{B.9}$$

The sum is commutative and associative

$$\mathbf{q}_1 \ + \ \mathbf{q}_2 = \mathbf{q}_2 \ + \ \mathbf{q}_1 \tag{B.10}$$

$$\mathbf{q}_1 \ + \ (\mathbf{q}_2 \ + \ \mathbf{q}_3) = (\mathbf{q}_1 \ + \ \mathbf{q}_2) \ + \ \mathbf{q}_3 \tag{B.11}$$

where the identity operation is operated using the zero quaternion defined by $\mathbf{q}_0 = 0$ and the inverse quaternion is given by the negative $-\mathbf{q}$.

B.2.0.4 Product

The product operator of quaternions is referred to using the symbol $\otimes$ and defined so that

$$\mathbf{q}_1 \ \otimes \ \mathbf{q}_2 = \begin{bmatrix} q_{w1} \ q_{w2} \ - \ \mathbf{q}_{v1}^T \mathbf{q}_{v2} \\ q_{w1}\mathbf{q}_{v2} \ + \ q_{w2}\mathbf{q}_{v1} \ + \ \mathbf{q}_{v1} \times \mathbf{q}_{v2} \end{bmatrix} \tag{B.12}$$

The product of quaternions is not commutative in the general case due to use of the cross-product $\times$

$$\mathbf{q}_1 \otimes \mathbf{q}_2 \ \neq \ \mathbf{q}_2 \otimes \mathbf{q}_1 \tag{B.13}$$

The commutativity is, however, true if the cross-product term yields to 0, which is the case when either one of the quaternions is actually a pure real quaternion or when both vector $\mathbf{q}_{v1}$ and $\mathbf{q}_{v2}$ are parallel. The quaternion product is however associative

$$(\mathbf{q}_1 \otimes \mathbf{q}_2) \otimes \mathbf{q}_3 \;=\; \mathbf{q}_1 \otimes (\mathbf{q}_2 \otimes \mathbf{q}_3) \tag{B.14}$$

and distributed over the sum so that

$$\mathbf{q}_1 \otimes (\mathbf{q}_2 + \mathbf{q}_3) \;=\; \mathbf{q}_1 \otimes \mathbf{q}_2 \;+\; \mathbf{q}_1 \otimes \mathbf{q}_3 \tag{B.15}$$

Quaternions endowed with the product operation $\otimes$ form a non-commutative group in which the identity element is $\mathbf{q}_{\mathrm{I}} = 1 = \begin{bmatrix} 1 \\ \mathbf{0}_v \end{bmatrix}$.

B.2.1 Conjugate

The conjugate of a quaternion is defined by

$$\mathbf{q}^* \triangleq q_w - \mathbf{q}_v = \begin{bmatrix} q_w \\ -\mathbf{q}_v \end{bmatrix} \tag{B.16}$$

with the following properties

$$\mathbf{q} \otimes \mathbf{q}^* = \mathbf{q}^* \otimes \mathbf{q} = \begin{bmatrix} q_w^2 + q_x^2 + q_y^2 + q_z^2 \\ \mathbf{0}_v \end{bmatrix} \tag{B.17}$$

$$(\mathbf{q}_1 \otimes \mathbf{q}_2)^* = \mathbf{q}_1^* \otimes \mathbf{q}_2^* \tag{B.18}$$

B.2.1.1 Norm

The norm of a quaternion is defined by

$$\|\mathbf{q}\| = \sqrt{\mathbf{q} \otimes \mathbf{q}^*} = \sqrt{\mathbf{q}^* \otimes \mathbf{q}} = \sqrt{q_w^2 + q_x^2 + q_y^2 + q_z^2} \tag{B.19}$$

so that

$$\|\mathbf{q}_{w1} \otimes \mathbf{q}_{w2}\| = \|\mathbf{q}_{w1}\| \|\mathbf{q}_{w2}\| \tag{B.20}$$

B.2.2 Inverse

The inverse quaternion $\mathbf{q}^{-1}$ is defined so that

$$\mathbf{q} \otimes \mathbf{q} = \mathbf{q}^{-1} \otimes \mathbf{q} = \mathbf{q}_{\mathrm{I}} \tag{B.21}$$

and can be computed with

$$\mathbf{q}^{-1} = \mathbf{q}^* / \left\| \mathbf{q}^2 \right\| \tag{B.22}$$

B.3 Quaternion and rotations

Let us define a vector $\mathbf{x}$ in $\mathbb{R}^3$ and $\mathbf{R}$ the rotation matrix that transforms $\mathbf{x}$ into $\mathbf{x}'$

$$\mathbf{x}' = \mathbf{R}\mathbf{x}$$

There exist an equivalent rotation $\mathbf{q}$ described in the quaternion space $\mathbb{H}$ to perform the same transformation so that

$$\bar{\mathbf{x}}' = \mathbf{q} \otimes \bar{\mathbf{x}} \otimes \mathbf{q}^* \tag{B.23}$$

where $\bar{\mathbf{x}}$ is in quaternion form, that is

$$\bar{\mathbf{x}} = xi + yj + zk = \begin{bmatrix} 0 \\ \bar{\mathbf{x}} \end{bmatrix} \tag{B.24}$$

However, since we can easily distinguish the context by the presence of the quaternion product operator $\otimes$, we will abuse of the notation in the following and write

$$\mathbf{x}' = \mathbf{q} \otimes \mathbf{x} \otimes \mathbf{q}^* = \mathbf{R}\mathbf{x} \tag{B.25}$$

The expression of the rotation matrix that is equivalent to the quaternion can be found by developing both $\mathbf{q} \otimes \mathbf{x} \otimes \mathbf{q}^*$ and $\mathbf{R}\mathbf{x}$ and by identifying terms. This yields to

$$\mathbf{R}\{\mathbf{q}\} = \begin{bmatrix} q_w^2 + q_x^2 - q_y^2 - q_z^2 & 2(q_x q_y - q_w q_z) & 2(q_x q_y + q_w q_y) \\ 2(q_x q_y + q_w q_z) & q_w^2 - q_x^2 + q_y^2 - q_z^2 & 2(q_y q_z - q_w q_x) \\ 2(q_x q_z - q_w q_y) & 2(q_y q_z + q_w q_x) & q_w^2 - q_x^2 - q_y^2 + q_z^2 \end{bmatrix} \tag{B.26}$$

$\mathbf{R}\{\mathbf{q}\}$ has the following properties with respect to quaternions

$$\mathbf{R}\{\mathbf{q}_\mathrm{I}\} = \mathbf{I} \tag{B.27}$$

$$\mathbf{R}\{-\mathbf{q}\} = \mathbf{R}\{\mathbf{q}\} \tag{B.28}$$

$$\mathbf{R}\{\mathbf{q}^*\} = \mathbf{R}\{\mathbf{q}\}^T \tag{B.29}$$

$$\mathbf{R}\{\mathbf{q}_1 \otimes \mathbf{q}_2\} = \mathbf{R}\{\mathbf{q}_1\}\mathbf{R}\{\mathbf{q}_2\} \tag{B.30}$$

where one can the following properties:

- the identity quaternion encodes the null rotation (B.27)
- a quaternion $\mathbf{q}$ and its negative encode the same rotation (B.28)
- the conjugate quaternion encodes the inverse rotation (B.29)
- the quaternion product and rotation matrices compose rotations in the same order (B.30)

From (B.30) we can find out how the composition of rotations work. Indeed, let us define $\mathbf{x}_A$ a 3D vector so that

$$\mathbf{x}_A = \mathbf{R}_{AB}\ \mathbf{R}_{BC}\ \mathbf{x}_C$$

where the chain rule makes it simple to understand the successive rotation operations. The rotation matrix $\mathbf{R}_{BC}$ is used to rotate the 3D vector $\mathbf{x}_C$ into $\mathbf{x}_B = \mathbf{R}_{BC}\ \mathbf{x}_C$ and $\mathbf{R}_{AB}$ is the rotation matrix that is used to transform $\mathbf{x}_B$ into $\mathbf{x}_A$. The equivalent transformation using quaternion is given by

$$\mathbf{x}_A = \mathbf{q}_{AB} \otimes \mathbf{q}_{BC} \otimes \mathbf{x}_C \otimes \mathbf{q}_{BC}^* \otimes \mathbf{q}_{AB}^*$$

and recalling (B.29), we finally get

$$\mathbf{x}_A = \underbrace{\mathbf{q}_{AB} \otimes \mathbf{q}_{BC}}_{\mathbf{q}_{AC}} \otimes\ \mathbf{x}_C\ \ \otimes \underbrace{\mathbf{q}_{CB} \otimes \mathbf{q}_{BA}}_{\mathbf{q}_{AC}^*}$$

We define the quaternion-by-vector product $\odot$ so that

$$\mathbf{q} \odot \mathbf{v} \triangleq \mathbf{q} \otimes \mathbf{v} \otimes \mathbf{q}^*\ , \tag{B.31}$$

where $\mathbf{q}^*$ denotes the dual quaternion of $\mathbf{q}$. The symbol $\otimes$ stands for the product of quaternions and $\odot$ corresponds to the quaternion-by-vector which performs a 3D rotation of an input vector $\mathbf{v}$. Notice that if $\mathbf{R}$ is the rotation matrix equivalent to the quaternion $\mathbf{q}$, then $\mathbf{q} \odot \mathbf{v} = \mathbf{R}\,\mathbf{v}$. This straightforward equivalence enables us to define all the forthcoming IMU pre-integration algebra in a way that allows a direct transcription between the S^3 and $SO(3)$ spaces of representation.

Appendix C

Definition of derivatives on manifolds

C.1 Definition of the derivatives in S^3 and $SO(3)$

C.1.1 Exp and Log maps in S^3 and $SO(3)$

We use vectorized versions of the exponential and logarithmic maps in the rotation groups S^3 (quaternion) and $SO(3)$ (rotation matrix), and denote them with capitalized names Exp() and Log() (see Fig. C.1, left). They operate directly on the vector space $\mathbb{R}^3$, and use either quaternions for S^3,

$$\mathbf{q} = \mathrm{Exp}(\boldsymbol{\theta}) \triangleq \begin{bmatrix} \cos(\theta/2) \\ \mathbf{u}\sin(\theta/2) \end{bmatrix} \qquad \in \mathbb{H} \tag{C.1a}$$

$$\theta\mathbf{u} = \mathrm{Log}(\mathbf{q}) \triangleq 2\,\mathbf{q}_v \frac{\arctan(\|\mathbf{q}_v\|, q_w)}{\|\mathbf{q}_v\|} \qquad \in \mathbb{R}^3\ , \tag{C.1b}$$

where $(\theta, \mathbf{u}) \in \mathbb{R}^3$ is the angle-axis representation of the rotation, and $\mathbf{q} \triangleq (q_w, \mathbf{q}_v) \in \mathbb{H}$ is a quaternion with its real and imaginary parts. For rotation matrices in $SO(3)$ we have,

$$\mathbf{R} = \mathrm{Exp}(\boldsymbol{\theta}) \triangleq \mathbf{I} + \sin\theta\,[\mathbf{u}]_\times + (1-\cos\theta)\,[\mathbf{u}]_\times^2 \ \in \mathbb{R}^{3\times 3} \tag{C.2a}$$

$$\theta\mathbf{u} = \mathrm{Log}(\mathbf{R}) \triangleq \frac{\theta(\mathbf{R}-\mathbf{R}^\top)^\vee}{2\sin\theta} \quad \in \mathbb{R}^3\ , \tag{C.2b}$$

with $\theta = \cos^{-1}\left(\frac{\mathrm{trace}(\mathbf{R})-1}{2}\right)$, and where $\bullet^\vee$, known as the *vee* operator, is the inverse of the *skew* operator $[\bullet]_\times$. Their exact form ($\mathbf{q}$ or $\mathbf{R}$) is always clear by the context. Since the quaternion implementation is one of our contributions, in the following we will refer to the rotation groups S^3 and $SO(3)$ with the unique name S^3, although everything applies equally to $SO(3)$.

C.1.2 The additive and subtractive operators in S^3 and $SO(3)$

The 'plus' operator, $\oplus : S^3 \times \mathbb{R}^3 \to S^3$, composes a reference element $\mathsf{R} \in S^3$ with a (often small) rotation specified by a vector of $\boldsymbol{\theta} \in \mathbb{R}^3$ that is tangent to the S^3 manifold at R, yielding an element $\mathsf{S} \in S^3$ (see Fig. C.1, right). The 'minus' operator, $\ominus : S^3 \times S^3 \to \mathbb{R}^3$ is the inverse of the above. These operators are defined

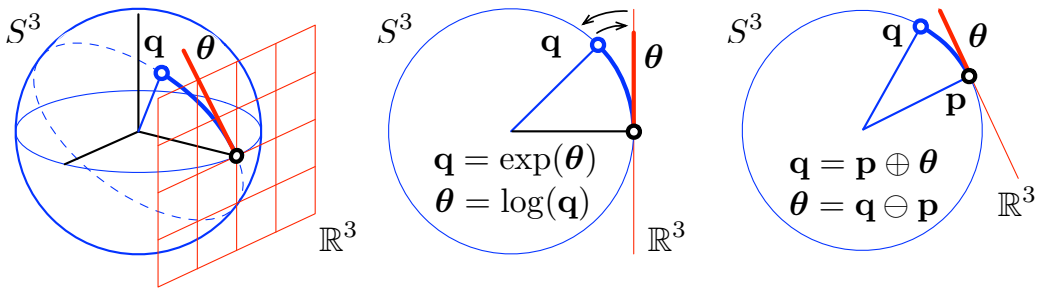


Figure C.1: The S^3 manifold is a unit sphere in $\mathbb{R}^4$, here represented by a unit circle (blue), where all unit quaternions live. The tangent space to the manifold is the hyperplane $\mathbb{R}^3$, here represented by a line (red). The Exp() and Log() operators map elements of $\mathbb{R}^3$ to/from elements of S^3. The $\oplus$ and $\ominus$ operators relate elements of the manifold with elements in the tangent space. (Likewise, these figures illustrate the $SO(3)$ manifold.)

for both $\mathbf{q}$ and $\mathbf{R}$,

$$\mathbf{q} = \mathbf{p} \oplus \boldsymbol{\theta} \triangleq \mathbf{p} \otimes \mathrm{Exp}(\boldsymbol{\theta}) \tag{C.3}$$

$$\mathbf{S} = \mathbf{R} \oplus \boldsymbol{\theta} \triangleq \mathbf{R}\,\mathrm{Exp}(\boldsymbol{\theta}) \tag{C.4}$$

$$\boldsymbol{\theta} = \mathbf{q} \ominus \mathbf{p} \triangleq \mathrm{Log}(\mathbf{p}^* \otimes \mathbf{q}) \tag{C.5}$$

$$\boldsymbol{\theta} = \mathbf{S} \ominus \mathbf{R} \triangleq \mathrm{Log}(\mathbf{R}^\top \mathbf{S})\ . \tag{C.6}$$

C.1.3 The four possible derivative definitions

For functions $f : \mathbb{R}^m \to \mathbb{R}^n$, the derivative is defined classically using the standard operators $\{+, -\}$,

$$\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} \triangleq \lim_{\delta\mathbf{x}\to 0} \frac{f(\mathbf{x}+\delta\mathbf{x}) - f(\mathbf{x})}{\delta\mathbf{x}} \qquad \in \mathbb{R}^{n\times m}\ ; \tag{C.7}$$

for functions $g : S^3 \to S^3$, we use the operators $\{\oplus, \ominus\}$,

$$\frac{\partial g(\mathsf{R})}{\partial \boldsymbol{\theta}} \triangleq \lim_{\delta\boldsymbol{\theta}\to 0} \frac{g(\mathsf{R}\oplus\delta\boldsymbol{\theta}) \ominus g(\mathsf{R})}{\delta\boldsymbol{\theta}} \qquad \in \mathbb{R}^{3\times 3}\ ; \tag{C.8}$$

for functions $h : \mathbb{R}^m \to S^3$, we use $\{+, \ominus\}$,

$$\frac{\partial h(\mathbf{x})}{\partial \mathbf{x}} \triangleq \lim_{\delta\mathbf{x}\to 0} \frac{h(\mathbf{x}+\delta\mathbf{x}) \ominus h(\mathbf{x})}{\delta\mathbf{x}} \qquad \in \mathbb{R}^{3\times m}\ ; \tag{C.9}$$

and for functions $k : S^3 \to \mathbb{R}^n$, we use $\{\oplus, -\}$,

$$\frac{\partial k(\mathsf{R})}{\partial \boldsymbol{\theta}} \triangleq \lim_{\delta\boldsymbol{\theta}\to 0} \frac{k(\mathsf{R}\oplus\delta\boldsymbol{\theta}) - k(\mathsf{R})}{\delta\boldsymbol{\theta}} \qquad \in \mathbb{R}^{n\times 3}\ . \tag{C.10}$$

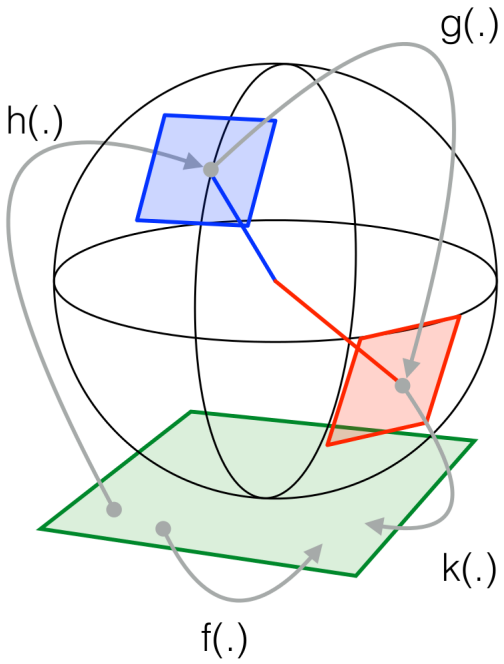


Figure C.2: Representation of the derivatives operation. We consider a situation so that variable can be either in the $\mathbb{R}^n$ or in one of the manifolds (blue and red spaces). Then we define the functions $f : \mathbb{R}^m \to \mathbb{R}^n$, $g : S^3 \to S^3$, $h : \mathbb{R}^m \to S^3$ and $k : S^3 \to \mathbb{R}^n$.

It might be worth noticing that all these Jacobians are independent of the representation chosen (S^3 or $SO(3)$).

C.1.4 Right Jacobian of S^3 and $SO(3)$

We define the right Jacobian as,

$$\mathbf{J}_r(\boldsymbol{\theta}) \triangleq \frac{\partial \operatorname{Exp}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \in \mathbb{R}^{3\times 3} \ , \tag{C.11}$$

and implement it using (C.9). It admits the closed forms [Chirikjian 2012, pag. 40],

$$\mathbf{J}_r(\boldsymbol{\theta}) = \mathbf{I} - \frac{1-\cos\|\boldsymbol{\theta}\|}{\|\boldsymbol{\theta}\|^2}[\boldsymbol{\theta}]_\times + \frac{\|\boldsymbol{\theta}\|-\sin\|\boldsymbol{\theta}\|}{\|\boldsymbol{\theta}\|^3}[\boldsymbol{\theta}]_\times^2 \tag{C.12}$$

$$\mathbf{J}_r^{-1}(\boldsymbol{\theta}) = \mathbf{I} + \frac{1}{2}[\boldsymbol{\theta}]_\times + \left(\frac{1}{\|\boldsymbol{\theta}\|^2} - \frac{1+\cos\|\boldsymbol{\theta}\|}{2\|\boldsymbol{\theta}\|\sin\|\boldsymbol{\theta}\|}\right)[\boldsymbol{\theta}]_\times^2 \ . \tag{C.13}$$

C.1.5 Examples

Since our defined derivatives map tangent Cartesian spaces, and these spaces coincide for the 3D rotation manifolds of S^3 and $SO(3)$, *i.e.*, $\boldsymbol{\theta} = \log(\mathbf{q}) = \log(\mathbf{R})$, it follows that the Jacobians are independent of the representation used ($\mathbf{q}$ or $\mathbf{R}$). We

can thus consider generic 3D rotation elements, and note them with the sans-serif font R.

C.1.5.1 Function $\mathbb{R}^3 \to S^3$

The function $f(\boldsymbol{\omega}) = \mathrm{Exp}(\boldsymbol{\omega}\delta t)$ produces elements of S^3 from vectors $\boldsymbol{\omega} \in \mathbb{R}^3$. Its Jacobian with respect to $\boldsymbol{\omega}$ follows from (C.9), but is better obtained from (C.11) and the chain rule,

$$\frac{\partial \mathrm{Exp}(\boldsymbol{\omega}\delta t)}{\partial \boldsymbol{\omega}} = \frac{\partial \mathrm{Exp}(\boldsymbol{\omega}\delta t)}{\partial (\boldsymbol{\omega}\delta t)} \frac{\partial (\boldsymbol{\omega}\delta t)}{\partial \boldsymbol{\omega}} = \mathbf{J}_r(\boldsymbol{\omega}\delta t)\delta t \ .$$

C.1.5.2 Function $S^3 \times \mathbb{R}^3 \to \mathbb{R}^3$

The rotation $f(\mathsf{R}, \mathbf{v}) = \mathbf{q} \odot \mathbf{v} = \mathbf{R}\,\mathbf{v}$ (coresponding to rotation action) produces vectors of $\mathbb{R}^3$ from elements $\mathsf{R} \in S^3$ and vectors $\mathbf{v} \in \mathbb{R}^3$. The first Jacobian is defined by (C.10) and developed as

$$\begin{aligned}
\mathbf{J}_{\mathsf{R}}^{\mathsf{R}\cdot\mathbf{v}} = \frac{\partial \mathbf{q} \odot \mathbf{v}}{\partial \boldsymbol{\theta}} = \frac{\partial \mathbf{R}\mathbf{v}}{\partial \boldsymbol{\theta}} &\triangleq \lim_{\delta\boldsymbol{\theta}\to 0} \frac{(\mathbf{R} \oplus \delta\boldsymbol{\theta})\mathbf{v} - \mathbf{R}\mathbf{v}}{\delta\boldsymbol{\theta}} \\
= \lim_{\delta\boldsymbol{\theta}\to 0} \frac{\mathbf{R}\,\mathrm{Exp}(\delta\boldsymbol{\theta})\mathbf{v} - \mathbf{R}\mathbf{v}}{\delta\boldsymbol{\theta}} &= \lim_{\delta\boldsymbol{\theta}\to 0} \frac{\mathbf{R}\cdot(\mathbf{I} + [\delta\boldsymbol{\theta}]_\times)\mathbf{v} - \mathbf{R}\mathbf{v}}{\delta\boldsymbol{\theta}} \\
&= \lim_{\delta\boldsymbol{\theta}\to 0} \frac{\mathbf{R}\,[\delta\boldsymbol{\theta}]_\times\,\mathbf{v}}{\delta\boldsymbol{\theta}} = \lim_{\delta\boldsymbol{\theta}\to 0} \frac{-\mathbf{R}\,[\mathbf{v}]_\times\,\delta\boldsymbol{\theta}}{\delta\boldsymbol{\theta}} = -\mathbf{R}\,[\mathbf{v}]_\times
\end{aligned}$$

where we used the properties $\mathrm{Exp}(\delta\boldsymbol{\theta}) \approx \mathbf{I} + [\delta\boldsymbol{\theta}]_\times$ and $[\mathbf{a}]_\times\,\mathbf{b} = -[\mathbf{b}]_\times\,\mathbf{a}$. The second Jacobian is defined by (C.7) and yields,

$$\frac{\partial \mathbf{q} \odot \mathbf{v}}{\partial \mathbf{v}} = \frac{\partial \mathbf{R}\mathbf{v}}{\partial \mathbf{v}} \triangleq \lim_{\partial\mathbf{v}\to 0} \frac{\mathbf{R}\cdot(\mathbf{v} + \partial\mathbf{v}) - \mathbf{R}\mathbf{v}}{\partial\mathbf{v}} = \mathbf{R} \ .$$

C.1.5.3 Function $S^3 \times S^3 \to S^3$

The function $f(\mathsf{Q}, \mathsf{R}) = \mathbf{q} \otimes \mathbf{r} = \mathbf{Q}\,\mathbf{R}$ produces rotation composition. Its Jacobians are computed from (C.8), using the property $\mathrm{Exp}(\mathbf{R}\boldsymbol{\theta}) = \mathbf{R}\,\mathrm{Exp}(\boldsymbol{\theta})\mathbf{R}^\top$,

$$\begin{aligned}
\mathbf{J}_{\mathsf{Q}}^{\mathsf{Q}\circ\mathsf{R}} = \frac{\partial \mathbf{q}(\boldsymbol{\theta}) \otimes \mathbf{r}}{\partial \boldsymbol{\theta}} = \frac{\partial \mathbf{Q}(\boldsymbol{\theta})\,\mathbf{R}}{\partial \boldsymbol{\theta}} &= \lim_{\delta\boldsymbol{\theta}\to 0} \frac{\mathrm{Log}\left((\mathbf{Q}\mathbf{R})^\top(\mathbf{Q}\,\mathrm{Exp}(\delta\boldsymbol{\theta})\mathbf{R})\right)}{\delta\boldsymbol{\theta}} \\
&= \lim_{\delta\boldsymbol{\theta}\to 0} \frac{\mathrm{Log}\left(\mathbf{R}^\top\,\mathrm{Exp}(\delta\boldsymbol{\theta})\mathbf{R}\right)}{\delta\boldsymbol{\theta}} \\
&= \lim_{\delta\boldsymbol{\theta}\to 0} \frac{\mathrm{Log}\left(\,\mathrm{Exp}(\mathbf{R}^\top\delta\boldsymbol{\theta})\right)}{\delta\boldsymbol{\theta}} = \mathbf{R}^\top \ , \\
\mathbf{J}_{\mathsf{R}}^{\mathsf{Q}\circ\mathsf{R}} = \frac{\partial \mathbf{q} \otimes \mathbf{r}(\boldsymbol{\phi})}{\partial \boldsymbol{\phi}} = \frac{\partial \mathbf{Q}\,\mathbf{R}(\boldsymbol{\phi})}{\partial \boldsymbol{\phi}} &= \lim_{\delta\boldsymbol{\phi}\to 0} \frac{\mathrm{Log}\left((\mathbf{Q}\mathbf{R})^\top(\mathbf{Q}\mathbf{R}\,\mathrm{Exp}(\delta\boldsymbol{\phi}))\right)}{\delta\boldsymbol{\phi}} \\
&= \lim_{\delta\boldsymbol{\phi}\to 0} \frac{\mathrm{Log}\left(\,\mathrm{Exp}(\delta\boldsymbol{\phi})\right)}{\delta\boldsymbol{\phi}} = \mathbf{I} \ .
\end{aligned}$$

C.1.6 Adjoint

We have from (4.21)

$$\begin{aligned}\mathrm{Adj}_{\mathsf{R}}\tau &\triangleq \log(\mathsf{R}\circ\exp(\tau^{\wedge})\circ\mathsf{R}^{-1})^{\vee}\\ &= \log(\mathbf{R}\exp([\tau]_{\times})\mathbf{R}^{\top})^{\vee}\\ &= \log(\exp(\mathbf{R}\,[\tau]_{\times}\,\mathbf{R}^{\top}))^{\vee}\\ &= \log(\exp([(\mathbf{R}\tau)]_{\times}))^{\vee}\\ &= ((\mathbf{R}\tau)^{\wedge})^{\vee} = \mathbf{R}\tau\end{aligned}$$

therefore

$$\mathrm{Adj}_{\mathsf{R}} = \mathbf{R}\ . \tag{C.14}$$

C.2 Maps, operators and derivatives of S^1 and $SO(2)$

C.2.1 Exp and log maps

Lift and retract are implemented via exponential maps directly from the scalar tangent space $\mathfrak{so}(2) = \mathcal{T}_{SO(2)} \simeq \mathcal{T}_{S^1} \simeq \mathbb{R}$. For S^1 we have,

$$\mathbf{z} = \exp(\theta) = \cos\theta + i\sin\theta \qquad \in \mathbb{C} \tag{C.15}$$

$$\theta = \log(\mathbf{z}) = \arctan(\mathrm{Im}(\mathbf{z}), \mathrm{Re}(\mathbf{z})) \qquad \in \mathbb{R}\ , \tag{C.16}$$

whereas for $SO(2)$,

$$\mathbf{R} = \exp(\theta) = \begin{bmatrix}\cos\theta & -\sin\theta\\ \sin\theta & \cos\theta\end{bmatrix} \qquad \in \mathbb{R}^{2\times 2} \tag{C.17}$$

$$\theta = \log(\mathbf{R}) = \arctan(r_{21}, r_{11}) \qquad \in \mathbb{R}\ . \tag{C.18}$$

C.2.2 Inverse, composition

We have

$$\mathsf{R}(\theta)^{-1} = \mathsf{R}(-\theta) \tag{C.19}$$

$$\mathsf{Q}\circ\mathsf{R} = \mathsf{R}\circ\mathsf{Q}\ , \tag{C.20}$$

since planar rotations are commutative. It follows that,

$$\exp(\theta_1+\theta_2) = \exp(\theta_1)\circ\exp(\theta_2) \tag{C.21}$$

$$\log(\mathsf{Q}\circ\mathsf{R}) = \log(\mathsf{Q}) + \log(\mathsf{R}) \tag{C.22}$$

$$\mathsf{Q}\ominus\mathsf{R} = \theta_Q - \theta_R\ . \tag{C.23}$$

C.2.3 Derivative blocks

From (4.23) and the properties above, the following scalar derivative blocks become trivial,

$$\mathbf{J}_{\mathsf{R}}^{\mathsf{R}^{-1}} = \lim_{\delta\theta\to 0} \frac{(-\theta - \delta\theta) - (-\theta)}{\delta\theta} = -1 \tag{C.24}$$

$$\mathbf{J}_{\mathsf{Q}}^{\mathsf{Q}\circ\mathsf{R}} = \lim_{\delta\theta_Q\to 0} \frac{(\theta_Q + \delta\theta_Q + \theta_R) - (\theta_Q + \theta_R)}{\delta\theta_Q} = 1 \tag{C.25}$$

$$\mathbf{J}_{\mathsf{R}}^{\mathsf{Q}\circ\mathsf{R}} = \lim_{\delta\theta_R\to 0} \frac{(\theta_Q + \theta_R + \delta\theta_R) - (\theta_Q + \theta_R)}{\delta\theta_R} = 1 \tag{C.26}$$

$$\mathbf{J}_r(\theta) = \lim_{\delta\theta\to 0} \frac{(\theta + \delta\theta) - \theta}{\delta\theta} = 1 \tag{C.27}$$

For the rotation action $\mathsf{R}\cdot\mathbf{v}$ we have

$$\begin{aligned}\mathbf{J}_{\mathsf{R}}^{\mathsf{R}\cdot\mathbf{v}} &= \lim_{\delta\theta\to 0} \frac{\mathbf{R}\exp(\delta\theta)\mathbf{v} - \mathbf{R}\mathbf{v}}{\delta\theta} \\ &= \lim_{\delta\theta\to 0} \frac{\mathbf{R}(\mathbf{I} + [\delta\theta]_\times)\mathbf{v} - \mathbf{R}\mathbf{v}}{\delta\theta} \\ &= \lim_{\delta\theta\to 0} \frac{\mathbf{R}\,[\delta\theta]_\times\,\mathbf{v}}{\delta\theta} = \mathbf{R}\,[1]_\times\,\mathbf{v} \quad \in \mathbb{R}^{2\times 1}\end{aligned} \tag{C.28}$$

with $[\theta]_\times = \begin{bmatrix} 0 & -\theta \\ \theta & 0 \end{bmatrix}$, and

$$\mathbf{J}_{\mathbf{v}}^{\mathsf{R}\cdot\mathbf{v}} = \frac{\partial \mathbf{R}\mathbf{v}}{\partial \mathbf{v}} = \mathbf{R}\ . \tag{C.29}$$

Bibliography

[Agarwal] Sameer Agarwal, Keir Mierle and Others. *Ceres Solver*. `http://ceres-solver.org`. (Cited in pages 56 and 68.)

[Agarwal 2018] Prashant Agarwal, Aman Gupta, Gaurav Verma, Himanshu Verma, Ashish Sharma and Sandeep Banarwal. *Wireless Monitoring and Indoor Navigation of a Mobile Robot Using RFID*. In Nature Inspired Computing, pages 83–90. Springer, 2018. (Cited in page 13.)

[Ahn 2012] SungHwan Ahn, Sukjune Yoon, Seungyong Hyung, Nosan Kwak and Kyung Shik Roh. *On-board odometry estimation for 3D vision-based SLAM of humanoid robot*. In Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on, pages 4006–4012. IEEE, 2012. (Cited in page 53.)

[Aisen 2007] Benjamin Baruch Aisen. *An inertial measurement-based gait detection system for active leg prostheses*. PhD thesis, Massachusetts Institute of Technology, 2007. (Cited in page 125.)

[Angermann 2010] Michael Angermann, Patrick Robertson, Thomas Kemptner and Mohammed Khider. *A high precision reference data set for pedestrian navigation using foot-mounted inertial sensors*. In Indoor Positioning and Indoor Navigation (IPIN), 2010 International Conference on, pages 1–6. IEEE, 2010. (Cited in pages 64 and 66.)

[Angermann 2012] Michael Angermann and Patrick Robertson. *Footslam: Pedestrian simultaneous localization and mapping without exteroceptive sensors—hitchhiking on human perception and cognition*. Proceedings of the IEEE, vol. 100, no. Special Centennial Issue, pages 1840–1848, 2012. (Cited in page 66.)

[Atchuthan 2018] Dinesh Atchuthan, Angel Santamaria-Navarro, Nicolas Mansard, Olivier Stasse and Joan Solà. *Odometry Based on Auto-Calibrating Inertial Measurement Unit Attached to the Feet*. In European Control Conference 2018, Limassol, Cyprus, June 2018. KIOS Research and Innovation Center of Excellence. (Cited in page 7.)

[Azarbayejani 1995] Ali Azarbayejani and Alex P Pentland. *Recursive estimation of motion, structure, and focal length*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 17, no. 6, pages 562–575, 1995. (Cited in page 16.)

[Barshan 1995] Billur Barshan and Hugh F Durrant-Whyte. *Inertial navigation systems for mobile robots*. IEEE Transactions on Robotics and Automation, vol. 11, no. 3, pages 328–342, 1995. (Cited in page 41.)

[Beravs 2012] Tadej Beravs, Janez Podobnik and Marko Munih. *Three-axial accelerometer calibration using Kalman filter covariance matrix for online estimation of optimal sensor orientation.* IEEE Transactions on Instrumentation and Measurement, vol. 61, no. 9, pages 2501–2511, 2012. (Cited in page 48.)

[Bestaven 2012] Emma Bestaven, Etienne Guillaud and Jean-René Cazalets. *Is "circling" behavior in humans related to postural asymmetry?* PloS one, vol. 7, no. 9, page e43861, 2012. (Cited in page 10.)

[Bloom 1988] Floyd E Bloom, Arlyne Lazerson, Laura Hofstadter*et al.* *Brain, mind, and behavior.* 1988. (Cited in page 1.)

[Bortole 2015] Magdo Bortole, Anusha Venkatakrishnan, Fangshi Zhu, Juan C Moreno, Gerard E Francisco, Jose L Pons and Jose L Contreras-Vidal. *The H2 robotic exoskeleton for gait rehabilitation after stroke: early findings from a clinical study.* Journal of neuroengineering and rehabilitation, vol. 12, no. 1, page 54, 2015. (Cited in page 125.)

[Brossard 2017] Martin Brossard, Silvère Bonnabel and Jean-Philippe Condomines. *Unscented Kalman filtering on Lie groups.* In Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on, pages 2485–2491. IEEE, 2017. (Cited in page 24.)

[Bruno 2011] Luigi Bruno and Patrick Robertson. *Wislam: Improving footslam with wifi.* In Indoor Positioning and Indoor Navigation (IPIN), 2011 International Conference on, pages 1–10. IEEE, 2011. (Cited in page 66.)

[Carpentier 2016] Justin Carpentier, Steve Tonneau, Maximilien Naveau, Olivier Stasse and Nicolas Mansard. *A versatile and efficient pattern generator for generalized legged locomotion.* In Robotics and Automation (ICRA), 2016 IEEE International Conference on, pages 3555–3561. IEEE, 2016. (Cited in pages 54 and 116.)

[Carpentier 2017] Justin Carpentier, Andrea Del Prete, Steve Tonneau, Thomas Flayols, Florent Forget, Alexis Mifsud, Kevin Giraud, Dinesh Atchuthan, Pierre Fernbach, Rohan Budhiraja, Mathieu Geisert, Joan Solà, Olivier Stasse and Nicolas Mansard. *Multi-contact Locomotion of Legged Robots in Complex Environments – The Loco3D project.* In RSS Workshop on Challenges in Dynamic Legged Locomotion, page 3p., Boston, United States, July 2017. (Cited in pages 115 and 116.)

[Carpentier 2018] Justin Carpentier, Florian Valenza, Nicolas Mansard*et al.* *Pinocchio: fast forward and inverse dynamics for poly-articulated systems.* https://stack-of-tasks.github.io/pinocchio, 2015–2018. (Cited in page 74.)

[Caruso 2017] David Caruso, Alexandre Eudes, Martial Sanfourche, David Vissière and Guy le Besnerais. *Robust indoor/outdoor navigation through magneto-visual-inertial optimization-based estimation.* In Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on, pages 4402–4409. IEEE, 2017. (Cited in page 8.)

[Chaumette 1991] Francois Chaumette, Patrick Rives and Bernard Espiau. *Positioning of a robot with respect to an object, tracking it and estimating its velocity by visual servoing.* In Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on, pages 2248–2253. IEEE, 1991. (Cited in page 4.)

[Chavez-Garcia 2016] Ricardo Omar Chavez-Garcia and Olivier Aycard. *Multiple sensor fusion and classification for moving object detection and tracking.* IEEE Transactions on Intelligent Transportation Systems, vol. 17, no. 2, pages 525–534, 2016. (Cited in page 119.)

[Chdid 2011] Dima Chdid, Raja Oueis, Hiam Khoury, Daniel Asmar and Imad Elhajj. *Inertial-vision sensor fusion for pedestrian localization.* In Robotics and Biomimetics (ROBIO), 2011 IEEE International Conference on, pages 1695–1701. IEEE, 2011. (Cited in page 66.)

[Cheuk 2012] Chi Ming Cheuk, Tak Kit Lau, Kai Wun Lin and Yunhui Liu. *Automatic calibration for inertial measurement unit.* In Control Automation Robotics & Vision (ICARCV), 2012 12th International Conference on, pages 1341–1346. IEEE, 2012. (Cited in page 48.)

[Chirikjian 2012] Gregory S. Chirikjian. Stochastic models, information theory, and lie groups, volume 2: Analytic Methods and Modern Applications of *Applied and Numerical Harmonic Analysis.* Birkhäuser„ Basel, 2012. (Cited in pages 88 and 139.)

[Chiuso 2002] Alessandro Chiuso, Paolo Favaro, Hailin Jin and Stefano Soatto. *Structure from motion causally integrated over time.* IEEE transactions on pattern analysis and machine intelligence, vol. 24, no. 4, pages 523–535, 2002. (Cited in page 16.)

[Cho 2005] Seong Yun Cho and Chan Gook Park. *A Calibration Technique for a Redundant IMU Containing Low-Grade Inertial Sensors.* ETRI journal, vol. 27, no. 4, pages 418–426, 2005. (Cited in page 48.)

[Codas 2010] A Codas, M Devy and C Lemaire. *Robot localization algorithm using odometry and RFID technology.* IFAC Proceedings Volumes, vol. 43, no. 16, pages 569–574, 2010. (Cited in page 13.)

[Comport 2010] Andrew I Comport, Ezio Malis and Patrick Rives. *Real-time quadrifocal visual odometry.* The International Journal of Robotics Research, vol. 29, no. 2-3, pages 245–266, 2010. (Cited in page 4.)

[Davison 2003] Andrew J Davison. *Real-time simultaneous localisation and mapping with a single camera.* In null, page 1403. IEEE, 2003. (Cited in page 17.)

[Day 1964] RH Day and G Singer. *Spatial aftereffects within and between kinesthesis and vision.* Journal of Experimental Psychology, vol. 68, no. 4, page 337, 1964. (Cited in page 1.)

[De Silva 2017] Varuna De Silva, Jamie Roche and Ahmet Kondoz. *Fusion of LiDAR and camera sensor data for environment sensing in driverless vehicles.* arXiv preprint arXiv:1710.06230, 2017. (Cited in page 119.)

[Dellaert 2006] Frank Dellaert and Michael Kaess. *Square Root SAM: Simultaneous localization and mapping via square root information smoothing.* The International Journal of Robotics Research, vol. 25, no. 12, pages 1181–1203, 2006. (Cited in page 29.)

[Dissanayake 2000] MWM Gamini Dissanayake, Paul Newman, Hugh F Durrant-Whyte, Steve Clark and M Csorba. *An experimental and theoretical investigation into simultaneous localisation and map building.* In Experimental robotics VI, pages 265–274. Springer, 2000. (Cited in page 27.)

[Douc 2005] Randal Douc and Olivier Cappé. *Comparison of resampling schemes for particle filtering.* In Image and Signal Processing and Analysis, 2005. ISPA 2005. Proceedings of the 4th International Symposium on, pages 64–69. IEEE, 2005. (Cited in page 27.)

[Doucet 2000] Arnaud Doucet, Simon Godsill and Christophe Andrieu. *On sequential Monte Carlo sampling methods for Bayesian filtering.* Statistics and computing, vol. 10, no. 3, pages 197–208, 2000. (Cited in page 25.)

[Doucet 2001] Arnaud Doucet, Nando De Freitas and Neil Gordon. *An introduction to sequential Monte Carlo methods.* In Sequential Monte Carlo methods in practice, pages 3–14. Springer, 2001. (Cited in page 25.)

[Eade] Ethan Eade. *Lie Groups for 2D and 3D Transformations.* Technical Report. (Cited in page 92.)

[Eade 2006] Ethan Eade and Tom Drummond. *Scalable monocular SLAM.* In Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Volume 1, pages 469–476. IEEE Computer Society, 2006. (Cited in pages 12 and 17.)

[Eade 2013] Ethan Eade. *Lie groups for 2d and 3d transformations.* URL http://ethaneade. com/lie. pdf, revised Dec, 2013. (Cited in page 88.)

[Einicke 1999] Garry A Einicke and Langford B White. *Robust extended Kalman filtering.* IEEE Transactions on Signal Processing, vol. 47, no. 9, pages 2596–2599, 1999. (Cited in page 24.)

[Endres 2012] Felix Endres, Jürgen Hess, Nikolas Engelhard, Jürgen Sturm, Daniel Cremers and Wolfram Burgard. *An evaluation of the RGB-D SLAM system.* In Robotics and Automation (ICRA), 2012 IEEE International Conference on, pages 1691–1696. IEEE, 2012. (Cited in page 28.)

[Esquenazi 2012] Alberto Esquenazi, Mukul Talaty, Andrew Packel and Michael Saulino. *The ReWalk powered exoskeleton to restore ambulatory function to individuals with thoracic-level motor-complete spinal cord injury.* American journal of physical medicine & rehabilitation, vol. 91, no. 11, pages 911–921, 2012. (Cited in page 125.)

[Everett 1995] HR Everett. Sensors for mobile robots. AK Peters/CRC Press, 1995. (Cited in pages vii and 4.)

[Fakoorian 2016] Seyed Abolfazl Fakoorian, Dan Simon, Hanz Richter and Vahid Azimi. *Ground reaction force estimation in prosthetic legs with an extended Kalman filter.* In Systems Conference (SysCon), 2016 Annual IEEE, pages 1–6. IEEE, 2016. (Cited in page 125.)

[Fallon 2014] Maurice F Fallon, Matthew Antone, Nicholas Roy and Seth Teller. *Drift-free humanoid state estimation fusing kinematic, inertial and lidar sensing.* In Humanoid Robots (Humanoids), 2014 14th IEEE-RAS International Conference on, pages 112–119. IEEE, 2014. (Cited in pages 53 and 76.)

[Featherstone 2014] Roy Featherstone. Rigid body dynamics algorithms. Springer, 2014. (Cited in page 74.)

[Fong 2008] WT Fong, SK Ong and AYC Nee. *Methods for in-field user calibration of an inertial measurement unit without external equipment.* Measurement Science and technology, vol. 19, no. 8, page 085202, 2008. (Cited in page 48.)

[Forster 2015] Christian Forster, Luca Carlone, Frank Dellaert and Davide Scaramuzza. *IMU preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation.* In Robotics: Science and Systems. Georgia Institute of Technology, 2015. (Cited in pages 55, 56, 59, and 61.)

[Forster 2016] Christian Forster, Luca Carlone, Frank Dellaert and Davide Scaramuzza. *On-Manifold Preintegration for Real-Time Visual-Inertial Odometry.* Technical Report, Robotics and Perception Group, University of Zurich, 2016. (Cited in page 8.)

[Forster 2017] Christian Forster, Luca Carlone, Frank Dellaert and Davide Scaramuzza. *On-Manifold Preintegration for Real-Time Visual–Inertial Odometry.* IEEE Transactions on Robotics, vol. 33, no. 1, pages 1–21, 2017. (Cited in pages 7, 8, 71, 100, 101, 102, and 105.)

[Foxlin 2005] Eric Foxlin. *Pedestrian tracking with shoe-mounted inertial sensors.* IEEE Computer graphics and applications, vol. 25, no. 6, pages 38–46, 2005. (Cited in page 68.)

[Germa 2010] Thierry Germa, Frédéric Lerasle, Noureddine Ouadah and Viviane Cadenat. *Vision and RFID data fusion for tracking people in crowds by a mobile robot.* Computer Vision and Image Understanding, vol. 114, no. 6, pages 641–651, 2010. (Cited in page 27.)

[Glanze 1994] Walter D Glanze, Kenneth Anderson, Lois E Anderson*et al.* Mosby's medical, nursing, and allied health dictionary. Mosby, 1994. (Cited in page 1.)

[Golub 2012] Gene H Golub and Charles F Van Loan. Matrix computations, volume 3. JHU Press, 2012. (Cited in page 37.)

[Goncalves 2005] Luis Goncalves, Enrico Di Bernardo, Dave Benson, Marcus Svedman, Jim Ostrowski, Niklas Karlsson and Paolo Pirjanian. *A visual front-end for simultaneous localization and mapping.* In Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on, pages 44–49. IEEE, 2005. (Cited in page 28.)

[Gonzalez 2016] Jesuspablo Gonzalez. *LOGIMATIC: Tight integration of EGNSS and on-board sensors for port vehicle automation.* "https://logimatic-project.eu/", 2016. [Online; accessed 31-Jul-2018]. (Cited in page 120.)

[Grisetti 2007a] Giorgio Grisetti, Cyrill Stachniss and Wolfram Burgard. *Improved techniques for grid mapping with rao-blackwellized particle filters.* IEEE transactions on Robotics, vol. 23, no. 1, pages 34–46, 2007. (Cited in page 27.)

[Grisetti 2007b] Giorgio Grisetti, Cyrill Stachniss, Slawomir Grzonka and Wolfram Burgard. *A Tree Parameterization for Efficiently Computing Maximum Likelihood Maps using Gradient Descent.* In Robotics: Science and Systems, volume 3, page 9, 2007. (Cited in page 29.)

[Groves 2007] Paul D Groves, Graham W Pulford, C Aaron Littlefield, David LJ Nash and Christopher J Mather. *Inertial navigation versus pedestrian dead reckoning: Optimizing the integration.* In Proc. ION GNSS, pages 2043–2055, 2007. (Cited in page 65.)

[Guennebaud 2010] Gaël Guennebaud, Benoît Jacob*et al.* *Eigen v3.* http://eigen.tuxfamily.org, 2010. (Cited in page 37.)

[Hamel 2006] Tarek Hamel, Robert E Mahony*et al.* *Attitude Estimation on SO [3] based on Direct Inertial Measurements.* In ICRA, pages 2170–2175, 2006. (Cited in page 41.)

[Hardegger 2012] Michael Hardegger, Daniel Roggen, Sinziana Mazilu and Gerhard Tröster. *ActionSLAM: Using location-related actions as landmarks in pedestrian SLAM.* In Indoor Positioning and Indoor Navigation (IPIN), 2012 International Conference on, pages 1–10. IEEE, 2012. (Cited in page 66.)

[Hartley 2018] Ross Hartley, Maani Ghaffari Jadidi, Lu Gan, Jiunn-Kai Huang, Jessy W Grizzle and Ryan M Eustice. *Hybrid Contact Preintegration for Visual-Inertial-Contact State Estimation within Factor Graphs.* arXiv preprint arXiv:1803.07531, 2018. (Cited in pages 8 and 76.)

[Hol 2011] Jeroen D Hol. *Sensor fusion and calibration of inertial sensors, vision, ultra-wideband and GPS.* PhD thesis, Linköping University Electronic Press, 2011. (Cited in page 118.)

[Hutchings 1998] Lawrence J Hutchings. *System and method for measuring movement of objects*, March 1998. US Patent 5,724,265. (Cited in page 65.)

[InvenSense] InvenSense. *IMU MPU6050.* https://www.invensense.com. (Cited in page 67.)

[Jimenez-Gonzalez 2018] Adrian Jimenez-Gonzalez. *GAUSS: Galileo-EGNOS as an asset for UTM safety and security.* "https://www.gsa.europa.eu/galileo-egnos-asset-utm-safety-and-security", 2018. [Online; accessed 31-Jul-2018]. (Cited in page 120.)

[Johnson 2009] William Brett Johnson, Stefania Fatone and Steven A Gard. *Walking mechanics of persons who use reciprocating gait orthoses.* Journal of Rehabilitation Research & Development, vol. 46, no. 3, 2009. (Cited in page 125.)

[Johnson 2017] Matthew Johnson, Brandon Shrewsbury, Sylvain Bertrand, Duncan Calvert, Tingfan Wu, Daniel Duran, Douglas Stephen, Nathan Mertins, John Carff, William Rifenburgh*et al. Team IHMC's lessons learned from the DARPA robotics challenge: finding data in the rubble.* Journal of Field Robotics, vol. 34, no. 2, pages 241–261, 2017. (Cited in pages 52 and 53.)

[Kaempchen 2003] Nico Kaempchen and Klaus Dietmayer. *Data synchronization strategies for multi-sensor fusion.* In Proceedings of the IEEE Conference on Intelligent Transportation Systems, pages 1–9, 2003. (Cited in page 118.)

[Kaess 2008a] Michael Kaess. *Incremental smoothing and mapping.* PhD thesis, Georgia Institute of Technology, 2008. (Cited in page 55.)

[Kaess 2008b] Michael Kaess, Ananth Ranganathan and Frank Dellaert. *iSAM: Incremental smoothing and mapping.* IEEE Transactions on Robotics, vol. 24, no. 6, pages 1365–1378, 2008. (Cited in pages 29 and 118.)

[Kaess 2012] Michael Kaess, Hordur Johannsson, Richard Roberts, Viorela Ila, John J Leonard and Frank Dellaert. *iSAM2: Incremental smoothing and mapping using the Bayes tree.* The International Journal of Robotics Research, vol. 31, no. 2, pages 216–235, 2012. (Cited in page 29.)

[Kalman 1960] Rudolph Emil Kalman. *A new approach to linear filtering and prediction problems.* Journal of basic Engineering, vol. 82, no. 1, pages 35–45, 1960. (Cited in page 20.)

[Kanehira 2002] Noriyuki Kanehira, TU Kawasaki, Shigehiko Ohta, T Ismumi, Tadahiro Kawada, Fumio Kanehiro, Shuuji Kajita and Kenji Kaneko. *Design and experiments of advanced leg module (HRP-2L) for humanoid robot (HRP-2) development.* In Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on, volume 3, pages 2455–2460. IEEE, 2002. (Cited in page 72.)

[Kaneko 2015] Kenji Kaneko, Mitsuharu Morisawa, Shuuji Kajita, Shin'ichiro Nakaoka, Takeshi Sakaguchi, Rafael Cisneros and Fumio Kanehiro. *Humanoid robot HRP-2Kai—Improvement of HRP-2 towards disaster response tasks.* In Humanoid Robots (Humanoids), 2015 IEEE-RAS 15th International Conference on, pages 132–139. IEEE, 2015. (Cited in page 53.)

[Karumanchi 2017] Sisir Karumanchi, Kyle Edelberg, Ian Baldwin, Jeremy Nash, Jason Reid, Charles Bergh, John Leichty, Kalind Carpenter, Matthew Shekels, Matthew Gildner*et al. Team RoboSimian: semi-autonomous mobile manipulation at the 2015 DARPA robotics challenge finals.* Journal of Field Robotics, vol. 34, no. 2, pages 305–332, 2017. (Cited in page 52.)

[Kawasaki 1999] Haruhisa Kawasaki, Tsuneo Komatsu, Kazunao Uchiyama and Takashi Kurimoto. *Dexterous anthropomorphic robot hand with distributed tactile sensor: Gifu hand II.* In Systems, Man, and Cybernetics, 1999. IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on, volume 2, pages 782–787. IEEE, 1999. (Cited in page 13.)

[Kerpa 2003] Oliver Kerpa, Karsten Weiss and Heinz Worn. *Development of a flexible tactile sensor system for a humanoid robot.* In Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on, volume 1, pages 1–6. IEEE, 2003. (Cited in page 13.)

[Kim 2018] Dongshin Kim, Seunghak Shin and In So Kweon. *On-Line Initialization and Extrinsic Calibration of an Inertial Navigation System With a Relative Preintegration Method on Manifold.* IEEE Transactions on Automation Science and Engineering, vol. 15, no. 3, 2018. (Cited in page 8.)

[Klein 2004] Lawrence A Klein and Lawrence A Klein. Sensor and data fusion: a tool for information assessment and decision making, volume 324. SPIE press Bellingham^ eWA WA, 2004. (Cited in page 118.)

[Kok 2013] Manon Kok, Niklas Wahlström, Thomas B Schön and Fredrik Gustafsson. *MEMS-based inertial navigation based on a magnetic field map.* In Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on, pages 6466–6470. IEEE, 2013. (Cited in page 124.)

[Kok 2015] Manon Kok, Jeroen D Hol and Thomas B Schön. *Indoor positioning using ultrawideband and inertial measurements.* IEEE Transactions on Vehicular Technology, vol. 64, no. 4, pages 1293–1303, 2015. (Cited in pages 119 and 124.)

[Koller 2009] Daphne Koller and Nir Friedman. Probabilistic graphical models: principles and techniques. MIT press, 2009. (Cited in pages 8, 29, and 55.)

[Konolige 2008] Kurt Konolige and Motilal Agrawal. *FrameSLAM: From bundle adjustment to real-time visual mapping.* IEEE Transactions on Robotics, vol. 24, no. 5, pages 1066–1077, 2008. (Cited in pages 17 and 41.)

[Kourogi 2010] Masakatsu Kourogi, Tomoya Ishikawa and Takeshi Kurata. *A method of pedestrian dead reckoning using action recognition.* In Position Location and Navigation Symposium (PLANS), 2010 IEEE/ION, pages 85–89. IEEE, 2010. (Cited in page 65.)

[Krasoulis 2017] Agamemnon Krasoulis, Iris Kyranou, Mustapha Suphi Erden, Kianoush Nazarpour and Sethu Vijayakumar. *Improved prosthetic hand control with concurrent use of myoelectric and inertial measurements.* Journal of neuroengineering and rehabilitation, vol. 14, no. 1, page 71, 2017. (Cited in page 125.)

[Kudruss 2015] Manuel Kudruss, Maximilien Naveau, Olivier Stasse, Nicolas Mansard, Christian Kirches, Philippe Soueres and K Mombaur. *Optimal control for whole-body motion generation using center-of-mass dynamics for predefined multi-contact configurations.* In Humanoid Robots (Humanoids), 2015 IEEE-RAS 15th International Conference on, pages 684–689. IEEE, 2015. (Cited in page 116.)

[Kumar 1989] RV Raja Kumar, Arun Tirumalai and Ramesh C Jain. *A nonlinear optimization algorithm for the estimation of structure and motion parameters.* In Computer Vision and Pattern Recognition, 1989. Proceedings CVPR'89., IEEE Computer Society Conference on, pages 136–143. IEEE, 1989. (Cited in page 16.)

[Kümmerle 2011] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige and Wolfram Burgard. *g 2 o: A general framework for graph optimization.* In Robotics and Automation (ICRA), 2011 IEEE International Conference on, pages 3607–3613. IEEE, 2011. (Cited in page 29.)

[Kwanmuang 2015] Surat Kwanmuang. *FILTERING AND TRACKING FOR A PEDESTRIAN DEAD-RECKONING SYSTEM*. PhD thesis, University of Michigan, May 2015. (Cited in page 65.)

[Kwon 2015] Bo-Kyu Kwon and Soohee Han. *A robust extended Kalman filtering for linearization errors*. In Control, Automation and Systems (ICCAS), 2015 15th International Conference on, pages 1485–1487. IEEE, 2015. (Cited in page 24.)

[Lauretti 2016] Clemente Lauretti, Angelo Davalli, Rinaldo Sacchetti, Eugenio Guglielmelli and Loredana Zollo. *Fusion of M-IMU and EMG signals for the control of trans-humeral prostheses*. In Biomedical Robotics and Biomechatronics (BioRob), 2016 6th IEEE International Conference on, pages 1123–1128. IEEE, 2016. (Cited in page 125.)

[Lenzi 2014] Tommaso Lenzi, Levi Hargrove and Jonathon Sensinger. *Speed-adaptation mechanism: Robotic prostheses can actively regulate joint torque*. IEEE Robotics & Automation Magazine, vol. 21, no. 4, pages 94–107, 2014. (Cited in page 125.)

[Leo 1988] W Leo. *Techniques for nuclear and particle physics experiments*. Nucl Instrum Methods Phys Res, vol. 834, page 290, 1988. (Cited in page 129.)

[Leutenegger 2015] Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart and Paul Furgale. *Keyframe-based visual–inertial odometry using nonlinear optimization*. The International Journal of Robotics Research, vol. 34, no. 3, pages 314–334, 2015. (Cited in page 38.)

[Li 2013] Mingyang Li, Byung Hyung Kim and Anastasios I Mourikis. *Real-time motion tracking on a cellphone using inertial sensing and a rolling-shutter camera*. In Robotics and Automation (ICRA), 2013 IEEE International Conference on, pages 4712–4719. IEEE, 2013. (Cited in pages 38 and 41.)

[Liu 2017] Yi Liu, Zhong Chen, Wenjuan Zheng, Hao Wang and Jianguo Liu. *Monocular Visual-Inertial SLAM: Continuous Preintegration and Reliable Initialization*. Sensors, vol. 17, no. 11, page 2613, 2017. (Cited in page 8.)

[Longuet-Higgins 1981] H Christopher Longuet-Higgins. *A computer algorithm for reconstructing a scene from two projections*. Nature, vol. 293, no. 5828, page 133, 1981. (Cited in page 11.)

[Lourakis 2005] Manolis IA Lourakis. *A brief description of the Levenberg-Marquardt algorithm implemented by levmar*. Foundation of Research and Technology, vol. 4, no. 1, pages 1–6, 2005. (Cited in page 37.)

[Lovelace 1989] Eugene A Lovelace. *Vision and kinesthesis in accuracy of hand movement*. Perceptual and Motor Skills, vol. 68, no. 3, pages 707–714, 1989. (Cited in page 1.)

[Lu 1997] Feng Lu and Evangelos Milios. *Robot pose estimation in unknown environments by matching 2d range scans.* Journal of Intelligent and Robotic systems, vol. 18, no. 3, pages 249–275, 1997. (Cited in page 29.)

[Lupton 2009] Todd Lupton and Salah Sukkarieh. *Efficient Integration of Inertial Observations into Visual SLAM without Initialization.* In IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 2009. (Cited in pages 7, 55, 59, 61, 101, and 105.)

[Magnussen 2015] Øyvind Magnussen, Morten Ottestad and Geir Hovland. *Calibration procedure for an inertial measurement unit using a 6-degree-of-freedom hexapod*, 2015. (Cited in page 47.)

[Marion 2017] Pat Marion, Maurice Fallon, Robin Deits, Andrés Valenzuela, Claudia Pérez D'Arpino, Greg Izatt, Lucas Manuelli, Matt Antone, Hongkai Dai, Twan Koolen*et al. Director: A user interface designed for robot operation with shared autonomy.* Journal of Field Robotics, vol. 34, no. 2, pages 262–280, 2017. (Cited in page 52.)

[Marquardt 1963] Donald W Marquardt. *An algorithm for least-squares estimation of nonlinear parameters.* Journal of the society for Industrial and Applied Mathematics, vol. 11, no. 2, pages 431–441, 1963. (Cited in page 37.)

[Mifsud 2017] Alexis Mifsud. *ESTIMATING AND STABILIZING THE STATUS OF A COMBINING HUMANOID ROBOT.* Theses, Institut national polytechnique de Toulouse (INPT), October 2017. (Cited in pages viii and 79.)

[Mirzaei 2008] Faraz M Mirzaei and Stergios I Roumeliotis. *A Kalman filter-based algorithm for IMU-camera calibration: Observability analysis and performance evaluation.* IEEE transactions on robotics, vol. 24, no. 5, pages 1143–1156, 2008. (Cited in page 38.)

[Montemerlo 2002] Michael Montemerlo, Sebastian Thrun and William Whittaker. *Conditional particle filters for simultaneous mobile robot localization and people-tracking.* In Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on, volume 1, pages 695–701. IEEE, 2002. (Cited in page 27.)

[Montemerlo 2007] Michael Montemerlo and Sebastian Thrun. *FastSLAM 2.0.* FastSLAM: A scalable method for the simultaneous localization and mapping problem in robotics, pages 63–90, 2007. (Cited in pages 17 and 27.)

[Moreno 2006] Juan C Moreno, Eduardo Rocon de Lima, Andrés F Ruíz, Fernando J Brunetti and José L Pons. *Design and implementation of an inertial measurement unit for control of artificial limbs: application on leg orthoses.* Sensors and Actuators B: Chemical, vol. 118, no. 1-2, pages 333–337, 2006. (Cited in page 125.)

[Mouragnon 2009] Etienne Mouragnon, Maxime Lhuillier, Michel Dhome, Fabien Dekeyser and Patrick Sayd. *Generic and real-time structure from motion using local bundle adjustment.* Image and Vision Computing, vol. 27, no. 8, pages 1178–1193, 2009. (Cited in page 17.)

[Mourikis 2007] Anastasios I Mourikis and Stergios I Roumeliotis. *A multi-state constraint Kalman filter for vision-aided inertial navigation.* In Robotics and automation, 2007 IEEE international conference on, pages 3565–3572. IEEE, 2007. (Cited in page 38.)

[Mourikis 2009] Anastasios I Mourikis, Nikolas Trawny, Stergios I Roumeliotis, Andrew E Johnson, Adnan Ansar and Larry Matthies. *Vision-aided inertial navigation for spacecraft entry, descent, and landing.* IEEE Transactions on Robotics, vol. 25, no. 2, pages 264–280, 2009. (Cited in page 38.)

[Mueller 2011] John Kyle P Mueller, Boyd M Evans, M Nance Ericson, Ethan Farquhar, Randall Lind, Kevin Kelley, Martin Pusch, Timo Von Marcard and Jason M Wilken. *A mobile motion analysis system using inertial sensors for analysis of lower limb prosthetics.* In Future of Instrumentation International Workshop (FIIW), 2011, pages 59–62. IEEE, 2011. (Cited in page 125.)

[Murphy 1998] Robin R Murphy. *Dempster-Shafer theory for sensor fusion in autonomous mobile robots.* IEEE Transactions on Robotics and Automation, vol. 14, no. 2, pages 197–206, 1998. (Cited in page 118.)

[Nakaoka 2007] Shin'ichiro Nakaoka, Shizuko Hattori, Fumio Kanehiro, Shuuji Kajita and Hirohisa Hirukawa. *Constraint-based dynamics simulator for humanoid robots with shock absorbing mechanisms.* In Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on, pages 3641–3647. IEEE, 2007. (Cited in page 53.)

[Ni 2007] Kai Ni, Drew Steedly and Frank Dellaert. *Out-of-core bundle adjustment for large-scale 3d reconstruction.* In Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on, pages 1–8. IEEE, 2007. (Cited in page 17.)

[Ni 2010] Kai Ni and Frank Dellaert. *Multi-level submap based slam using nested dissection.* In Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on, pages 2558–2565. IEEE, 2010. (Cited in page 17.)

[Nilsson 2014a] John-Olof Nilsson, Amit K Gupta and Peter Händel. *Foot-mounted inertial navigation made easy.* In Indoor Positioning and Indoor Navigation (IPIN), 2014 International Conference on, pages 24–29. IEEE, 2014. (Cited in page 41.)

[Nilsson 2014b] John-Olof Nilsson, Isaac Skog and Peter Händel. *Aligning the forces—Eliminating the misalignments in IMU arrays.* IEEE Transactions

on Instrumentation and Measurement, vol. 63, no. 10, pages 2498–2500, 2014. (Cited in page 48.)

[Novak 2015] Domen Novak and Robert Riener. *A survey of sensor fusion methods in wearable robotics*. Robotics and Autonomous Systems, vol. 73, pages 155–170, 2015. (Cited in page 119.)

[Ohno 2003] Kazunori Ohno, Takashi Tsubouchi, Bunji Shigematsu, Shoichi Maeyama and Shin'ichi Yuta. *Outdoor navigation of a mobile robot between buildings based on DGPS and odometry data fusion*. In Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on, volume 2, pages 1978–1984. IEEE, 2003. (Cited in page 3.)

[Ohno 2004] Kazunori Ohno, Takashi Tsubouchi, Bunji Shigematsu and Shin'ichi Yuta. *Differential GPS and odometry-based outdoor navigation of a mobile robot*. Advanced Robotics, vol. 18, no. 6, pages 611–635, 2004. (Cited in page 3.)

[Ojeda 2007] Lauro Ojeda and Johann Borenstein. *Personal dead-reckoning system for GPS-denied environments*. In Safety, Security and Rescue Robotics, 2007. SSRR 2007. IEEE International Workshop on, pages 1–6. IEEE, 2007. (Cited in page 65.)

[Oliensis 1991] J Oliensis and J Inigo Thomas. *Incorporating motion error in multi-frame structure from motion*. In Visual Motion, 1991., Proceedings of the IEEE Workshop on, pages 8–13. IEEE, 1991. (Cited in page 16.)

[Ollero 2015] Aníbal Ollero. *AEROARMS: AErial RObotics System integrating multiple ARMS and advanced manipulation capabilities for inspection and maintenance*. "https://aeroarms-project.eu/", 2015. [Online; accessed 31-Jul-2018]. (Cited in page 120.)

[Olofsson 2016] Björn Olofsson, Jacob Antonsson, Henk G Kortier, Bo Bernhardsson, Anders Robertsson and Rolf Johansson. *Sensor fusion for robotic workspace state estimation*. IEEE/ASME Transactions on Mechatronics, vol. 21, no. 5, pages 2236–2248, 2016. (Cited in page 119.)

[Olson 2006] Edwin Olson, John Leonard and Seth Teller. *Fast iterative alignment of pose graphs with poor initial estimates*. In Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on, pages 2262–2269. IEEE, 2006. (Cited in page 29.)

[Olson 2008] Edwin B Olson. *Robust and efficient robotic mapping*. 2008. (Cited in page 28.)

[Olsson 2016] Fredrik Olsson, Manon Kok, Kjartan Halvorsen and Thomas B Schön. *Accelerometer calibration using sensor fusion with a gyroscope*. In

Statistical Signal Processing Workshop (SSP), 2016 IEEE, pages 1–5. IEEE, 2016. (Cited in page 118.)

[Panahandeh 2012] Ghazaleh Panahandeh, Nasser Mohammadiha, Arne Leijon and Peter Händel. *Chest-mounted inertial measurement unit for pedestrian motion classification using continuous hidden Markov model.* In Instrumentation and Measurement Technology Conference (i2mtc), 2012 IEEE International, pages 991–995. IEEE, 2012. (Cited in page 65.)

[Park 2015] Hae-Won Park, Patrick M Wensing, Sangbae Kim*et al.* *Online planning for autonomous running jumps over obstacles in high-speed quadrupeds.* 2015. (Cited in page 6.)

[Pittelkau 2006] Mark E Pittelkau. *Cascaded and decoupled RIMU calibration filters.* The Journal of the Astronautical Sciences, vol. 54, no. 3-4, pages 449–466, 2006. (Cited in page 48.)

[Pomerleau 2013] François Pomerleau, Francis Colas, Roland Siegwart and Stéphane Magnenat. *Comparing ICP variants on real-world data sets.* Autonomous Robots, vol. 34, no. 3, pages 133–148, 2013. (Cited in page 53.)

[Puyol 2012] Maria Garcia Puyol, Patrick Robertson and Oliver Heirich. *Complexity-reduced FootSLAM for indoor pedestrian navigation.* In Indoor Positioning and Indoor Navigation (IPIN), 2012 International Conference on, pages 1–10. IEEE, 2012. (Cited in page 66.)

[Reina 2007] Giulio Reina, Andres Vargas, Keiji Nagatani and Kazuya Yoshida. *Adaptive kalman filtering for gps-based mobile robot localization.* In Safety, Security and Rescue Robotics, 2007. SSRR 2007. IEEE International Workshop on, pages 1–6. IEEE, 2007. (Cited in page 3.)

[Rotella 2016] Nicholas Rotella, Sean Mason, Stefan Schaal and Ludovic Righetti. *Inertial sensor-based humanoid joint state estimation.* In Robotics and Automation (ICRA), 2016 IEEE International Conference on, pages 1825–1831. IEEE, 2016. (Cited in page 124.)

[Roussillon 2011] Cyril Roussillon, Aurélien Gonzalez, Joan Solà, Jean-Marie Codol, Nicolas Mansard, Simon Lacroix and Michel Devy. *RT-SLAM: a generic and real-time visual SLAM implementation.* In International Conference on Computer Vision Systems, pages 31–40. Springer, 2011. (Cited in pages 17, 41, and 118.)

[Ruiz 2012] Antonio Ramón Jiménez Ruiz, Fernando Seco Granja, José Carlos Prieto Honorato and Jorge I Guevara Rosas. *Accurate pedestrian indoor navigation by tightly coupling foot-mounted IMU and RFID measurements.* IEEE Transactions on Instrumentation and Measurement, vol. 61, no. 1, pages 178–189, 2012. (Cited in page 66.)

[Santamaria-Navarro 2017a] Angel Santamaria-Navarro, Patrick Grosch, Vincenzo Lippiello, Joan Solà and Juan Andrade-Cetto. *Uncalibrated visual servo for unmanned aerial manipulation.* IEEE/ASME Transactions on Mechatronics, vol. 22, no. 4, pages 1610–1621, 2017. (Cited in page 120.)

[Santamaria-Navarro 2017b] Angel Santamaria-Navarro, Giuseppe Loianno, Joan Solà, Vijay Kumar and Juan Andrade-Cetto. *Autonomous navigation of micro aerial vehicles using high-rate and low-cost sensors.* Autonomous Robots, pages 1–18, 2017. (Cited in page 120.)

[Santoso 2017] Fendy Santoso, Matthew A Garratt and Sreenatha G Anavatti. *Visual–inertial navigation systems for aerial robotics: Sensor fusion and technology.* IEEE Transactions on Automation Science and Engineering, vol. 14, no. 1, pages 260–275, 2017. (Cited in page 119.)

[Schauer 2016] T Schauer, T Seel, ND Bunt, P Müller and JC Moreno. *Realtime EMG analysis for transcutaneous electrical stimulation assisted gait training in stroke patients.* IFAC-PapersOnLine, vol. 49, no. 32, pages 183–187, 2016. (Cited in pages vii and 65.)

[Schulz 2001] Dirk Schulz, Wolfram Burgard, Dieter Fox and Armin B Cremers. *Tracking multiple moving targets with a mobile robot using particle filters and statistical data association.* In Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on, volume 2, pages 1665–1670. IEEE, 2001. (Cited in page 27.)

[Seel 2014] Thomas Seel, Jörg Raisch and Thomas Schauer. *IMU-based joint angle measurement for gait analysis.* Sensors, vol. 14, no. 4, pages 6891–6909, 2014. (Cited in page 125.)

[Seel 2016a] Thomas Seel. *Learning control and inertial realtime gait analysis in biomedical applications.* 2016. (Cited in page 125.)

[Seel 2016b] Thomas Seel, Cordula Werner, Jörg Raisch and Thomas Schauer. *Iterative learning control of a drop foot neuroprosthesis—Generating physiological foot motion in paretic gait by automatic feedback control.* Control Engineering Practice, vol. 48, pages 87–97, 2016. (Cited in page 125.)

[Shum 1999] Heung-Yeung Shum, Qifa Ke and Zhengyou Zhang. *Efficient bundle adjustment with virtual key frames: A hierarchical approach to multi-frame structure from motion.* In Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on., volume 2, pages 538–543. IEEE, 1999. (Cited in page 17.)

[Siegwart 2011] Roland Siegwart, Illah Reza Nourbakhsh and Davide Scaramuzza. Introduction to autonomous mobile robots. MIT press, 2011. (Cited in pages vii, 4, and 111.)

[Sim 2005] Robert Sim, Pantelis Elinas, Matt Griffin, James J Little*et al. Vision-based SLAM using the Rao-Blackwellised particle filter.* In IJCAI Workshop on Reasoning with Uncertainty in Robotics, volume 14, pages 9–16, 2005. (Cited in page 12.)

[Skog 2006] Isaac Skog and Peter Händel. *Calibration of a MEMS inertial measurement unit.* In XVII IMEKO world congress, pages 1–6, 2006. (Cited in page 47.)

[Skog 2012] Isaac Skog, John-Olof Nilsson, Dave Zachariah and Peter Händel. *Fusing the information from two navigation systems using an upper bound on their maximum spatial separation.* In Indoor Positioning and Indoor Navigation (IPIN), 2012 International Conference on, pages 1–5. IEEE, 2012. (Cited in page 66.)

[Šlajpah 2014] S Šlajpah, Roman Kamnik and Marko Munih. *Kinematics based sensory fusion for wearable motion assessment in human walking.* Computer methods and programs in biomedicine, vol. 116, no. 2, pages 131–144, 2014. (Cited in page 125.)

[Soatto 1993] Stefano Soatto, Pietro Perona, Ruggero Frezza and Giorgio Picci. *Recursive motion and structure estimation with complete error characterization.* In Computer Vision and Pattern Recognition, 1993. Proceedings CVPR'93., 1993 IEEE Computer Society Conference on, pages 428–433. IEEE, 1993. (Cited in page 16.)

[Solà Ortega 2016] Joan Solà Ortega. *Quaternion kinematics for the error-state KF.* 2016. (Cited in page 131.)

[Sola 2007] Joan Sola. *Towards visual localization, mapping and moving objects tracking by a mobile robot: a geometric and probabilistic approach.* PhD thesis, 2007. (Cited in pages 8 and 25.)

[Sola 2016] Joan Sola. *Course on SLAM.* Institut de Robotica i Informatica Industrial (IRI), 2016. (Cited in pages 8, 11, 34, and 37.)

[Solà 2018] Joan Solà, Jeremie Deray and Dinesh Atchuthan. *A micro Lie theory for state estimation in robotics.* arXiv preprint arXiv:1812.01537, 2018. (Cited in pages 7, 8, and 113.)

[Spanos 2005] Demetri P Spanos, Reza Olfati-Saber and Richard M Murray. *Distributed sensor fusion using dynamic consensus.* In IFAC World Congress. Citeseer, 2005. (Cited in page 118.)

[Spetsakis 1991] Minas Spetsakis and John Yiannis Aloimonos. *A multi-frame approach to visual motion perception.* International Journal of Computer Vision, vol. 6, no. 3, pages 245–255, 1991. (Cited in page 16.)

[Stachniss 2013] Cyrill Stachniss. *Robot Mapping*. http://ais.informatik.uni-freiburg.de/teaching/ws13/mapping/, 2013. [Online; accessed 17-May-2018]. (Cited in page 11.)

[Stasse 2006] Olivier Stasse, Andrew J Davison, Ramzi Sellaouti and Kazuhito Yokoi. *Real-time 3d slam for humanoid robot considering pattern generator information*. In Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on, pages 348–355. IEEE, 2006. (Cited in page 53.)

[Stasse 2017] Olivier Stasse, Thomas Flayols, Rohan Budhiraja, Kevin Giraud-Esclasse, Justin Carpentier, Joseph Mirabel, Andrea Del Prete, Philippe Souères, Nicolas Mansard, Florent Lamiraux*et al. TALOS: A new humanoid research platform targeted for industrial applications*. In Humanoid Robotics (Humanoids), 2017 IEEE-RAS 17th International Conference on, pages 689–695. IEEE, 2017. (Cited in page 117.)

[Steedly 2001] Drew Steedly and Irfan Essa. *Propagation of innovative information in non-linear least-squares structure from motion*. In Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on, volume 2, pages 223–229. IEEE, 2001. (Cited in page 17.)

[Steins 2014] Dax Steins, Ian Sheret, Helen Dawes, Patrick Esser and Johnny Collett. *A smart device inertial-sensing method for gait analysis*. Journal of biomechanics, vol. 47, no. 15, pages 3780–3785, 2014. (Cited in page 125.)

[Stone 2013] James V Stone. Bayes' rule: A tutorial introduction to bayesian analysis. Sebtel Press, 2013. (Cited in page 129.)

[Strasdat 2010] Hauke Strasdat, JMM Montiel and Andrew J Davison. *Real-time monocular SLAM: Why filter?* In Robotics and Automation (ICRA), 2010 IEEE International Conference on, pages 2657–2664. IEEE, 2010. (Cited in pages 11, 17, and 18.)

[Strasdat 2012] Hauke Strasdat, José MM Montiel and Andrew J Davison. *Visual SLAM: why filter?* Image and Vision Computing, vol. 30, no. 2, pages 65–77, 2012. (Cited in pages 17 and 38.)

[Szeliski 1994] Richard Szeliski and Sing Bing Kang. *Recovering 3D shape and motion from image streams using nonlinear least squares*. Journal of Visual Communication and Image Representation, vol. 5, no. 1, pages 10–28, 1994. (Cited in page 17.)

[Tedaldi 2013] David Tedaldi. *IMU calibration without mechanical equipment.(Calibrazione di IMU svincolata da apparati meccanici)*. 2013. (Cited in pages vii, 42, and 44.)

[Thrun 2005] Sebastian Thrun, Wolfram Burgard and Dieter Fox. Probabilistic robotics. MIT press, 2005. (Cited in pages 12, 15, and 21.)

[Thrun 2006] Sebastian Thrun and Michael Montemerlo. *The graph SLAM algorithm with applications to large-scale mapping of urban structures.* The International Journal of Robotics Research, vol. 25, no. 5-6, pages 403–429, 2006. (Cited in pages 29 and 55.)

[Titterton 2004] David Titterton, John L Weston and John Weston. Strapdown inertial navigation technology, volume 17. IET, 2004. (Cited in page 43.)

[Triggs 1999] Bill Triggs, Philip F McLauchlan, Richard I Hartley and Andrew W Fitzgibbon. *Bundle adjustment—a modern synthesis.* In International workshop on vision algorithms, pages 298–372. Springer, 1999. (Cited in page 16.)

[Tsakiris 1997] Dimitris P Tsakiris, Patrick Rives and Claude Samson. *Applying visual servoing techniques to control nonholonomic mobile robots.* In International Conference on Intelligent Robots and Systems, 1997. (Cited in page 4.)

[Vallvé 2015] Joan Vallvé and Juan Andrade-Cetto. *Active pose SLAM with RRT.* In Robotics and Automation (ICRA), 2015 IEEE International Conference on, pages 2167–2173. IEEE, 2015. (Cited in page 120.)

[Vallvé 2017] Joan Vallvé, Joan Solà and Juan Andrade-Cetto. *Factor descent optimization for sparsification in graph SLAM.* In Mobile Robots (ECMR), 2017 European Conference on, pages 1–6. IEEE, 2017. (Cited in page 120.)

[Vallvé 2018] Joan Vallvé, Joan Solà and Juan Andrade-Cetto. *Graph SLAM sparsification with populated topologies using factor descent optimization.* IEEE Robotics and Automation Letters, vol. 3, no. 2, pages 1322–1329, 2018. (Cited in page 120.)

[Victorino 2003] Alessandro Corrêa Victorino, Patrick Rives and Jean-Jacques Borrelly. *Safe navigation for indoor mobile robots. Part I: a sensor-based navigation framework.* The International Journal of Robotics Research, vol. 22, no. 12, pages 1005–1118, 2003. (Cited in page 116.)

[Wagstaff 2017] Brandon Wagstaff, Valentin Peretroukhin and Jonathan Kelly. *Improving Foot-Mounted Inertial Navigation Through Real-Time Motion Classification.* arXiv preprint arXiv:1707.01152, 2017. (Cited in page 65.)

[Wan 2000] Eric A Wan and Rudolph Van Der Merwe. *The unscented Kalman filter for nonlinear estimation.* In Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000, pages 153–158. Ieee, 2000. (Cited in page 24.)

[Weng 1993] Juyang Weng, Narendra Ahuja and Thomas S Huang. *Optimal motion and structure estimation.* IEEE Transactions on pattern analysis and machine intelligence, vol. 15, no. 9, pages 864–884, 1993. (Cited in page 16.)

[Young 1945] Olive G Young. *A study of kinesthesis in relation to selected movements*. Research Quarterly. American Association for Health, Physical Education and Recreation, vol. 16, no. 4, pages 277–287, 1945. (Cited in page 1.)

[Zhao 2016] Huihua Zhao, Jonathan Horn, Jacob Reher, Victor Paredes and Aaron D Ames. *Multicontact locomotion on transfemoral prostheses via hybrid system models and optimization-based control*. IEEE Transactions on Automation Science and Engineering, vol. 13, no. 2, pages 502–513, 2016. (Cited in page 125.)

Abstract:
Estimation in robotics is an important subject affected by trade-offs between some major critera from which we can cite the computation time and the accuracy. The importance of these two criteria are application-dependent. If the computation time is not important for off-line methods, it becomes critical when the application has to run on real-time. Similarly, accuracy requirements are dependant on the applications. EKF estimators are widely used to satisfy real-time constraints while achieving acceptable accuracies. One sensor widely used in trajectory estimation problems remains the inertial measurement units (IMUs) providing data at a high rate. The main contribution of this thesis is a clear presentation of the preintegration theory yielding in a better use IMUs. We apply this method for estimation problems in both pedestrian and humanoid robots navigation to show that real-time estimation using a low-cost IMU is possible with smoothing methods while formulating the problems with a factor graph. We also investigate the calibration of the IMUs as it is a critical part of those sensors. All the development made during this thesis was thought with a visual-inertial SLAM background as a mid-term perspective. Firthermore, this work tries to rise another question when it comes to legged robots. In opposition to their usual architecture, could we use multiple low-cost IMUs on the robot to get valuable information about the motion being executed?
Keywords:
Factor Graph, low-cost IMU, sensor calibration, real-time state estimation, SLAM, pedestrian navigation, IMU preintegration on manifolds, robot