UTS UNIVERSITY OF TECHNOLOGY SYDNEY

Gaussian Process Preintegration for Inertial-Aided Navigation Systems

by Cedric Le Gentil

Thesis submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy

under the supervision of A/Prof. Teresa Vidal-Calleja and A/Prof. Shoudong Huang

University of Technology Sydney
Faculty of Engineering and IT

February 2021

Certificate of Original Authorship

I, Cedric Le Gentil declare that this thesis, is submitted in fulfilment of the requirements for the award of degree of Doctor of Philosophy, in the Faculty of Engineering and IT (FEIT) at the University of Technology Sydney (UTS).

This thesis is wholly my own work unless otherwise reference or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

This document has not been submitted for qualifications at any other academic institution.

This research is supported by the Australian Government Research Training Program.

Signature: Production Note: Signature removed prior to publication.

Date: 24/02/2021

Gaussian Process Preintegration for Inertial-Aided Navigation Systems

by

Cedric Le Gentil

A thesis submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy

Abstract

To perform any degree of autonomy, a system needs to localise itself, generally requiring knowledge about its environment. While satellite technologies, like GPS or Galileo, allow individuals to navigate throughout the world, the level of accuracy of such systems, and the necessity to have a direct view of the sky, do not match the precision and robustness requirements needed to deploy robots in the real world. To overcome these limitations, roboticists developed localisation and mapping algorithms traditionally based on camera images or radar/LiDAR data. Across the last two decades, Inertial Measurement Units (IMUs) became ubiquitous. Thus, LiDAR-inertial and visual-inertial pose estimation algorithms represent now the majority of the state estimation literature.

Preintegration became a standard method to aggregate inertial measurement units (IMUs) readings into pseudo-measurements for navigation systems. This thesis presents a novel preintegration theory that leverages data-driven continuous representations of the inertial data to perform analytical inference of the signal integrals. The proposed method probabilistically infers the pseudo-measurements, called *Gaussian Preintegrated Measurements* (GPMs), over any time interval, using Gaussian Process (GP) regression to model the IMU measurements and leveraging the application of linear operators to the GP covariance kernels. Thus, the GPMs do not rely on any explicit motion-model.

This thesis presents two inertial-aided systems that leverage the GPMs in offline batch-optimisation algorithms. The first one is a framework called IN2LAAMA for *INertial Lidar Localisation Autocalibration And MApping*. The proposed method addresses the issue of *motion distortion* present in most of today's LiDARs' data thoroughly by using GPMs for each of the LiDAR points.

The second GPM application is an event-based visual-inertial odometry method that uses lines to represent the environment. Event-cameras generate highly asynchronous streams of events that are individually triggered by each of the camera pixels upon illumination

changes. Our framework, called IDOL for *IMU-DVS Odometry using Lines*, estimates the system's pose as well as the position of 3D lines in the environment by considering the camera events in the framework's cost function individually (no aggregation in image-like data). The GPMs allow for the continuous characterisation of the system's trajectory, therefore accommodating the asynchronous nature of event-camera data.

Extensive benchmarking of the GPMs is performed on simulated data. The performance of IN2LAAMA is thoroughly demonstrated throughout simulated and real-world experiments, both indoor and outdoor. Evaluations on public datasets show that IDOL performs at the same order of magnitude as current frame-based state-of-the-art visual-inertial odometry frameworks.

"Vous savez, moi je ne crois pas qu'il y ait de bonne ou de mauvaise situation. Moi, si je devais résumer ma vie aujourd'hui avec vous, je dirais que c'est d'abord des rencontres."[1]

Édouard Beaer

[1] "Well, you see... I don't believe that there are good or bad situations. If I had to summarise my life, here, with you, I would say that it is all about encounters."

Acknowledgements

The first person I want to thank is someone that I forgot the name of. When people ask me how I started my PhD, I refer to him as the "Spanish dude". During a smoko[2] on a building site in April 2016, I told him that I was thinking to quit my labourer job and attempt to volunteer in random software companies to try to "get back in my field". To this, he pronounced words that profoundly changed my life:

"Why don't you go see a university for that?".

I cannot thank Shoudong Huang enough for answering my email; even if I suspect the terms *"I am ready to work on a volunteer basis"* had a non-negligible role in his decision to invite me to visit the lab and introduce me to Teresa Vidal-Calleja. After four and something years, it is always a pleasure to work with Teresa. Her passion for her job is an inexhaustible source of motivation for everyone she collaborates with. She believed in me and introduced me to the academic world. She gave me access to unique opportunities that made me grow not only as a researcher, but also as a person. Both for their technical expertise and their human qualities, Teresa and Shoudong have been wonderful supervisors that allowed me to make the best out of my PhD experience.

I want to thank every member of the Centre for Autonomous Systems that I had the chance to interact with. I especially express my gratitude to Raphaël Falque, Lakshitha Dantanarayana, and Buddhi Wijerathna for many answers to many questions, for the laughs and the road trips, more broadly for their friendship inside and outside the lab.

I want to thank Delphine and Flavien Dyiévre-Hamon who have demonstrated invaluable support when I needed it most. My PhD experience would not have been the same without rock climbing, its beautiful community, and the UTS Outdoor Adventure Club. Among the people I actually trust my life with on a weekly basis, I want to thank James Millar, who is always keen for crazy adventures, Thibaut Géry and Ronny Onggo for teaching me so much, and Marta Khomyn for changing my vision of the world and life itself. Marta deserves even more gratitude for her moral support and helping to proofread the very first version of this thesis.

Last but not least, I want to thank my parents, Valérie and Christophe Le Gentil, for their love despite the kilometres. During my whole existence, they offered me the best, even when life was not the easiest for them. The freedom and trust they gave me have been a blessing. Thank you for understanding and accepting the different choices I made throughout my life.

[2] Australian slang term designating a short break during work. Mostly used in the construction industry.

Contents

Acronyms & Abbreviations

1D	One-Dimensional
2D	Two-Dimensional
3D	Three-Dimensional
CAS	Centre for Autonomous Systems
CPU	Central Processing Unit
DoF	Degree-of-Freedom
DVS	Dynamic Vision Sensor
EKF	Extended Kalman filter
FoV	Field-of-View
GNSS	Global Navigation Satellite System
GP	Gaussian Process
GPM	Gaussian Preintegrated Measurement
GPS	Global Position System
GPU	Graphic Processing Unit
HDR	High Dynamic Range
ICP	Iterative Closest Point
IDOL	IMU-DVS Odometry using Lines

IMU	Inertial Measurement Unit
IN2LAAMA	INertial Lidar Localisation Autocalibration And MApping
KF	Kalman Filter
LiDAR	Light Detection And Ranging Sensor
LPM	Linear Preintegrated Measurement
MAP	Maximum A Posteriori
MLE	Maximum Likelihood Estimation
NDT	Normal Distribution Transform
PM	Standard Preintegrated Measurement
RRBT	Rapidly Exploring Random Belief Trees
RGB	Red-Green-Blue
RGBD	Red-Green-Blue-Depth
RMS	Root Mean Squared
RMSE	Root Mean Squared Error
SE(3)	Special Euclidean group in three dimensions
SLAM	Simultaneous Localisation And Mapping
$\mathfrak{so}$**(3)**	Lie algebra of special orthonormal group in three dimensions
SO(3)	Special orthonormal group in three dimensions
UPM	Upsampled-Preintegrated-Measurement
UTS	University of Technology, Sydney
VI	Visual-Inertial
VIO	Visual-Inertial Odometry
VO	Visual Odometry

Chapter 1

Introduction

1.1 Research topic

To navigate through its environment, a system needs to know where it is in space. Performing localisation corresponds to estimating the system's pose (position and orientation) given data collected with one or several sensors. Systems relying on multiple sensors that provide different type of data are called *multi-modal*. Any sensor can be classified into two categories: *proprioceptive*, that provides information about the system's state, or *exteroceptive*, that measures physical values about the system's surroundings. Inertial Measurement Units (IMUs), constituted of gyroscopes and accelerometers, are proprioceptive sensors that measure a system's angular velocities and proper accelerations. A multi-modal system that comprises at least one IMU is called an *inertial-aided* system. Thanks to the development of consumer electronics, IMUs are now lightweight and affordable. Consequently, they became ubiquitous and are used in a large portion of the pose estimation algorithms present in the literature.

To produce pose information, an IMU's data need to be processed using the laws of classical mechanics. The traditional way is to directly compute the integral of the inertial measurements from known initial conditions. In the context of state estimation, these initial conditions are not accurately known and change during the estimation process. Recently, the concept of preintegration was introduced in [1] allowing for the pre-processing

of the inertial signals in a way that does not rely on the knowledge of the initial conditions. Since, the efficiency of inertial-aided navigation systems has greatly improved.

Gaussian Processes (GPs) are widely used for data-driven signal modelling. As GPs are stochastic processes that define signals as multivariate Gaussian distributions, GP regression is a non-parametric interpolation method. Given noisy discrete observations of a signal's value, it allows probabilistic and continuous inference of the signal's value and its variance.

This thesis focuses on using GPs as continuous representations of IMU data to perform accurate preintegration for inertial-aided navigation systems.

1.2 Scope

1.2.1 State estimation

State estimation is the science of evaluating a system's state given sensory information. In some scenarios, the nature of the observed system is subject to physical constraints that make some state configurations unrealistic. Let us consider the problem of pose estimation of a car. By design and in normal-use conditions, a car cannot drive sideways. This information can be incorporated in the pose estimation process under the form of a *motion model* that constrains the trajectory to be tangential to the rear-wheel axle. While motion models can originate from physical constraints, they are also often used for their ability to reduce the computational complexity of an estimation problem. Thus, instead of estimating the system's pose at the time of each sensor measurement, the pose can be estimated at a lower frequency along with the model parameters. *Constant-velocity* and *constant-acceleration* are two popular parametric motion models in the literature.

Motion models necessarily rely on simplifying assumptions about the real world. When it comes to computing the integrals of acceleration data numerically from an IMU, the choice of the integration technique embeds some kind of motion model. For example, using the rectangle rule corresponds to assuming the acceleration is constant in between each measurement; and using the trapezoidal rule corresponds to constant jerk. It is equivalent

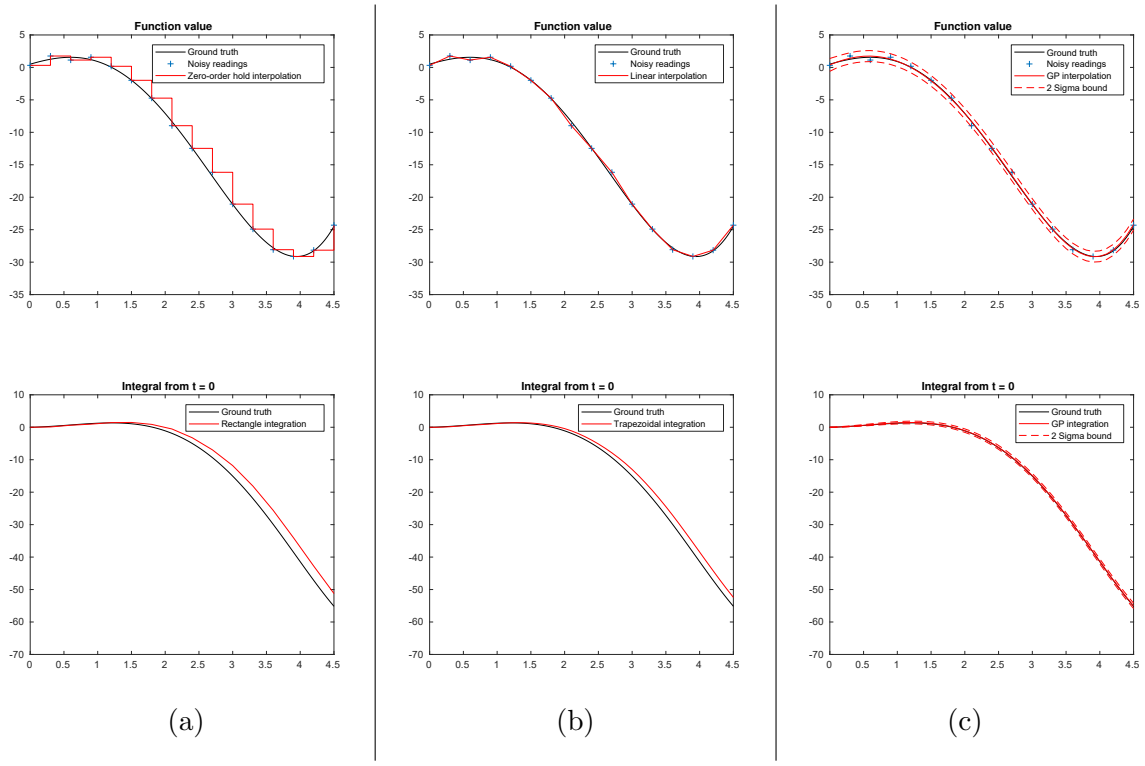


FIGURE 1.1: Illustration of the integration of noisy measurements with different methods. (a) uses the rectangle rule for numerical integration, whereas (b) uses the trapezoidal rule. If the signal represents accelerometer data, (a) and (b) correspond to constant-acceleration and constant-jerk motion models, respectively. (c) integrates analytically and probabilistically the input signal modelled with a GP. This method does not enforce any motion model.

to performing analytical integration on signals that have been interpolated with a zero-order hold and affine functions respectively. GP regression is a data-driven non-parametric interpolation method that uses a kernel function to represent the signal's covariance. Unlike the methods mentioned above, it does not embed any explicit motion model. The analytical inference of the signal integral is possible via the application of linear operators to the covariance kernel function. Fig. 1.1 illustrates the different integration methods mentioned in this paragraph and shows that GP-based integration is far more accurate than the other-two model-based methods. Previous preintegration works [1] and [2] rely on the assumption of constant acceleration between IMU measurements (Fig. 1.1(a)).

FIGURE 1.2: Sensors with built-in IMUs. From left to right: a Velodyne LiDAR HDL-32 (image source: velodynelidar.com), an RGBD Intel Realsense Camera D435i (image source: intelrealsense.com), and an Inivation event-camera DAVIS346 (image source: rpg.ifi.uzh.ch).

1.2.2 Localisation and mapping

In many pose estimation scenarios, the environment is not known beforehand. Sometimes this absence of knowledge is actually the very reason why a sensory system is used. In robotics, the word *map* refers to a digital model of the real world. In a society that relies more and more on automation and robotics, the need for Three-Dimensional (3D) maps is increasing at a fast pace, but not solely for autonomous systems navigation purposes. A growing number of fields are gaining interest in dense and accurate representations, especially for monitoring or inspection operations, and augmented reality content. The term Simultaneous Localisation And Mapping (SLAM) encapsulates the methods that perform both mapping and pose estimation at the same time.

Originally SLAM techniques were relying solely on one modality; generally camera vision, Two-Dimensional (2D) Light Detection And Ranging Sensors (LiDARs), or radars [3]. The democratisation of consumer electronics led to the development of more accurate and affordable sensors along with access to greater computational power. Accordingly, SLAM and navigation systems started to leverage data from multiple sensors in multi-modal sensor-fusion algorithms. As mentioned in Section 1.1, IMUs became ubiquitous. They are the cornerstones of numerous frameworks among the constellation of methods present in the literature [4]. While high-end IMUs are generally sold individually and rigidly mounted to a system along with other sensors, it is now common to find cameras or LiDARs with built-in IMUs (Fig. 1.2). Nowadays, lidar-inertial, visual-inertial, and lidar-visual-inertial systems represent the majority of every-use autonomous systems.

1.2.3 Extrinsic calibration

To perform best, a multi-sensor system needs to include an accurate extrinsic calibration between the different sensory devices. The term extrinsic calibration can encapsulate several concepts. It mostly refers to the estimation of the geometric transformations that spatially separate the sensors, but it can also involve the characterisation of the temporal asynchronism present in multi-sensory data.

1.2.4 Lidars and motion distortion

LiDARs (2D and 3D) are widely used in robotics. The information provided by a rotating LiDAR can be interpreted in two ways. One can see LiDAR readings as being full 360-degree scans of the scene or as an extremely fast succession of range measurements at different angles of azimuth and elevation. In the first case, it is commonly assumed that all the points collected are expressed in a single reference frame. In the second case, each of the points is expressed in a temporally unique reference frame. When the system is static, both ways give the same outcome. In other words, all the point reference frames spatially coincide with the scan reference frame. This is not the case when the system moves. By assuming the points are directly expressed in a common scan reference frame, the resulting point cloud is distorted. This phenomenon is called *motion distortion* and is present in every sensor that acquires sequential exteroceptive information of a physical area. Rolling-shutter cameras are another example of such sensors. For navigation and SLAM, an accurate continuous characterisation of the system's motion is needed.

1.2.5 Event-cameras

Event-cameras, or Dynamic Vision Sensors (DVS's), are novel sensors that differ from traditional cameras (or frame-cameras) in the nature of the data they produce (Fig. 1.4). When the pixels of a frame-camera are triggered at a fixed frequency to acquire an ensemble of light-intensity readings (called image), the pixels of an event-camera contain a circuitry that independently triggers *events* upon change of intensity. Consequently, event-cameras produce high-bandwidth asynchronous streams of events when the system moves or when

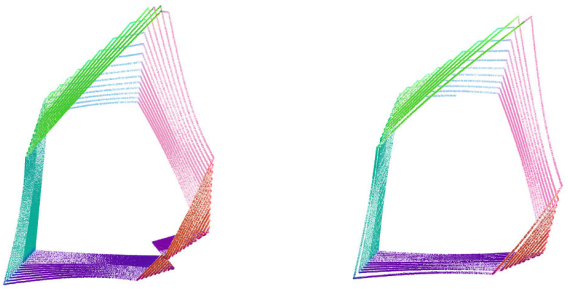


FIGURE 1.3: Example of LiDAR data with and without motion distortion. Left: point cloud collected while the system is moving. The points' mismatch at the bottom shows the evident motion distortion present in the scan. Right: scan of the same location without motion distortion. Both point clouds are coloured with the normals computed from the 100-closest neighbours.

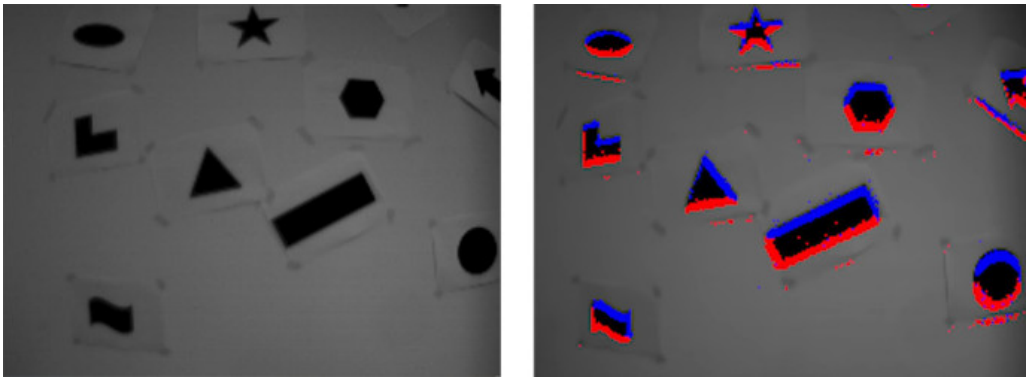


FIGURE 1.4: Event-camera data example. Left: grayscale image of the scene. Right: temporal accumulation of events collected with a Inilabs DAVIS240c overlaid on the corresponding grayscale image. The event-camera moves in a static environment. The events' colours, blue and red, correspond to the direction of the intensity change, positive and negative, respectively (image source: rpg.ifi.uzh.ch).

dynamic elements are present in the environment. These challenging, unconventional data have a huge potential for navigation under very fast motions and in scenes with High Dynamic Range (HDR) of illumination.

1.2.6 Motivation

This thesis proposes a preintegration framework that allows accurate and asynchronous inference of inertial data for any given time-stamp. While this method has a wide range of applications for inertial-aided navigation, it is especially suited for use with highly asynchronous sensors like LiDARs and event-cameras. We apply our novel technique in an

autocalibrating lidar-inertial localisation and mapping framework, and in an event-based visual-inertial odometry method.

1.3 Objectives and contributions

The objectives of the work undertaken in this thesis are:

1. Derivation of a new conceptual method for continuous and probabilistic preintegration that does not rely on any motion model.
2. Design and implementation of a novel LiDAR-IMU autocalibration, localisation and mapping framework that accurately addresses the issue of motion distortion based on the proposed preintegration method.
3. Demonstration of the effectiveness and versatility of the proposed preintegration method as part of a novel event-based visual-inertial odometry framework that does not accumulate events into traditional-like images.

For the first objective, we derive a preintegration method that leverages GPs to represent the inertial data provided by a 6-Degree-of-Freedom (DoF) IMU. With the help of linear operators applied to the GP covariance kernels, the integration of the inertial signals is performed analytically (as illustrated in Fig. 1.1). The resulting pseudo-measurements are denoted Gaussian Preintegrated Measurements (GPMs). The characteristics of the GPs make the Gaussian preintegration able to probabilistically infer GPMs at any time-stamp without relying on any motion model. The data-driven nature of GP regression and the possibility to learn the kernels' hyperparameters from the inertial data make the GPMs suitable for virtually any type of motion.

Generally, the accelerometer and gyroscope biases are not accurately known at the time of integration. A postintegration correction mechanism is derived to allow estimation of the IMU biases when GPMs are employed in multi-modal estimation frameworks. This mechanism relies on a first-order Taylor expansion. To further the versatility of the GPMs, we also derived the correction scheme to account for unknown time-shift between the

sensors of the system. Formally, the contributions of our work on Gaussian preintegration are as follows:

- A versatile GP-based preintegration method that performs analytical integration of inertial signals over any time interval. The absence of a motion model and the continuous representation of the inertial data makes the GPM especially suited for sensor fusion between IMUs and asynchronous sensors.
- A postintegration correction mechanism to allow the seamless integration of the GPMs into inertial-aided navigation systems.

Regarding the second objective, most of the state-of-the-art LiDAR-IMU SLAM techniques do not propose ways to deal thoroughly with motion distortion and asynchronism between sensors. Often, a simple motion model is employed, and expensive hardware synchronisation is used. In this thesis, an framework for LiDAR-IMU localisation, mapping and autocalibration is proposed to demonstrate the use of GPMs for continuous motion characterisation. The method relies on the registration of edge and plane features in consecutive LiDAR frames. The feature extraction is performed in the front-end part of the framework, while the system trajectory (thus, the motion distortion correction) is estimated in the back-end part through an offline discrete-state batch optimisation. Because LiDAR scans are originally distorted due to the system's motion, the associated features might not accurately capture the true nature of the observed surface. Thus, our approach manages tight interactions between front-end and back-end to address the issue of motion distortion also at the front-end level. The design of the proposed framework, called INertial Lidar Localisation Autocalibration And MApping (IN2LAAMA), leads to the following contributions:

- A probabilistic localisation and mapping LiDAR-IMU framework (IN2LAAMA) that rigorously addresses the issue of motion distortion by using GPMs for each of the collected LiDAR points.
- An target-less extrinsic calibration procedure that is performed alongside localisation and mapping in IN2LAAMA.

- The tight interaction between front-end and back-end for accurate motion-distortion correction.
- A novel approach for edge and plane feature extraction in sparse LiDAR data. The proposed technique has the advantage to compute feature scores that are consistent with the observed surface (i.e., robust to different viewpoint).
- The use of LiDAR scans that sweep more than 360 °, associated with a back-and-forth data association between consecutive LiDAR frames, to allow for robust motion distortion correction.
- The integration of a simple loop-closure detection mechanism based on the proximity of pose estimates to correct the inherent drift of frame-to-frame estimation techniques. The proposed mechanism accounts for the collection pattern of the LiDAR.
- The thorough evaluation of the proposed framework through simulated and real-world experiments, both indoor and outdoor.

The third objective is to further demonstrate the abilities and versatility of the GPMs. We propose an event-based visual-inertial odometry framework that uses 3D lines to represent the environment. The batch optimisation formulation considers the asynchronous events individually without resorting to event-accumulation into traditional image-like data. The event data stream can be represented into a 3D spatio-temporal space where 3D-geometry operations can be performed. In the method's front-end, events that originate from lines in the environment are extracted by clustering together events based on the events' normal vectors in the spatio-temporal space. At the back-end level, in the cost function of the batch optimisation, the events belonging to line clusters individually lead to point-to-line distance residuals leveraging the associated GPMs. As the line clustering does not provide information about the actual position of the line extremities, we introduce a mechanism to estimate the position of the line ends in space automatically. The framework is called IMU-DVS Odometry using Lines (IDOL), and results in the following contributions:

- An optimisation-based odometry framework (IDOL) for event-based visual-inertial navigation. The proposed method presents a novel paradigm by accounting for the

events individually in the cost function without resorting to event aggregation into image-like data.

- Novel residual types to automatically estimate line endpoints through a mechanism of attraction / repulsion.
- The evaluation on public datasets and preliminary results showing the potential of the proposed method for event-based Visual-Inertial Odometry (VIO).

1.4 Thesis outline

This thesis consists of six chapters with three of them representing the technical contribution of our research. Additionally, appendices provide some supplementary derivations and algorithms used to support the technical chapters.

Chapter 2 presents the literature review. The work in this thesis is related to multiple robotics subfields. Each of these topics is treated separately in the different sections of that chapter. First, we discuss the existing work related to the concept of preintegration. Then, we turn to multiple continuous state representations present in the literature. We also review different LiDAR-based and event-based pose estimation frameworks, before presenting the last section on extrinsic calibration.

Chapter 3 first introduces the background knowledge on GP regression, linear operators, and IMU preintegration. Then we derive our new conceptual method so called Gaussian preintegration. The proposed pipeline first infers the system's orientation based on the gyroscope measurements. The newly obtained rotations are used to project the accelerometer readings into a common frame. Finally, the projected acceleration information is analytically integrated into velocity and position pseudo-measurements. Note that the exact process differs depending on the nature of the rotational part of the system's motion. This chapter also contains thorough experiments showing the accuracy gain provided by the use of GP models.

Chapter 4 proposes a LiDAR-IMU framework for localisation, mapping, and extrinsic calibration. Following existing estimation methods built upon exteroceptive data, our

algorithm is split into a front-end part that pre-processes the sensor measurements, and a back-end part that conducts on-manifold non-linear least-square optimisations to estimate the system's state. This separation is visible in the organisation of this chapter. Extensive experiments have been conducted to compare the proposed method against the current state-of-the-art.

Chapter 5 introduces a novel approach for event-based visual-inertial odometry. The key innovation is to use the camera events individually in the optimisation cost function. Consequently, our work drastically differs from traditional vision-based state estimation methods. It proposes a new paradigm in the way to deal with event-camera data. This chapter also contains back-end and front-end sections to detail the different elements of the proposed technique. Due to the lack of open-source event-based visual-inertial odometry frameworks, in this chapter, we compare our work with the current state-of-the-art frame-based visual-inertial odometry methods on public datasets.

Chapter 6 is split into two sections. The first discusses the conclusion of our research, while the second provides the reader with elements of future work. This last section also discusses concepts and ideas that are currently being explored in our research centre: the Centre for Autonomous Systems (CAS).

1.5 List of publications

This section presents the list of publications that are relevant to this thesis. The symbol * indicates the shared first-authorship of a publication.

1.5.1 Core contributions

- C. Le Gentil, T. Vidal-Calleja, S. Huang, "IN2LAAMA: INertial Lidar Localisation Autocalibration And Mapping", IEEE Transactions on Robotics, 2021
- C. Le Gentil*, F. Tschopp*, I. Alzugaray, T. Vidal-Calleja, R. Siegwart, J. Nieto, "IDOL: A framework for IMU-DVS Odometry using Lines", IEEE/RSJ International Conference on Intelligent Robots and Systems, 2020

- C. Le Gentil, T. Vidal-Calleja, S. Huang, "Gaussian Process Preintegration for Inertial-Aided State Estimation", IEEE Robotics and Automation Letters, vol. 5, no. 2, pp. 2108–2114, 2020
- C. Le Gentil, T. Vidal-Calleja, S. Huang, "IN2LAMA: INertial Lidar Localisation And Mapping", IEEE International Conference on Robotics and Automation, 2019
- C. Le Gentil, T. Vidal-Calleja, S. Huang, "3D Lidar-IMU Calibration based on Upsampled Preintegrated Measurements for Motion Distortion Correction", IEEE International Conference on Robotics and Automation, 2018

1.5.2 Side contributions

- C. Le Gentil*, M. Vayugundla*, R. Giubilato, W. Sturzl, T. Vidal-Calleja, R. Triebel, "Gaussian Process Gradient Maps for Loop-Closure Detection in Unstructured Planetary Environments", IEEE/RSJ International Conference on Intelligent Robots and Systems, 2020
- M. Usayiwevu, C. Le Gentil, J. Mehami, C. Yoo, R. Fitch, T. Vidal-Calleja, "Information Driven Self-Calibration for Lidar-Inertial Systems", IEEE/RSJ International Conference on Intelligent Robots and Systems, 2020
- B. Dai*, C. Le Gentil*, T. Vidal-Calleja, "Connecting the Dots for Real-Time LiDAR-based Object Detection with YOLO", Australasian Conference on Robotics and Automation, 2018

Chapter 2

Review of Related Work

In its early days, as shown in [5], localisation of autonomous vehicles was mainly relying on beacons at known positions in the environment. The same concept is used by Global Navigation Satellite Systems (GNSS's), in which multiple satellites orbiting about twenty thousand kilometres above our heads serve as beacons. While GNSS's provides substantial localisation information in many outdoor environments worldwide, they cannot be used indoors or in scenarios where line-of-sight with the satellites cannot be guaranteed. Over the years, the robotics community have been exploring diverse ways to estimate a system's pose using various sensor modalities that do not rely on beacons.

The affordability of both monocular cameras and IMUs makes visual-inertial algorithms quite popular in the literature [4]. Initially limited by computational power, online state estimation was mostly relying on filter-based methods such as the one presented in [6]. Later frameworks like [7] and [8] combined both local and global optimisations to estimate the system trajectory in real-time as well as creating consistent maps with loop-closure detections. Originally expensive and bulky, LiDARs are now affordable sensors that provide reliable geometric information about the surrounding environment. The system presented in [9] combines both a 2D LiDAR with an IMU on a spring mechanism to create 3D maps. Still considered as a state-of-the-art method, LOAM [10] cleverly uses frame-to-frame and frame-to-map scan registrations loosely coupled with IMU data to outperform any

other LiDAR-inertial approaches on the KITTI odometry benchmark [11]. All the methods above leverage only two sensors at a time, but fusion with more sensors is possible, as demonstrated in [12] with a visual-inertial-GPS system, or [13] with a LiDAR-visual-inertial odometry pipeline.

2.1 Preintegration

Whether it is for visual-inertial [7, 8] or LiDAR-inertial [14, 15] estimation, many methods rely on the concept of preintegration originally presented in [1]. The key idea is to dissociate the computation of the accelerometer data integrals from the initial pose and velocity. In the context of visual-inertial localisation and mapping, [1] introduces preintegrated measurements that combine IMU readings collected between two visual key-frames. Unlike the classic integration of inertial data, preintegration provides pseudo-measurements that are independent of the linearisation point preventing repetitive computations during the process of state estimation.

As shown in [16], the authors of [2, 17] introduce a lightweight and accurate VIO algorithm. The high performance of this algorithm has been made possible with the extension of the preintegration theory to the rotation manifold [2, 17], instead of the Euler angles representation used in [1]. In [14], the inertial data are combined into closed-form preintegrated measurements as per [18] relying on the assumption of constant local acceleration.

In our preliminary work [19], we introduce the concept of upsampled preintegration that first interpolates the IMU data before performing numerical integrations as in [17]. The Upsampled-Preintegrated-Measurements (UPMs) are based on GP regression and do not rely on any explicit assumption about the system dynamics. In this thesis, we further the use of GPs and introduce a novel application of linear operators on covariance kernels [20] to infer analytical integrals of inertial signals.

2.2 Continuous time state representation

One issue that arises with multi-sensor systems is the data synchronisation. Even if the readings of each modality are collected along with accurate real-time timestamps, the sensor frequencies are generally different and out of phase. Frameworks like [9] and [21] address this problem by using continuous state representations. In [9, 10, 22–24], the continuous state is computed through linear pose interpolations over small time intervals. It corresponds to a constant velocity motion model over short periods of time. The approach presented in [21] assumes that the continuous state of the system is a linear combination of basis functions (*B-splines*). As part of a Maximum Likelihood Estimation (MLE) batch optimisation, the coefficients of the linear combination are optimised by comparing, in the cost function, the sensor readings and the estimated state at the corresponding timestamps. With appropriately chosen basis functions, these methods can reduce the size of the state vector. Later, [25] extends this concept to address the issue of relative motion estimation allowing for the approximation of the full MLE solution with constant computational complexity. The approach presented in [26] also leverages B-splines as in [21]. However, [26] presents a singularity-free state representation by locally considering the Special Euclidean group in three dimensions (SE(3)) manifold as opposed to the Caley-Gibbs-Rodrigues [27] in [21]. This work has later been extended in [28]. While providing greater representability compared to traditional discrete models, the performance of continuous state frameworks depends on the veracity of the models assumed.

Based on GP regression [29], a new type of state estimation technique has been presented in [30] and [31]. Respectively with a minimal orientation representation and with Special orthonormal group in three dimensions (SO(3)) matrices, the continuous state inference is guided by embedding a simple motion model in the kernel of the GP-interpolation. While dealing with a continuous state, this method does not reduce the size of the estimated state compared to the number of sensor readings. The authors of [30] and [31] explicitly states in [32], that these methods still discretise the timeline to carry out a Maximum A Posteriori (MAP) estimation of the state at each sensor-reading timestamp before being able to infer the state at any other given time.

In [15, 19], we propose a different paradigm where GPs are used as continuous representations of the inertial data allowing the characterisation of the system's pose in a continuous manner while relying on a discrete state. The preintegration method developed in this thesis is built on a similar concept (GP representation of the inertial data). Like the UPMs from [19], the GPMs can be both used as a substitute for continuous state representations, as well as be seamlessly integrated into traditional discrete estimation frameworks. The non-parametric nature of the GP regression makes the proposed methods agnostic to any motion model.

2.3 Lidar localisation and mapping

Lidar scan geometric registration is the foundation of most of the laser-based localisation algorithms. The introduction of Iterative Closest Point (ICP) [33] and generalised ICP [34] allowed the estimation of the rigid transformation between two point clouds. The method in [35] estimates a system trajectory using ICP for frame-to-frame relative poses estimation, and a pose-graph to correct the drift inherent to any odometry-like frameworks. This method, among others, does not address the phenomenon of motion distortion in LiDAR scans.

Over the years, different approaches have been presented for state estimation of moving LiDAR/rolling-shutter sensors. The authors of [22] extend the 2D standard ICP to account for the LiDAR motion using the assumption of constant velocity during the sweep. This motion model assumption is often used in the literature as in [10] and [23]. Both techniques rely on linear pose-interpolation between state variables used as trajectory control points. In real-world scenarios, the constant-velocity assumption does not necessarily represent the true nature of the system's motion. In [10], the 6-DoF pose of the system is estimated based on data provided by a 3D LiDAR. While also achieving 6-DoF trajectory estimation, the method introduced in [23] (extended in [9]) uses a 2D LiDAR mounted on a spring with an IMU. Given that the other end of the spring is fixed to a moving vehicle, the 3D abilities of this system rely on the random and irregular motion of the spring-mounted sensor suite.

To reach real-time operations, techniques like [36] and [37] consider the correction of motion distortion at the front-end level, and not as part of the estimation process. This is also the case in [38] where point clouds are accumulated according to the platform's inertial-kinematic state before being registered via an auto-tuned ICP algorithm [39]. In other words, the prior knowledge of the actual motion is used to undistort the incoming point clouds, but no other action is taken later to improve the distortion correction according to the new state estimate. With such a strategy, there is a risk of accumulating drift due to inaccurate initial conditions. The GPMs proposed in this thesis enable accurate characterisation of the system's motion providing inertial information for each point of a point cloud. Such a feature allows for estimation frameworks to consider and correct motion distortion seamlessly all along the process.

At the time of writing, there is no global consensus in the literature upon how to represent and leverage LiDAR data in estimation frameworks. *Surfels* [40] are surface elements used to represent 3D information in the form of disks or ellipsoids. In the context of robotics state estimation, [41] introduces a probabilistic method to merge multiple point clouds made of surfels. As demonstrated in [9, 24, 36, 42], surfels can provide rich surface information and enable localisation and mapping. The method in [9] aggregates data provided by a 2D LiDAR into 3D surfels according to the IMU data before refining the system's trajectory in an open-loop scan-matching fashion. The nature of the system used makes the registration of LiDAR points fairly challenging. In such a case, surfels acquired under very fast motion are automatically discarded. The authors of [24] introduce a two-step mapping approach that reuses the concepts of [9] for local mapping, and [43] for global mapping. Originally developed for RGBD sensors, [43] presents a map-centric approach to create consistent global maps based on trajectory-agnostic batch optimisation. Compared to feature-based techniques discussed later, surfel-based methods tend to use more information for scan registration, generally resulting in computation-intensive processes. As seen in [36] and [42], a focus of the surfel-based LiDAR mapping is the development of efficient algorithms. Both methods make use of sub-mapping schemes (with a hierarchical optimisation mechanism in [42]) to reduce the computation cost of the state estimation.

As for visual data from cameras [4], the use of particular feature points instead of raw data is a common approach that allows for efficient computation of frame-to-frame LiDAR

scan registration. The vast majority of today's LiDARs consist of one or multiple lasers swept across the Field-of-View (FoV). It generally implies that LiDAR data are generally dense in the axis of the lasers' motion but quite sparse perpendicularly to the sweeping motion. For this reason, feature descriptors as in [44–47] are not adapted to raw LiDAR data. In 2014, the authors of [10] introduce a per-channel feature extraction technique as part of an odometry and mapping framework. This technique computes a curvature score for each of the points collected in the different LiDAR channels. Depending on that score, points are classified as edge or plane feature. Associating plane features of one scan with tuples of three plane features from another scan allows for the computation of point-to-plane distances that are minimised in the scan registration process. In the same way, the association of edge features with pairs of edge features lead to point-to-line distances that are minimised alongside with the point-to-plane ones. This paradigm is especially suited for spinning LiDARs that have a low vertical resolution and can be found in different frameworks [37, 48]. It inspired our front-end development in Chapter 4.

IMLS-SLAM [49] leverages more generic "features", that can be used in weakly structured environments, by using a specific sampling of LiDAR scans, . In [50], the point sampling is performed according to a 3D grid. The approach presented in [14] estimates both the system trajectory and the position of planes in the environment based on a novel plane representation. Such formulation reduces the complexity of the optimisation problem (further than in the methods aforementioned) and filters the individual measurements' noise. Nonetheless, this method is suitable solely for highly structured environments that contain very large planar surfaces. The authors of [51] proposed a LiDAR feature extraction method that efficiently detects planes and 3D-lines in structured environments.

In this paragraph, we give a rapid overview of several LiDAR-camera frameworks for localisation and mapping. In [52], the authors present a visual-LiDAR algorithm for pose estimation furthering their previous works on LiDAR odometry [10] and LiDAR-aided monocular navigation [53]. Later, this visual-LiDAR method is improved by the integration of inertial data into a coarse-to-fine sequential data processing pipeline [54] that differs both from traditional filter-based and optimisation-based state estimation. In addition to achieving fast computation, the authors claim that this novel paradigm offers more robustness with respect to sensor failures and observability degradation of

the individual sensors (e.g., a LiDAR moving in a tunnel does not allow for the position estimation along the tunnel axis). In [55], a 3-DoF IMU is used to aid an Extended Kalman filter (EKF)-based visual-LiDAR odometry pipeline. The method in [56] presents a technique to extract visual feature tracks' depth from LiDAR data allowing for robustified bundle adjustment.

Recently, with the development of machine learning algorithms for semantic understanding of 3D data [57], work has been conducted to perform localisation and mapping leveraging high-level information about the environment. In [58], a pipeline for loop-closure detection in LiDAR data is presented. It relies on learnt *random forest* classifiers [59] using engineered features (based on Eigenvalues and histograms) extracted from geometrically segmented point clouds. These concepts are extended to a standalone LiDAR localisation and mapping framework in [60] with neural-network-based learnt segment descriptors. Later, the authors of [61] reuse the principle of segmentation and descriptor extraction on LiDAR data. By associating this previous work with vision-based descriptors from neural networks [62], and efficiently maintaining a *k-d tree* database of previously visited areas, [61] can perform instant localisation of visual-LiDAR systems. In [63], a more traditional surfel-based framework [42] is extended with semantic information obtained with RangeNet++ [64], to filter dynamic object in the environment that could perturb the geometric LiDAR scan odometry. Note that the methods described in this paragraph are mostly designed for structured environments, especially for urban navigation scenarios.

2.4 Event-based odometry

Traditional frame-based and event-based vision have major differences. Frame-based Visual Odometry (VO) is challenged in many scenarios. For example, the images collected in the presence of fast motion are affected by motion blur due to the pixels' exposure time. This makes feature extraction and tracking especially difficult, and often leads to trajectory estimation failures. On the other hand, the data produced by event-cameras correspond to intensity changes observed in each individual pixel [65, 66]. Hence, the resulting data stream is not subject to motion-blur. Another advantage of event-based vision over frame-based vision is the HDR ability that prevents event-cameras from being

"blinded" by extremely bright light sources in the FoV. These aforementioned characteristics make event cameras especially appealing sensors for the task of VO. Nonetheless, the unconventional nature of the data produced represents a challenge for the robotics research community, and requires the design of novel algorithms and paradigms.

Despite the relatively recent interest in event cameras, we can already identify several works on VO employing event cameras over the last few years. The first event-based 2D SLAM approach is presented in [67]. It relies on the combination of a particle-filter-based pose estimation algorithm and a dynamically updated map. More recently, the authors of [68] achieve the estimation of the six DoFs of the camera pose by leveraging EKFs, and separating the mapping and pose tracking components as in [69]. The method introduced in [70] proposes a parallel mapping and tracking approach that iteratively co-localises the pose of the camera against a local map of edges represented by with voxel grid.

The aforementioned approaches avoid the explicit definition of features at the expense of being relatively demanding in terms of computational resources. Modern event-based VIO pipelines such as [71] and [72], however, rely on the detection and tracking of corners employing intermediate image-like representations by accumulating events over time. These two approaches also make use of IMUs, profiting from their high-rate inertial readings and making them an appealing type of sensor to be combined with event cameras. Despite being a promising modality for visual-inertial state estimation, especially for fast motion and HDR scenes, event-cameras can be challenged in scenarios in which the system is almost static. In [73], the authors leverage the complementary of events and traditional images to address the system's drift, especially in the context of hovering drones.

While the majority of event-based VO approaches still rely on the traditional concept of key-frames, it is only natural that continuous-time approaches would emerge. In one of the seminal works [74], the state of the camera is estimated associating events with line segments in given maps of the environment. The method is evaluated both in simulation and real-world experiments using fiducial markers detected in intensity images. The continuous-time estimation relies on cumulative B-Spline interpolation between estimated camera poses. The same authors later expand their approach to consider inertial measurements in [75].

Over the years, significant efforts in the community have been dedicated to the definition of reliable visual features for event data. It has led to the development of mainly corner detection and tracking approaches. In [76], the Harris corner detection algorithm [77] is adapted to the asynchronous nature of the event data by considering local contrast maps around each event. The local contrast maps are defined as the agglomeration of a fixed number of events. While this definition is relatively invariant to the systems speed, it is scene-dependent. The authors of [78] introduce a corner extraction method inspired by eFAST [79]. It proposes a version of the *surface of active events* [79, 80] (also referred as *time surface* [81]) that filters redundant events to accurately represent the position of corners in the image space. Another event-based asynchronous corner detector is presented in [82]. It relies on a novel region descriptor for event data. The approach described in [83] learns to detect corner events using random forests [59] over a new speed-invariant time surface. In [84], the authors provide an extensive survey of event-based vision methods.

The progress of line-based features has been comparatively less notable. Among other works, we could highlight the approaches proposed in [85], detecting and tracking line clusters in an event-by-event fashion, and in [86], that propose a plane fitting approach on the event stream based on principal component analysis to track and cluster lines.

The proposed framework in Chapter 5 takes inspiration from [74] and [75] as we also leverage continuous-time representation and inertial data. However, where [74] and [75] use a continuous formulation of the state, our framework uses GPMs associated with a discrete state to accounts rigorously for the continuous trajectories. At the front-end level, we draw concepts from event-driven line-tracking and clustering approaches (such as the methods described in [85] and [86]), avoiding the need for supplementary sensing modalities as frame-based images and operating directly on the asynchronous event stream.

2.5 Extrinsic calibration

Behind the terms of extrinsic calibration stands the technique of estimating the relative spatial position between two sensors. In most of the cases, the calibration is a punctual process happening several times in the life of a sensor system. Therefore, the majority

of the calibration methods make use of calibration rigs or targets. This extra hardware allows to partially control, and therefore to partially know, the environment in which the system is operated. In this section, the reader will find an overview of the current state of the literature for extrinsic calibration techniques, which are related to IMUs and LiDARs.

The rise of the self-driving cars resulted in a number of specific calibration problems. One of them was the calibration between a 3D-LiDAR and an inertial navigation system [87–89]. An inertial navigation system comprises not only an IMU but also a GNSS and potentially wheel encoders. In [88], the authors estimate the relative transformation between the sensors using a grid search algorithm and relying on the weak assumption that 3D-points tend to lie on contiguous surfaces. Because current spinning LiDARs provide sparse geometric information about the environment, it is common to assume that planes are present in the surroundings of the system. In many real-life scenarios, it is a rightful supposition, especially in human-made environment.

To determine the relative position between a LiDAR and an IMU, one can use a camera as a variety of camera-LiDAR, and camera-IMU calibration methods can be found in the literature.

As stated above, most of the calibration methods make use of particular set-ups. Over the years, the robotics community has seen the complexity of the calibration rigs decreasing. In a visual-inertial context, [90] and [91] are examples of the need for complex calibration rigs. These methods rely on a spinning table with shaft encoders to be able to recover the geometric transformation between the camera and the IMU. A few years later, techniques such as [92] and [93] demonstrate high accuracy calibration using only a checkerboard as a calibration target. In [94], using a similar visual pattern, the authors extend the scope of the visual-inertial calibration considering separately the three single-axis accelerometers that an IMU comprises.

When it comes to aligning LiDAR point clouds and camera images, the checkerboard is again a widely used calibration target. Whether it be in the monocular modality or the 2D and 3D range data, this planar visual pattern can be automatically extracted relatively easily. There are examples in [95], [96] and the implementation of Autoware [97]. Other

visual-geometric patterns are used in [98] for 2D and 3D-LiDARs, and in [99, 100] for 3D-LiDARs only.

As the manufacture of any physical object cannot be perfect, when one uses a particular installation (rig or target) for its calibration procedure, it incurs some unquantified uncertainty into the transformation estimated. This fact partially explains the trend in the literature to limit the use of calibration-dedicated hardware. In addition, having target-less calibration procedures allows a continuous calibration of the sensors, therefore removing the necessity for the user to stop the system's operation to calibrate it regularly. Based on the assumption that high colour gradient in the visual data corresponds to a high spatial gradient in 3D-point clouds, [101] and [102] offer good examples of target-less calibration techniques for cameras and LiDARs. Another interesting work is presented in [103]. After showing the correlation between the laser reflectivity and the camera intensity values, the authors recover the inter-sensor spatial transformation by optimising the mutual information in the multi-modal data.

Most calibration procedures are sensor-specific and rely on an overlap, or a "bridge", between modalities that often requires a constant FoV overlap. One can think about planes simultaneously visible in LiDAR and camera data (e.g., [96]), the correlation between appearance and depth gradients observed with visual-LiDAR systems (e.g., [101]), etc. In [104], a target-less extrinsic calibration involving a 2D-LiDAR and a visual-inertial sensor is described. This method does not need any FoV overlap as it optimises the planarity of LiDAR point clouds that are registered according to the trajectory of the visual-inertial sensor and the calibration parameters. For this method to be efficient and accurate, the authors specify a set of assumptions including the presence of planar patches in the LiDAR data (as in [88]), and the ability for the visual-inertial sensor to be capable of estimating a sufficiently accurate trajectory.

Pushing further the abstraction, there are in the literature some generic multi-modal calibration frameworks that perform calibration based on high-level information. The work presented in [105] (for coplanar sensors only) and [89] base their approach on the different sensors' motion. Such methods are very generic for sensors that are able to estimate their trajectory self-sufficiently. Consequently, this paradigm allows the extrinsic calibration of

sensors that do not have overlapping modalities nor intersecting FoVs. In other words, as long as their trajectories can be computed independently, the inter-sensor transformation can be estimated. Despite their versatility, these techniques are not suitable for calibrating with an IMU constituted of only a 3-axis accelerometer and a 3-axis gyroscope. In real-life scenarios, it is impossible to recover an accurate motion estimation solely from inertial data as the initial conditions are not precisely known, and the noise of the measurements rapidly introduces drift in the estimation. Extensive work on multi-modal calibration is done in [106].

Despite the popularity of LiDARs and IMUs, the calibration between these sensors has not been extensively studied. Our work in [19] proposes a calibration process based on a simple calibration target (a set of planes). As we saw in this section, throughout the years, efforts have been produced to move toward target-less calibration procedures. Our proposed method described in Chapter 4 follows this line of thought by allowing target-less extrinsic calibration of a 3D-LiDAR and a 6-DoF-IMU, thus removing the need for a dedicated calibration rig/environment/pre-defined actions.

Chapter 3

Gaussian Preintegration

3.1 Introduction

As IMUs became ubiquitous, numerous autonomous systems rely on inertial-aided sensor-fusion algorithms for navigation. If we consider the example of frame-based VIO systems, the IMU produces measurements at a higher frequency than the camera. Combining IMU measurements between visual keyframes became a standard way to reduce the computational cost of pose estimation algorithms.

Given inertial measurements provided by a 6-DoF IMU, constituted of a 3-axis accelerometer and a 3-axis gyroscope, one can use the laws of classical mechanics to estimate the trajectory of a system by integration from a set of known initial conditions. The trajectory estimate is tightly coupled with the values of the initial conditions as an accelerometer measures the *proper acceleration* of the system. The proper acceleration is the acceleration relative to a free-fall observer who is at rest with respect to the system under observation. In other words, the accelerometer's readings are affected by the Earth's gravity field. Therefore, the accurate attitude of the system based on the initial orientation conditions is needed to deduce the coordinate acceleration from the accelerometer's readings. In the context of state estimation, the initial conditions are part of the variables to be estimated. With the classic integration scheme, any change to the orientation variable leads to the

re-computation of the integrals to update the trajectory estimate. The concept of preintegration, originally introduced in [1] and later extended to the SO(3) manifold in [2], allows a dissociation between the computation of the integrals and the initial conditions. This mechanism helped the state estimation research community to build computationally more efficient frameworks without sacrificing accuracy.

The proposed method extends further the concept of preintegration by considering continuous and probabilistic representations of the inertial measurements. This novel approach allows the computation of preintegrated measurements that do not suffer from the numerical integration noise present in [1] and [2].

This chapter presents a method called Gaussian preintegration. It is a continuous preintegration technique based on GP regression and the application of linear operators to the kernel covariance functions. Given 6-DoF IMU measurements as input, the proposed method outputs pseudo-measurements denoted GPMs that are particularly suited for inertial-aided state estimation.

The work presented in this chapter corresponds to the contribution published in [107] as listed in Section 1.5. The rest of this chapter is organised as follows. First, we expose the problem statement and give background knowledge in Section 3.2 and Section 3.3 respectively. Then, Section 3.4 presents the core of the proposed preintegration method. The created preintegrated measurements come in parallel with postintegration mechanisms for bias and time-shift corrections. These mechanisms are developed in Section 3.5. The effectiveness of our method is demonstrated in Section 3.6. Finally, conclusions are displayed in Section 3.7.

3.2 Problem statement

3.2.1 System description

Let us consider a 6-DoF-IMU composed of a 3-axis accelerometer and a 3-axis gyroscope. The inertial data acquired consists of proper accelerations $\tilde{\mathbf{f}}(\mathrm{t}_i)$ and angular velocities $\tilde{\boldsymbol{\omega}}(\mathrm{t}_i)$ at time t_i $(i = 1, \ldots, Q)$ measured in the inertial frame $\mathfrak{F}_I$. The IMU orientation

(represented by a rotation matrix), position and velocity at time t_i are denoted by $\mathbf{R}_W^{t_i}$, $\mathbf{p}_W^{t_i}$ and $\mathbf{v}_W^{t_i}$, respectively. The subscript $_W$ corresponds to the world fixed frame $\mathfrak{F}_W$.

The inertial measurements are modelled considering additive biases and noise terms as follows:

$$\tilde{\mathbf{f}}(t) = \mathbf{R}_W^t(t)^\top(\mathbf{f}(t) - \mathbf{g}) + \mathbf{b}_f(t) + \eta_f(t), \tag{3.1}$$

$$\tilde{\boldsymbol{\omega}}(t) = \boldsymbol{\omega}(t) + \mathbf{b}_\omega(t) + \eta_\omega(t), \tag{3.2}$$

with $\mathbf{f}$ being the true linear acceleration of the sensor in the world frame $\mathfrak{F}_W$, $\boldsymbol{\omega}$ the true instantaneous angular velocity of the inertial frame relative to $\mathfrak{F}_W$, $\mathbf{g}$ the gravity vector in $\mathfrak{F}_W$, $\mathbf{b}_f$ and $\mathbf{b}_\omega$ slowly varying sensor biases, η_f and η_ω zero-mean Gaussian noises for the linear accelerations and angular velocities respectively.

By definition, the dynamics of the sensor is given by:

$$\dot{\mathbf{R}}_W^t(t) = \mathbf{R}_W^t(t)\boldsymbol{\omega}(t)^\wedge, \tag{3.3}$$

$$\dot{\mathbf{v}}_W^t(t) = \mathbf{f}(t), \tag{3.4}$$

$$\dot{\mathbf{p}}_W^t(t) = \mathbf{v}_W^t(t), \tag{3.5}$$

where $\dot{}$ is the differentiation operator, and $^\wedge$ is the operator that transforms a 3-by-1 vector into a skew-symmetric matrix as follows

$$\boldsymbol{\omega}^\wedge = \begin{bmatrix}\omega_1\\ \omega_2\\ \omega_3\end{bmatrix}^\wedge = \begin{bmatrix}0 & -\omega_3 & \omega_2\\ \omega_3 & 0 & -\omega_1\\ -\omega_2 & \omega_1 & 0\end{bmatrix}. \tag{3.6}$$

3.2.2 IMU preintegration

Given known initial conditions at $t = t_1$, the sensor pose and velocity at $t_2 > t_1$ can be computed by integrating (3.3), (3.4) and (3.5):

$$\mathbf{R}_W^{t_2} = \mathbf{R}_W^{t_1}\Big(\prod_{t_1}^{t_2} \exp\big(\boldsymbol{\omega}(t)^\wedge\big)^{dt}\Big), \tag{3.7}$$

$$\mathbf{v}_W^{t_2} = \mathbf{v}_W^{t_1} + \int_{t_1}^{t_2} \mathbf{f}(t)dt, \tag{3.8}$$

$$\mathbf{p}_W^{t_2} = \mathbf{p}_W^{t_1} + \mathbf{v}_W^{t_1}\Delta t + \int_{t_1}^{t_2}\int_{t_1}^{t} \mathbf{f}(s)dsdt \tag{3.9}$$

with $\Delta t = t_2 - t_1$ and $\exp(\boldsymbol{\phi}^\wedge)$ the mapping from Lie algebra of special orthonormal group in three dimensions ($\mathfrak{so}(3)$) to SO(3) computed as (ϕ a 3-by-1 angle-axis vector):

$$\exp(\boldsymbol{\phi}^\wedge) = \mathbf{I} + \frac{\sin(\|\boldsymbol{\phi}\|)}{\|\boldsymbol{\phi}\|}\boldsymbol{\phi}^\wedge + \frac{1-\cos(\|\boldsymbol{\phi}\|)}{\|\boldsymbol{\phi}\|^2}(\boldsymbol{\phi}^\wedge)^2. \tag{3.10}$$

In the absence of sensor noise and given accurate prior knowledge of the accelerometer and gyroscope biases $\bar{\mathbf{b}}_f$ and $\bar{\mathbf{b}}_\omega$, (3.7), (3.8) and (3.9) can be expressed as a function of the IMU readings $\tilde{\mathbf{f}}$ and $\tilde{\boldsymbol{\omega}}$:

$$\mathbf{R}_W^{t_2} = \mathbf{R}_W^{t_1}\Big(\prod_{t_1}^{t_2} \exp\big((\tilde{\boldsymbol{\omega}}(t) - \bar{\mathbf{b}}_\omega(t))^\wedge\big)^{dt}\Big), \tag{3.11}$$

$$\mathbf{v}_W^{t_2} = \mathbf{v}_W^{t_1} + \mathbf{g}\Delta t + \int_{t_1}^{t_2} \mathbf{R}_W^t(t)\big(\tilde{\mathbf{f}}(t) - \bar{\mathbf{b}}_f(t)\big)dt, \tag{3.12}$$

$$\begin{aligned}\mathbf{p}_W^{t_2} =& \mathbf{p}_W^{t_1} + \mathbf{v}_W^{t_1}\Delta t + \frac{\mathbf{g}\Delta t^2}{2} \\ &+ \int_{t_1}^{t_2}\int_{t_1}^{t} \mathbf{R}_W^s(s)\big(\tilde{\mathbf{f}}(s) - \bar{\mathbf{b}}_f(s)\big)dsdt.\end{aligned} \tag{3.13}$$

The preintegration originally presented in [1] reformulates (3.12) and (3.13) using the fact that $\mathbf{R}_W^t(t) = \mathbf{R}_W^{t_1}\mathbf{R}_{t_1}^t(t)$:

$$\mathbf{v}_W^{t_2} = \mathbf{v}_W^{t_1} + \mathbf{g}\Delta t + \mathbf{R}_W^{t_1}\int_{t_1}^{t_2}\mathbf{R}_{t_1}^t(t)\big(\tilde{\mathbf{f}}(t) - \bar{\mathbf{b}}_f(t)\big)dt, \tag{3.14}$$

$$\mathbf{p}_W^{t_2} = \mathbf{p}_W^{t_1} + \mathbf{v}_W^{t_1}\Delta t + \frac{\mathbf{g}\Delta t^2}{2} + \mathbf{R}_W^{t_1}\int_{t_1}^{t_2}\int_{t_1}^{t}\mathbf{R}_{t_1}^s(s)\big(\tilde{\mathbf{f}}(s) - \bar{\mathbf{b}}_f(s)\big)dsdt. \tag{3.15}$$

The preintegrated measurements are defined as

$$\Delta\mathbf{R}_{t_1}^{t_2} = \prod_{t_1}^{t_2}\exp\left((\tilde{\boldsymbol{\omega}}(t) - \bar{\mathbf{b}}_\omega(t))^\wedge\right)^{dt}, \tag{3.16}$$

$$\Delta\mathbf{v}_{t_1}^{t_2} = \int_{t_1}^{t_2}\mathbf{R}_{t_1}^t(t)\big(\tilde{\mathbf{f}}(t) - \bar{\mathbf{b}}_f(t)\big)dt, \tag{3.17}$$

$$\Delta\mathbf{p}_{t_1}^{t_2} = \int_{t_1}^{t_2}\int_{t_1}^{t}\mathbf{R}_{t_1}^s(s)\big(\tilde{\mathbf{f}}(s) - \bar{\mathbf{b}}_f(s)\big)dsdt, \tag{3.18}$$

and (3.11), (3.12), and (3.13) become

$$\mathbf{R}_W^{t_2} = \mathbf{R}_W^{t_1}\Delta\mathbf{R}_{t_1}^{t_2}, \tag{3.19}$$

$$\mathbf{v}_W^{t_2} = \mathbf{v}_W^{t_1} + \mathbf{g}\Delta t + \mathbf{R}_W^{t_1}\Delta\mathbf{v}_{t_1}^{t_2}, \tag{3.20}$$

$$\mathbf{p}_W^{t_2} = \mathbf{p}_W^{t_1} + \mathbf{v}_W^{t_1}\Delta t + \frac{\mathbf{g}\Delta t^2}{2} + \mathbf{R}_W^{t_1}\Delta\mathbf{p}_{t_1}^{t_2}. \tag{3.21}$$

In [1] and [2], (3.16), (3.17) and (3.18) are integrated numerically considering the discrete raw IMU measurements directly. This work presents a novel approach that models the IMU measurements with GPs and performs probabilistic analytical inference of the different integrals.

3.3 Definitions and background

The proposed method in this chapter relies on the use of GP as a continuous probabilistic representation of the inertial readings, and the application of linear operators on the GP covariance kernels to infer preintegrated measurements directly. In this section, we first

offer a succinct reminder of the standard GP regression and the application of linear operators. Then, we present the sensory system that will consider in this work. Finally, we give a reminder about the preintegration concept.

3.3.1 Gaussian Process regression

GP regression is a kernel-based non-parametric probabilistic interpolation method. While more details about GP regression can be found in [29], this subsection presents the standard inference method for zero-mean GP signals based on noisy observations.

The key principle of GP regression is the characterisation of a signal by its covariance function. This function is called the kernel. Let us consider a signal $h : \mathbb{R} \to \mathbb{R}$ represented by a GP as

$$h(x) \sim \mathcal{GP}\big(0, k_h(x, x')\big). \tag{3.22}$$

Note that in this work, we only consider single-input-single-output signals. Consequently, $h : \mathbb{R} \to \mathbb{R}$. The kernel function $k_h(x, x')$ gives the covariance between the signal values at input x and x':

$$\text{cov}\big(h(x), h(x')\big) = k_h(x, x'). \tag{3.23}$$

The stacked vector of noiseless observations $\mathbf{h}$ of h at input x_i $(i = 1, \cdots, Q)$ (so called training inputs), and h^* an inferred value of h at input x (also call test value) follows a multivariate Gaussian distribution

$$\begin{bmatrix} \mathbf{h} \\ h^* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K}_h(\mathbf{x}, \mathbf{x}) & \mathbf{k}_h(\mathbf{x}, x) \\ \mathbf{k}_h(x, \mathbf{x}) & k_h(x, x) \end{bmatrix}\right) \tag{3.24}$$

with

$$\mathbf{h} = \begin{bmatrix} h(x_1) \\ \cdots \\ h(x_Q) \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ \cdots \\ x_Q \end{bmatrix}, \tag{3.25}$$

$$\mathbf{k}_h(x, \mathbf{x}) = \begin{bmatrix} k_h(x, x_1) & \cdots & k_h(x, x_Q) \end{bmatrix}, \quad \mathbf{k}_h(\mathbf{x}, x) = \mathbf{k}_h(x, \mathbf{x})^\top$$

and

$$\mathbf{K}_h(\mathbf{x}, \mathbf{x}) = \begin{bmatrix} k_h(x_1, x_1) & k_h(x_1, x_2) & \cdots & k_h(x_1, x_Q) \\ k_h(x_2, x_1) & k_h(x_2, x_2) & \cdots & k_h(x_2, x_Q) \\ \vdots & \vdots & \ddots & \vdots \\ k_h(x_Q, x_1) & k_h(x_Q, x_2) & \cdots & k_h(x_Q, x_Q) \end{bmatrix}.$$

By *conditioning* the joint distribution (3.24), the test value follows a distribution characterised as

$$\begin{aligned} h^*|x, \mathbf{x}, \mathbf{h} \ \sim \mathcal{N}\big(&\mathbf{k}_h(x, \mathbf{x})\mathbf{K}_h(\mathbf{x}, \mathbf{x})^{-1}\mathbf{h}, \\ &k_h(x, x) - \mathbf{k}_h(x, \mathbf{x})\mathbf{K}_h(\mathbf{x}, \mathbf{x})^{-1}\mathbf{k}_h(\mathbf{x}, x)\big). \end{aligned} \tag{3.26}$$

When considering noisy observations,

$$y_i = h(x_i) + \eta \quad \text{with} \quad \eta \sim \mathcal{N}(0, \sigma_y^2), \tag{3.27}$$

the covariance function becomes

$$\operatorname{cov}\big(y(x), y(x')\big) = k_h(x, x') + \sigma_y^2 \delta_{xx'}, \tag{3.28}$$

with $\delta_{xx'}$ the Kronecker delta function (equal to one if and only if $x = x'$, zero otherwise). Consequently, the joint probability of the training and test values is

$$\begin{bmatrix} \mathbf{y} \\ h^* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K}_h(\mathbf{x}, \mathbf{x}) + \sigma_y^2 \mathbf{I} & \mathbf{k}_h(\mathbf{x}, x) \\ \mathbf{k}_h(x, \mathbf{x}) & k_h(x, x) \end{bmatrix}\right), \tag{3.29}$$

and after conditioning, the prediction is obtained as

$$h^*(x) = \mathbf{k}_h(x, \mathbf{x})\big[\mathbf{K}_h(\mathbf{x}, \mathbf{x}) + \sigma_y^2\mathbf{I}\big]^{-1}\mathbf{y} \tag{3.30}$$

$$\mathrm{var}\big(h^*(x)\big) = k_h(x, x) - \mathbf{k}_h(x, \mathbf{x})\big[\mathbf{K}_h(\mathbf{x}, \mathbf{x}) + \sigma_y^2\mathbf{I}\big]^{-1}\mathbf{k}_h(\mathbf{x}, x). \tag{3.31}$$

3.3.2 Gaussian Process inference with linear operators

In this subsection, we expose the application of linear operators in the context of GP regression, originally introduced in [20], tailored to our specific usage in the rest of this chapter.

From a GP modelled signal

$$h(x) \sim \mathcal{GP}\big(0, k_h(x, x')\big), \tag{3.32}$$

with noisy observations

$$y_i = h(x_i) + \eta \quad \text{with} \quad \eta \sim \mathcal{N}(0, \sigma_y^2), \tag{3.33}$$

we aim at inferring the output $g(x)$ resulting from the application of a linear operator $\mathcal{L}_g$ to $h(x)$:

$$g(x) = \mathcal{L}_g^x h(x). \tag{3.34}$$

In the case of linear operators, $\mathcal{L}_g^x h(x)$ does not correspond to the multiplication of a matrix or vector $\mathcal{L}_g^x$ with $h(x)$. It represents the application of the operator $\mathcal{L}_g^x$ on $h(x)$. This work considers the integration or differentiation of single-input-single-output signals. Therefore, $h : \mathbb{R} \to \mathbb{R}$ and $g : \mathbb{R} \to \mathbb{R}$. Note that the superscript $^\bullet$ of a linear operator $\mathcal{L}_\star^\bullet$, in (3.34) and the rest of this document, represents the variable on which the operator is applied. Examples of linear operators leveraged in this work are shown in Fig. 3.1, that includes the *derivative* and the *antiderivative* operators.

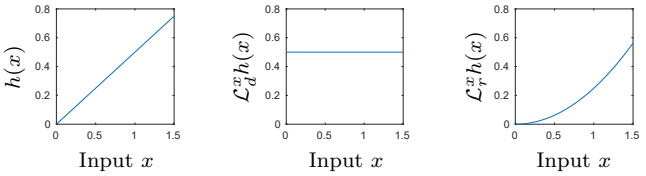


FIGURE 3.1: Graphical examples of two linear operators used in this work (center and right) applied to the signal $h(x)$ (shown on the left). With $\mathcal{L}_d^x h(x) = \frac{\partial h(x)}{\partial x}$ and $\mathcal{L}_r^x h(x) = \int_{x_1}^{x} h(t)dt$ (with $x_1 = 0$).

According to the rules for linear transformations of GPs, the joint distribution of the training and test values is

$$\begin{bmatrix} \mathbf{y} \\ g^* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K}_h(\mathbf{x},\mathbf{x}) + \sigma_y^2\mathbf{I} & \mathbf{k}_h(\mathbf{x},x)\mathcal{L}_g^x \\ \mathcal{L}_g^x\mathbf{k}_h(x,\mathbf{x}) & \mathcal{L}_g^x k_h(x,x)\mathcal{L}_g^x \end{bmatrix}\right). \tag{3.35}$$

Applying the conditioning as in (3.26):

$$\begin{aligned} g^*(x) &= \mathcal{L}_g^x\mathbf{k}_h(x,\mathbf{x})\big[\mathbf{K}_h(\mathbf{x},\mathbf{x}) + \sigma_y^2\mathbf{I}\big]^{-1}\mathbf{y} \\ \operatorname{var}\big(g^*(x)\big) &= \mathcal{L}_g^x k_h(x,x)\mathcal{L}_g^x - \mathcal{L}_g^x\mathbf{k}_h(x,\mathbf{x})\big[\mathbf{K}_h(\mathbf{x},\mathbf{x}) + \sigma_y^2\mathbf{I}\big]^{-1}\mathbf{k}_h(\mathbf{x},x)\mathcal{L}_g^x. \end{aligned} \tag{3.36}$$

Note that the right product of the linear operator implies its application to the second argument of the preceding kernel function. This is emphasised by the superscript of the linear operator.

3.4 Gaussian Preintegrated Measurements

Our work in [19] and [15] considers continuous representations of the inertial data to independently upsample these signals before numerically integrating (3.16), (3.17) and (3.18), as in [2]. GP regression is used to perform the probabilistic upsampling of the inertial data. We name these interpolated measurements Upsampled Preintegrated Measurements (UPMs). A rapid overview of this method is presented in Appendix A.

Here, the proposed method also leverages continuous models of inertial measurements. The major difference is that in this case we leverage the use of linear operators [20] to analytically compute the integral of inertial readings. Fig. 3.2 shows the two step approach

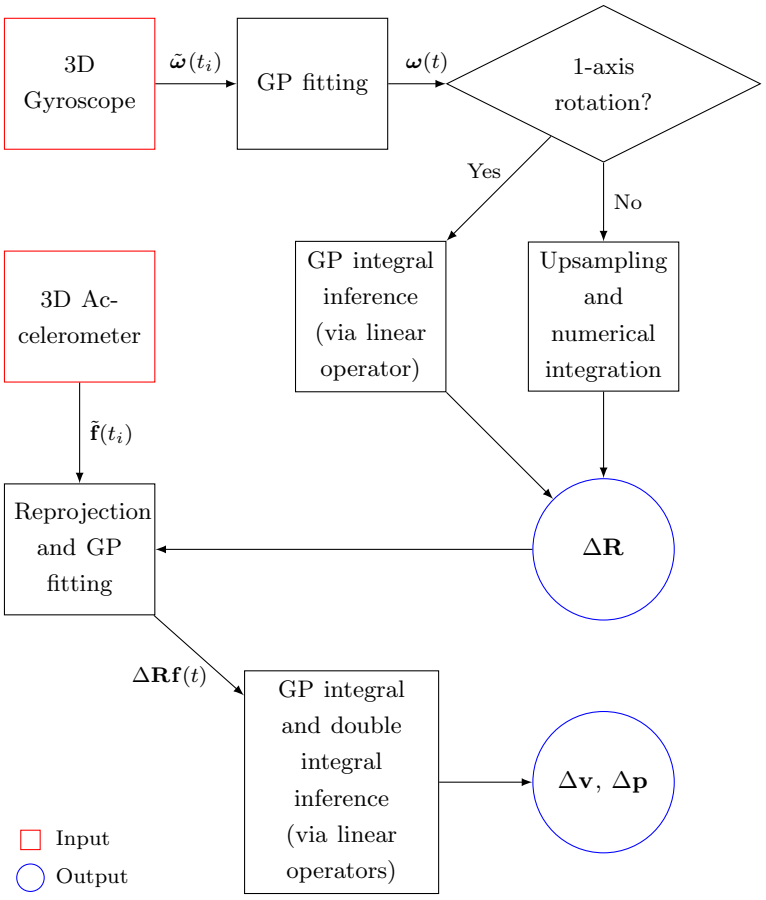


FIGURE 3.2: Overview of the proposed method of Gaussian Process (GP) preintegration

(rotation first, then velocity and position) proposed for inference of GPMs. In this section, we assume that the sensors biases are known. In real-world scenarios, the value of the biases is not accurately known a priori. A first-order expansion technique is presented in Section 3.5 to allow for postintegration bias and inter-sensor time-shift corrections.

3.4.1 GPM - Rotation

Equation (3.3) does not have a general solution [108]. Nonetheless, if the sensor rotation is constrained around one-axis, as in many ground vehicle applications, the infinitesimal rotations become commutative. Therefore, in that case, (3.3) can be solved as:

$$\Delta\mathbf{R}_{t_1}^{t_2} = \exp\left(\Delta\mathbf{r}_{t_1}^{t_2\wedge}\right), \quad \Delta\mathbf{r}_{t_1}^{t_2} = \int_{t_1}^{t_2} (\tilde{\boldsymbol{\omega}}(t) - \bar{\mathbf{b}}_{\omega}(t))dt, \tag{3.37}$$

where the three components of the integral can be computed independently. The integral is then rewritten with a linear operator

$$\Delta\mathbf{r}_{t_1}^{t_2} = \int_{t_1}^{t_2} (\tilde{\boldsymbol{\omega}}(t) - \bar{\mathbf{b}}_\omega(t))dt = \begin{bmatrix} \Delta r_1{}_{t_1}^{t_2} \\ \Delta r_2{}_{t_1}^{t_2} \\ \Delta r_3{}_{t_1}^{t_2} \end{bmatrix} = \begin{bmatrix} \mathcal{L}_r^t(\omega_1(t) - \bar{b}_{\omega_1}(t)) \\ \mathcal{L}_r^t(\omega_2(t) - \bar{b}_{\omega_2}(t)) \\ \mathcal{L}_r^t(\omega_3(t) - \bar{b}_{\omega_3}(t)) \end{bmatrix}. \tag{3.38}$$

Modelling the three components of $(\tilde{\boldsymbol{\omega}}(t) - \bar{\mathbf{b}}_\omega(t))$ with GPs independently,

$$(\omega_j(t) - \bar{b}_{\omega_j}(t)) \sim \mathcal{GP}\big(0, k_{r_j}(t, t')\big), \tag{3.39}$$

the components of $\Delta\mathbf{r}_{t_1}^{t_2}$ are inferred independently, following (3.36), as

$$\Delta r_j{}_{t_1}^{t_2 *} = \mathcal{L}_r^t \mathbf{k}_{r_j}(t, \mathfrak{t}) \big[\mathbf{K}_{r_j}(\mathfrak{t}, \mathfrak{t}) + \sigma_{\omega_j}^2 \mathbf{I}\big]^{-1} \boldsymbol{\omega}_j, \tag{3.40}$$

$$\mathrm{var}(\Delta r_j{}_{t_1}^{t_2 *}) = \mathcal{L}_r^t k_{r_j}(t, t) \mathcal{L}_r^t - \mathcal{L}_r^t \mathbf{k}_{r_j}(t, \mathfrak{t}) \big[\mathbf{K}_{r_j}(\mathfrak{t}, \mathfrak{t}) + \sigma_{\omega_j}^2 \mathbf{I}\big]^{-1} \mathbf{k}_{r_j}(\mathfrak{t}, t) \mathcal{L}_r^t. \tag{3.41}$$

with $\boldsymbol{\omega}_j$ being the vector of training values $(\tilde{\boldsymbol{\omega}}(t) - \bar{\mathbf{b}}_\omega(t))_j$.

If the rotation spans over multiple axes between t_1 and t_2, the rotational preintegrated measurement $\Delta\mathbf{R}_{t_1}^{t_2}$ is computed numerically as in [19] and [15]. First, the gyroscope data is upsampled with the classic GP inference (3.31) at virtual measurement times t_{v_i} ($i = 1, \cdots, V$ with $t_{v_i} < t_{v_{i+1}}$, $t_1 = t_{v_1}$, and $t_2 = t_{v_V}$): $\tilde{\boldsymbol{\omega}}^*(t_{v_i})$. Then, the rotation preintegrated measurements are iteratively computed as

$$\Delta\mathbf{R}_{t_1}^{t_2} = \prod_{i=1}^{V-1} \exp\Big(\big((\tilde{\boldsymbol{\omega}}^*(t_{v_i}) - \bar{\mathbf{b}}_\omega(t_{v_i}))(t_{v_{i+1}} - t_{v_i})\big)^\wedge\Big). \tag{3.42}$$

The uncertainty is also computed iteratively as derived in [2] and [109]:

$$\mathrm{cov}\big(\Delta\mathbf{r}_{t_1}^{t_{v_{(i+1)}}}\big) = \mathbf{A}_i \mathrm{cov}\big(\Delta\mathbf{r}_{t_1}^{t_{v_{(i)}}}\big)\mathbf{A}_i^\top + \mathbf{B}_i \mathrm{cov}\big(\tilde{\boldsymbol{\omega}}^*(t_{v_i})\big)\mathbf{B}_i^\top, \tag{3.43}$$

with $\mathrm{cov}\big(\Delta\mathbf{r}_{t_1}^{t_{v_{(1)}}}\big) = \mathbf{0}_{3\times 3}$,

$$\mathbf{A}_i = \exp\Big(\big((\tilde{\boldsymbol{\omega}}^*(t_{v_i}) - \bar{\mathbf{b}}_\omega(t_{v_i}))(t_{v_{i+1}} - t_{v_i})\big)^\wedge\Big)^\top \tag{3.44}$$

$$\mathbf{B}_i = \mathbf{J}_r\Big((\tilde{\boldsymbol{\omega}}^*(t_{v_i}) - \bar{\mathbf{b}}_\omega(t_{v_i}))(t_{v_{i+1}} - t_{v_i})\Big)(t_{v_{i+1}} - t_{v_i}), \tag{3.45}$$

and $\mathbf{J}_r$ the right Jacobian of SO(3)

$$\mathbf{J}_r(\boldsymbol{\phi}) = \mathbf{I} - \frac{1-\cos(\|\boldsymbol{\phi}\|)}{\|\boldsymbol{\phi}\|^2}\boldsymbol{\phi}^\wedge + \frac{\|\boldsymbol{\phi}\|-\sin(\|\boldsymbol{\phi}\|)}{\|\boldsymbol{\phi}\|^3}(\boldsymbol{\phi}^\wedge)^2. \tag{3.46}$$

3.4.2 GPM - Velocity and position

In the case of noiseless measurements and perfect knowledge of the gyroscope biases, $\mathbf{R}^\bullet_{t_1}(\bullet) = \Delta\mathbf{R}^\bullet_{t_1}(\bullet)$. Consequently, velocity and position preintegrated measurements are functions of both the gyroscope and accelerometer readings. Unfortunately, the preintegrated measurements $\Delta\mathbf{v}^{t_2}_{t_1}$ and $\Delta\mathbf{p}^{t_2}_{t_1}$ cannot be expressed analytically solely based on linear operators and the independent GP models of raw inertial measurements. To overcome this issue and allow for the direct (non-iterative) GP inference of velocity and position preintegrated measurements, the proposed method models the accelerometer data after their reprojection in $\mathfrak{F}^{t_1}_I$, the inertial frame at t_1, according to the rotational preintegrated measurements. Thus, modelling these new signals with GP models as

$$\big(\Delta\mathbf{R}^t_{t_1}(t)(\tilde{\mathbf{f}}(t) - \bar{\mathbf{b}}_f(t))\big)_j \sim \mathcal{GP}\big(0, k_{f_j}(t,t')\big), \tag{3.47}$$

and rewriting (3.17) and (3.18) as

$$\Delta\mathbf{v}^{t_2}_{t_1} = \begin{bmatrix}\Delta v_1{}^{t_2}_{t_1}\\ \Delta v_2{}^{t_2}_{t_1}\\ \Delta v_3{}^{t_2}_{t_1}\end{bmatrix} = \begin{bmatrix}\mathcal{L}^t_v\big(\Delta\mathbf{R}^t_{t_1}(t)(\tilde{\mathbf{f}}(t) - \bar{\mathbf{b}}_f(t))\big)_1\\ \mathcal{L}^t_v\big(\Delta\mathbf{R}^t_{t_1}(t)(\tilde{\mathbf{f}}(t) - \bar{\mathbf{b}}_f(t))\big)_2\\ \mathcal{L}^t_v\big(\Delta\mathbf{R}^t_{t_1}(t)(\tilde{\mathbf{f}}(t) - \bar{\mathbf{b}}_f(t))\big)_3\end{bmatrix} \text{ and} \tag{3.48}$$

$$\Delta\mathbf{p}^{t_2}_{t_1} = \begin{bmatrix}\Delta p_1{}^{t_2}_{t_1}\\ \Delta p_2{}^{t_2}_{t_1}\\ \Delta p_3{}^{t_2}_{t_1}\end{bmatrix} = \begin{bmatrix}\mathcal{L}^t_p\big(\Delta\mathbf{R}^t_{t_1}(t)(\tilde{\mathbf{f}}(t) - \bar{\mathbf{b}}_f(t))\big)_1\\ \mathcal{L}^t_p\big(\Delta\mathbf{R}^t_{t_1}(t)(\tilde{\mathbf{f}}(t) - \bar{\mathbf{b}}_f(t))\big)_2\\ \mathcal{L}^t_p\big(\Delta\mathbf{R}^t_{t_1}(t)(\tilde{\mathbf{f}}(t) - \bar{\mathbf{b}}_f(t))\big)_3\end{bmatrix}, \tag{3.49}$$

each component of $\Delta\mathbf{v}^{t_2}_{t_1}$ and $\Delta\mathbf{p}^{t_2}_{t_1}$ is inferred independently using (3.36):

$$\Delta v_j{}^{t_2*}_{t_1} = \mathcal{L}^t_v\mathbf{k}_{f_j}(t,\mathfrak{t})\big[\mathbf{K}_{f_j}(\mathfrak{t},\mathfrak{t}) + \sigma^2_{f_j}\mathbf{I}\big]^{-1}\mathbf{f}_j, \tag{3.50}$$

$$\mathrm{var}(\Delta v_j{}^{t_2*}_{t_1}) = \mathcal{L}^t_v k_{f_j}(t,t)\mathcal{L}^t_v - \mathcal{L}^t_v\mathbf{k}_{f_j}(t,\mathfrak{t})\big[\mathbf{K}_{f_j}(\mathfrak{t},\mathfrak{t}) + \sigma^2_{f_j}\mathbf{I}\big]^{-1}\mathbf{k}_{f_j}(\mathfrak{t},t)\mathcal{L}^t_v, \tag{3.51}$$

and

$$\Delta p_j{}^{t_2*}_{t_1} = \mathcal{L}^t_p \mathbf{k}_{f_j}(t, \mathfrak{t}) \big[\mathbf{K}_{f_j}(\mathfrak{t}, \mathfrak{t}) + \sigma^2_{f_j}\mathbf{I}\big]^{-1} \mathbf{f}_j, \tag{3.52}$$

$$\mathrm{var}(\Delta p_j{}^{t_2*}_{t_1}) = \mathcal{L}^t_p k_{f_j}(t,t)\mathcal{L}^t_p - \mathcal{L}^t_p \mathbf{k}_{f_j}(t, \mathfrak{t}) \big[\mathbf{K}_{f_j}(\mathfrak{t}, \mathfrak{t}) + \sigma^2_{f_j}\mathbf{I}\big]^{-1} \mathbf{k}_{f_j}(\mathfrak{t}, t)\mathcal{L}^t_p. \tag{3.53}$$

The vector $\mathbf{f}_j$ consists of the set of training values $\big(\Delta\mathbf{R}^{\mathfrak{t}}_{t_1}(t)(\tilde{\mathbf{f}}(\mathfrak{t}) - \bar{\mathbf{b}}_f(\mathfrak{t}))\big)_j$ at $t = \mathfrak{t}_i$ for every $\mathfrak{t}_i$ present in the training data.

3.5 Postintegration bias and inter-sensor time-shift corrections

By itself, the integration of inertial data is prone to large drift. In addition to Gaussian noise and the actual physical values of acceleration or angular velocity, the raw inertial measurements contain a generally unknown bias component. In the context of sensor fusion, and under some specific observability conditions [110], these biases can be estimated. For that purpose, the authors of the pioneer paper on preintegration [1] introduced a first-order expansion to correct biases after integration under the assumption of constant biases during the integration interval. This work extends that concept to integrate the estimation of inter-sensor time-shift δ_t. The preintegrated measurements (3.16), (3.17) and (3.18) can be rewritten as

$$\Delta\mathbf{R}^{t_2}_{t_1}(\mathbf{b}_\omega, \delta_t) \approx \Delta\mathbf{R}^{t_2}_{t_1}(\bar{\mathbf{b}}_\omega, \bar{\delta}_t) \exp\left(\left(\frac{\partial \Delta\mathbf{r}^{t_2}_{t_1}}{\partial \bar{\mathbf{b}}_\omega}\hat{\mathbf{b}}_\omega + \frac{\partial \Delta\mathbf{r}^{t_2}_{t_1}}{\partial \delta_t}\hat{\delta}_t\right)^\wedge\right), \tag{3.54}$$

$$\Delta\mathbf{v}^{t_2}_{t_1}(\mathbf{b}_f, \mathbf{b}_\omega, \delta_t) \approx \Delta\mathbf{v}^{t_2}_{t_1}(\bar{\mathbf{b}}_f, \bar{\mathbf{b}}_\omega, \bar{\delta}_t) + \frac{\partial \Delta\mathbf{v}^{t_2}_{t_1}}{\partial \mathbf{b}_f}\hat{\mathbf{b}}_f + \frac{\partial \Delta\mathbf{v}^{t_2}_{t_1}}{\partial \bar{\mathbf{b}}_\omega}\hat{\mathbf{b}}_\omega + \frac{\partial \Delta\mathbf{v}^{t_2}_{t_1}}{\partial \delta_t}\hat{\delta}_t, \tag{3.55}$$

$$\Delta\mathbf{p}^{t_2}_{t_1}(\mathbf{b}_f, \mathbf{b}_\omega, \delta_t) \approx \Delta\mathbf{p}^{t_2}_{t_1}(\bar{\mathbf{b}}_f, \bar{\mathbf{b}}_\omega, \bar{\delta}_t) + \frac{\partial \Delta\mathbf{p}^{t_2}_{t_1}}{\partial \mathbf{b}_f}\hat{\mathbf{b}}_f + \frac{\partial \Delta\mathbf{p}^{t_2}_{t_1}}{\partial \bar{\mathbf{b}}_\omega}\hat{\mathbf{b}}_\omega + \frac{\partial \Delta\mathbf{p}^{t_2}_{t_1}}{\partial \delta_t}\hat{\delta}_t, \tag{3.56}$$

with $\mathbf{b}_f = \bar{\mathbf{b}}_f + \hat{\mathbf{b}}_f$, $\mathbf{b}_\omega = \bar{\mathbf{b}}_\omega + \hat{\mathbf{b}}_\omega$, and $\delta_t = \bar{\delta}_t + \hat{\delta}_t$. Note that $\bar{\bullet}$ denotes the prior knowledge of the value at the time of preintegration and $\hat{\bullet}$ represents the correction.

The rest of this section describes how the different Jacobians in (3.54), (3.55), and (3.56) are computed according to the GPM formulation.

3.5.1 Rotation GPM Jacobians

Similarly to the GPMs calculation, the computation of the Jacobians for the rotational preintegrated measurements is split into two scenarios. If the motion contains multi-axis rotations, the Jacobian of $\Delta\mathbf{R}_{t_1}^{t_2}$ with respect to the gyroscope biases $\mathbf{b}_\omega$ is computed iteratively as in [2] and [109]:

$$\frac{\partial \Delta\mathbf{r}_{t_1}^{t_{v_{i+1}}}}{\partial \mathbf{b}_\omega} = \mathbf{A}_i \frac{\partial \Delta\mathbf{r}_{t_1}^{t_{v_i}}}{\partial \mathbf{b}_\omega} + \mathbf{B}_i, \tag{3.57}$$

with $\mathbf{A}_i$ and $\mathbf{B}_i$ defined as in (3.44) and (3.45), and $\frac{\partial \Delta\mathbf{r}_{t_1}^{t_{v_1}}}{\partial \mathbf{b}_\omega} = \mathbf{I}_{3\times 3}$.

The Jacobian of $\Delta\mathbf{R}_{t_1}^{t_2}$ with respect to the inter-sensor time-shift δ_t can easily be computed numerically by offsetting the integration limits of $\mathcal{L}_r^t$. The following subsections detail the computation of these Jacobians in the case of 1-axis rotations.

3.5.1.1 Gyroscope biases

In the inference of $\Delta {r_j}_{t_1}^{t_2}$, as per (3.40), the biases only impact the training values $\boldsymbol{\omega}_j$. Consequently,

$$\frac{\partial \Delta {r_j}_{t_1}^{t_2}}{\partial \mathbf{b}_\omega} = \mathcal{L}_r^t k_{r_j}(t,\mathfrak{t})\big[k_{r_j}(\mathfrak{t},\mathfrak{t}) + \sigma_{\omega_j}^2 \mathbf{I}\big]^{-1} \frac{\partial \boldsymbol{\omega}_j}{\partial \mathbf{b}_\omega}. \tag{3.58}$$

The Jacobian of $\boldsymbol{\omega}_j$ with respect to $\mathbf{b}_\omega$ is a column vector of 1 (a simple additive bias).

3.5.1.2 Inter-sensor time-shift

The Jacobian of $\Delta r_j{}_{t_1}^{t_2}$ with respect to δ_t (the partial derivative of (3.38) with respect to t_1) can be written using another linear operator:

$$\frac{\partial \Delta \mathbf{r}_{t_1}^{t_2}}{\partial \delta_t} = \frac{\partial}{\partial t_1} \int_{t_1}^{t_1+\Delta t} (\tilde{\boldsymbol{\omega}}(t) - \bar{\mathbf{b}}_{\omega}(t)) dt = \begin{bmatrix} \mathcal{L}_{r_{\delta_t}}^t (\omega_1(t) - \bar{b}_{\omega_1}(t)) \\ \mathcal{L}_{r_{\delta_t}}^t (\omega_2(t) - \bar{b}_{\omega_2}(t)) \\ \mathcal{L}_{r_{\delta_t}}^t (\omega_3(t) - \bar{b}_{\omega_3}(t)) \end{bmatrix}, \tag{3.59}$$

with $\Delta t = t_2 - t_1$. Therefore the Jacobian can be computed with (3.40) replacing $\mathcal{L}_r^t$ with $\mathcal{L}_{r_{\delta_t}}^t$.

3.5.2 Velocity and position GPM Jacobians

The derivation of the Jacobians in (3.55) and (3.56) is similar for both the velocity and position cases.

3.5.2.1 Accelerometer and gyroscope biases

As for $\Delta \mathbf{R}_{t_1}^{t_2}$, the inference of $\Delta \mathbf{v}_{t_1}^{t_2}$ and $\Delta \mathbf{p}_{t_1}^{t_2}$ depends on the IMU biases solely via their training values $\mathbf{f}_j$. Therefore,

$$\frac{\partial \Delta v_j{}_{t_1}^{t_2}}{\partial b_\bullet} = \mathcal{L}_v^t \mathbf{k}_{r_j}(t, \mathfrak{t}) \big[\mathbf{K}_{r_j}(\mathfrak{t}, \mathfrak{t}) + \sigma_{\omega_j}^2 \mathbf{I}\big]^{-1} \frac{\partial \mathbf{f}_j}{\partial b_\bullet} \tag{3.60}$$

$$\frac{\partial \Delta p_j{}_{t_1}^{t_2}}{\partial b_\bullet} = \mathcal{L}_p^t \mathbf{k}_{r_j}(t, \mathfrak{t}) \big[\mathbf{K}_{r_j}(\mathfrak{t}, \mathfrak{t}) + \sigma_{\omega_j}^2 \mathbf{I}\big]^{-1} \frac{\partial \mathbf{f}_j}{\partial b_\bullet} \tag{3.61}$$

with $b_\bullet$ being either the accelerometer biases $\mathbf{b}_f$ or the gyroscope biases $\mathbf{b}_\omega$. Note that $\mathbf{f}_j$ depends on $\mathbf{b}_\omega$ as $\Delta \mathbf{R}_{t_1}^t(t)$ depends on $\mathbf{b}_\omega$. The derivations to obtain $\frac{\partial \mathbf{f}_j}{\partial \mathbf{b}_f}$ and $\frac{\partial \mathbf{f}_j}{\partial \mathbf{b}_\omega}$ are depicted in Appendix B.

3.5.2.2 Inter-sensor time-shift

The differentiation of $\Delta\mathbf{v}_{t_1}^{t_2}$ and $\Delta\mathbf{p}_{t_1}^{t_2}$ with respect to δ_t cannot be directly inferred with the help of a linear operator as in the case of $\Delta\mathbf{r}_{t_1}^{t_2}$. The main difference comes from the fact that $\mathbf{f}_j$ depends on δ_t as $\Delta\mathbf{R}_{t_1}^{t}(t)$ depends on δ_t. Consequently the Jacobian of $\Delta\mathbf{v}_{t_1}^{t_2}$ is computed as

$$\begin{aligned}\frac{\partial \Delta v_j{}_{t_1}^{t_2}}{\partial \delta_t} =& \mathcal{L}_{v_{\delta_t}}^{t} k_{r_j}(t,\mathfrak{t})\big[k_{r_j}(\mathfrak{t},\mathfrak{t}) + \sigma_{\omega_j}^2\mathbf{I}\big]^{-1}\mathbf{f}_j \\ &+ \mathcal{L}_{v}^{t} k_{r_j}(t,\mathfrak{t})\big[k_{r_j}(\mathfrak{t},\mathfrak{t}) + \sigma_{\omega_j}^2\mathbf{I}\big]^{-1}\frac{\partial \mathbf{f}_j}{\partial \delta_t}\end{aligned} \tag{3.62}$$

with

$$\mathcal{L}_{v_{\delta_t}}^{t} = \frac{\partial}{\partial t_1}\mathcal{L}_{v}^{t} = \mathcal{L}_{d}^{t_1}\mathcal{L}_{v}^{t}. \tag{3.63}$$

The computation of the Jacobian of $\Delta\mathbf{p}_{t_1}^{t_2}$ with respect to δ_t follows (3.62) replacing $\mathcal{L}_{v_{\delta_t}}$ with

$$\mathcal{L}_{p_{\delta_t}}^{t} = \frac{\partial}{\partial t_1}\mathcal{L}_{p}^{t} = \mathcal{L}_{d}^{t_1}\mathcal{L}_{p}^{t}. \tag{3.64}$$

Note that the Jacobians of $\mathbf{f}_j$ with respect to δ_t can be computed numerically or deduced from $\frac{\partial \Delta r_j{}_{t_1}^{t}(t)}{\partial \delta_t}$ computed in the previous subsection.

3.6 Experiments and results

This section presents quantitative evaluations of the proposed method on simulated data in the context of low and high-frequency inferences. As examples of applications, one can assimilate the low-frequency scenario to VIO-like systems where the inertial information is needed between frames or keyframes. The high-frequency scenario corresponds to the need of trajectory characterisation in systems containing high-frequency sensors such as LiDARs or event cameras. In the following chapters of this thesis, the GPMs will be

applied to inertial-aided localisation and mapping systems and validated with real-world experiments. Our implementation uses the square exponential kernel parameterised by a length scale and the signal's variance. Learning the length scale directly from the inertial data allows for the GP models to adapt their smoothness to the actual shape of the signals at hand.

3.6.1 Low-frequency benchmarks (0.2 - 20 Hz)

In this subsection, we aim to evaluate the performance of the GPM method. We compare it against our work [19] (UPM, also shown in Appendix A for completeness), and the original on-manifold preintegration method [2] (Standard Preintegrated Measurement (PM)). The evaluation consists of generating random trajectories from sinusoidal functions and computing the preintegrated measurements using each of the three aforementioned methods between randomly chosen timestamps (t_1 and t_2). The frequencies of the sinusoidal functions range between 0.05 and 0.4 Hz for the position, and between 0.15 and 0.7 Hz for the rotation. The different metrics for evaluation are presented below.

3.6.1.1 Accuracy

This set of experiments have been designed to simulate the VIO scenarios where preintegration is performed at low frequency to constrain the system pose between consecutive keyframes (set-up one) or for tracking feature from frame to frame (set-up two). Our first set-up computes preintegrated measurements over durations ranging anywhere between 1 and 5 seconds. As the trajectories have different characteristics, and the integration interval length is not fixed across runs, the chosen metric is the relative error with respect to the travelled (linear or angular) distance. The second experiment has been designed around smaller integration intervals. We compute the absolute pose error for different fixed inference frequencies between 1 and 20 Hz. Based on Table 3.1 and 3.2, GPMs and UPMs outperform PMs by around an order of magnitude. In other words, the assumption of constant acceleration during the IMU period introduces a non-negligible integration noise. UPMs upsample the inertial measurements (here by a factor 10) before conducting numerical integration. It reduces the integration noise significantly. The amount of

1-axis rotations (relative errors %)						
Motion	PM [2]		UPM [19]		GPM (proposed)	
	Rot er.	Pos er.	Rot er.	Pos er.	Rot er.	Pos er.
Slow	0.253	3.54	0.038	0.381	**0.028**	**0.143**
Fast	0.221	2.90	0.024	0.301	**0.008**	**0.073**

Multi-axis rotations (relative errors %)						
Motion	PM [2]		UPM [19]		GPM (proposed)	
	Rot er.	Pos er.	Rot er.	Pos er.	Rot er.	Pos er.
Slow	0.316	4.79	**0.035**	0.493	**0.035**	**0.311**
Fast	0.308	4.35	**0.031**	0.433	**0.031**	**0.396**

TABLE 3.1: Average relative error of preintegrated measurements with respect to trajectory length in simulated environments (over 100 trials) for different types of motion. The integration interval length is between 1s and 5s. Characteristics of the trajectories during integration interval (1D rot. slow/1D rot. fast/3D rot. slow/3D rot. fast): avg. distance = 9.7/15/8.8/23 m, avg. velocity = 1.8/3.9/1.9/4.7 m/s , avg angular distance = 2.2/6.8/3.8/16 rad, avg angular velocity = 0.7/2.5/1.3/4.9 rad/s. IMU frequency 100 Hz, accelerometer noise sd 0.02 m/s^2, gyroscope noise sd 0.002 rad/s, UPM upsampled frequency 1 kHz.

integration noise in UPMs directly depends on the upsampled frequency. This creates a trade-off between accuracy and computation time. GPMs do not suffer from the integration noise thanks to their analytical approach to the integration. The factor that limits the accuracy of the GPMs is the ability for the GPs to model the true underlying signal accurately.

3.6.1.2 Robustness to noise

This set-up evaluates the three methods for different variations of noise. Figure 3.3 plots the corresponding relative errors. For motion containing only 1-axis rotations, the absence of noise pushes the GPMs error towards zero where UPMs and PMs cannot go below a certain error. This observation supports the hypothesis that GPMs are not subject to integration noise. The final error is mainly driven by the kernel ability to model noisy signals. In the presence of multi-axis rotations, UPMs and GPMs share the same numerical approach to estimate the rotational part of the preintegrated measurements. For that reason, the error difference between the two methods is smaller, but GPMs still outperform UPMs.

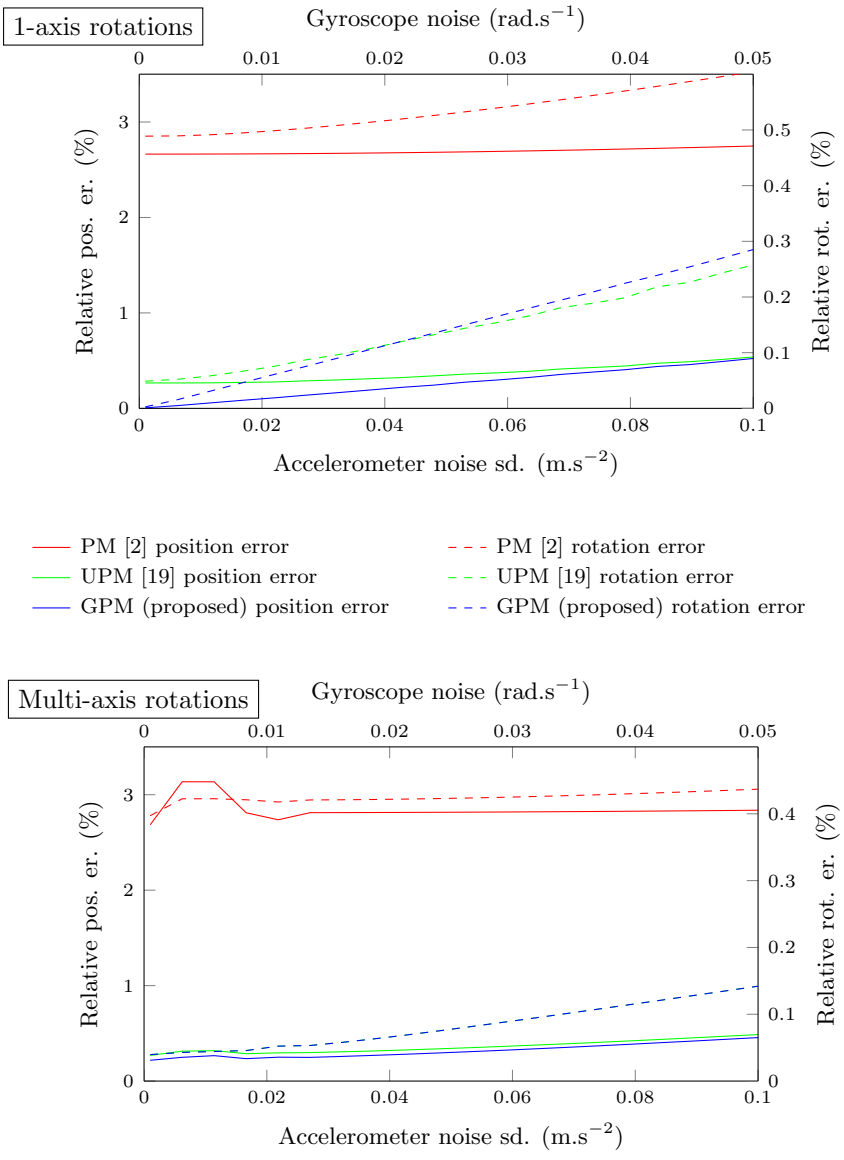


FIGURE 3.3: Average relative error of preintegrated measurements with respect to trajectory length in simulated environments (over 50 trials) for different levels of IMU noise. All the plots in one graph are subject to the bottom and top axis (in other words, the gyroscope and accelerometer noises are simultaneously changing along the x axis). The position error (pos er.) plots relate to the left axis and the rotation error (rot er.) relate to the right axis. The integration interval lengths are between 1 s and 2 s. IMU frequency 100 Hz, UPM upsampled frequency 1 kHz.

1-axis rotations (absolute errors in mrad and mm)						
Query	PM [2]		UPM [19]		GPM (proposed)	
rate (Hz)	Rot er.	Pos er.	Rot er.	Pos	Rot er.	Pos er.
20	2.41	5.61	0.28	0.60	**0.10**	**0.02**
10	4.91	12.1	1.61	1.32	**1.40**	**0.16**
2	21.3	61.3	3.09	7.19	**0.20**	**0.61**
1	24.9	125	2.51	12.6	**0.28**	**1.80**

Multi-axis rotations (absolute errors in rad and mm)						
Query	PM [2]		UPM [19]		GPM (proposed)	
rate (Hz)	Rot er.	Pos er.	Rot er.	Pos	Rot er.	Pos er.
20	5.48	6.06	**0.80**	0.65	**0.80**	**0.02**
10	11.1	13.1	7.53	2.27	**7.18**	**0.48**
2	35.6	65.0	4.72	7.29	**3.56**	**1.74**
1	37.5	159	**3.75**	15.9	**3.75**	**10.3**

TABLE 3.2: Average absolute error of preintegrated measurements for fast trajectories in simulated environments (over 100 trials) for different fixed query rates.

3.6.1.3 Computation time

While providing more accurate estimates, UPMs and GPMs can be significantly slower than the original preintegration. This experiment collects the average run time (over 50 runs) of the benchmarked methods. The code is executed on a laptop equipped with an Intel i5-6300U Central Processing Unit (CPU) working at 2.40 GHz, and 24 GiB of RAM. Table 3.3 presents the values obtained for different length of integration interval (from 10 Hz to 0.2 Hz). The total execution time of both the UPMs and GPMs can be divided into two parts: hyperparameter training and inference. With the square exponential kernel, descent hyperparameters can be deduced from the sensor characteristics and a vague knowledge of the motion aggressiveness. In the absence of prior knowledge about the signals or when using complex kernels, the hyperparameter training step is highly recommended. Nonetheless, for signals that keep consistent smoothness characteristics over time, the learnt hyperparameters can be safely reused from one integration interval to another. In such a case, the UPM and GPM computations comply with real-time constraints as the inference time stays under the integration interval length. The difference of computation time between UPMs and GPMs is explained by the larger number of inferences, and the iterative numerical integration carried out to generate UPMs. While

PMs and UPMs execution times are consistent for the different motion types (1-axis and multi-axis rotations), the GPMs are significantly slower when computed over movements that contain multi-axis rotations. As pointed out before, this is due to the numerical approach needed to estimate $\boldsymbol{\Delta}\mathbf{R}$.

The results of this experiment show that for integration intervals of few seconds, GPMs can be used in close to real-time operations. Furthermore, one should note that these values, for all of the three methods, are generated from single-thread Matlab code that is not optimised for high performance. The GPM computation can easily be parallelised on 3 CPU cores as the hyperparameter training and inferences are independent on each axis.

The training data covariance matrices $\mathbf{K}_{\bullet}(\mathrm{t},\mathrm{t})$, presented above, solely depends on the kernel hyperparameters, the number of training samples used for the integration interval, and the relative temporal position of these samples. In many practical scenarios, the IMU readings are generated at a fixed frequency, and the integration interval stays the same all along the estimation pipeline. When hyper-parameter training is not required, these conditions can be leveraged to speed-up the UPMs and GPMs inference time by pre-computing the matrices $\left[\mathbf{K}_{\bullet}(\mathrm{t},\mathrm{t}) + \sigma_{\star}^2\mathbf{I}\right]^{-1}$. Consequently, the inference complexity is reduced from $\mathcal{O}(n^3)$ to $\mathcal{O}(n^2)$ as matrix inversion is no longer needed.

3.6.2 High-frequency benchmarks ($> 100\,\mathrm{kHz}$)

The VIO-like scenario of the previous subsection aims at combining numerous IMU measurements into a single set of preintegrated measurements (rotation, velocity, and position) over a given integration window. In other words, the frequency of the preintegrated measurements that are generated is smaller than the IMU frequency. In the present subsection, we benchmark the use of GPMs in the context of sensor-fusion with high-frequency sensors. Unlike above, here, the objective is to create preintegrated measurements to characterise the system's motion at the acquisition times of a higher frequency sensor. For example, in Chapter 4, a 300 kHz LiDAR (rotation speed of 10 Hz) is coupled with a 100 Hz IMU. The generation of GPMs for each LiDAR point allows accurate undistortion of the LiDAR scans.

1-axis rotations (computation time ms)					
Interval length (s)	PM [2]	UPM [19]		GPM (proposed)	
		Training	Inference	Training	Inference
0.05	0.8	294	18.7	356	1.3
0.1	0.9	335	25.6	399	1.8
0.5	3.1	392	67.3	396	4.2
1	5.8	498	113	502	8.1
2	11.6	1088	279	1085	28.2
3	18.6	2349	534	2332	68.9
4	23.3	4159	824	4144	127
5	28.5	6939	1244	6832	212

Multi-axis rotations (computation time ms)					
Interval length (s)	PM [2]	UPM [19]		GPM (proposed)	
		Training	Inference	Training	Inference
0.05	0.6	269	16.9	277	10.2
0.1	0.9	291	22.0	298	11.1
0.5	3.1	362	60.2	367	24.8
1	5.8	530	121	529	51.9
2	11.8	1148	297	1110	138
3	17.5	2347	538	2281	270
4	24.1	4352	875	4243	482
5	28.5	7295	1392	7102	817

TABLE 3.3: Average computation time of preintegrated measurements (over 50 trials) for different integration interval lengths. For the UPMs and GPMs, both the hyperparameter training time and the inference time are shown. IMU frequency 100 Hz, UPM upsampled frequency 1 kHz.

3.6.2.1 Accuracy

This set-up aims at comparing the accuracy of different preintegration methods for inference at 300 kHz in a time window of 0.3 s given IMU measurements at 100 Hz. In addition to the method benchmarked in Subsection 3.6.1, we introduce the Linear Preintegrated Measurements (LPMs) that follow the same pipeline as the GPM generation presented in Fig. 3.2, but using linear interpolation between IMU measurements instead of GP regression. We also distinguish two versions of the PMs. The first one, noted PM-IMU, corresponds to the computation of the preintegrated measurements for each IMU timestamp followed by the search and copy of the closest IMU-time preintegrated measurements for each of the 300 kHz query timestamps. The second one, noted PM-HF, corresponds to the

1-axis rotations		
Method	Rotation error (mrad)	Position error (mm)
PM-IMU [2]	23.5 ± 14	12.3 ± 3.86
PM-HF	5.36 ± 3.9	1.59 ± 1.5
LPM (linear)	0.11 ± 0.035	0.16 ± 0.062
UPM [19]	$\mathbf{0.073 \pm 0.028}$	$\mathbf{0.12 \pm 0.050}$
GPM (proposed)	$\mathbf{0.073 \pm 0.028}$	$\mathbf{0.12 \pm 0.051}$

Multi-axis rotations		
Method	Rotation error (mrad)	Position error (mm)
PM-IMU [2]	20.9 ± 6.2	13.6 ± 4.70
PM-HF	4.96 ± 1.7	2.8 ± 1.28
LPM (linear)	0.119 ± 0.035	0.15 ± 0.066
UPM [19]	$\mathbf{0.102 \pm 0.032}$	0.14 ± 0.068
GPM (proposed)	$\mathbf{0.102 \pm 0.032}$	$\mathbf{0.13 \pm 0.062}$

TABLE 3.4: Average pose RMSE of preintegrated measurements for fast trajectories in simulated environments (over 100 trials) for a 300 kHz query rate over a time window of 0.3 s given IMU measurements at 100 Hz.

computation of the preintegrated measurements for each of the 300 kHz query timestamps based on the constant-measurement assumption in [2]. It is equivalent to the computation of the UPMs using a zero-order-hold interpolation instead of GP regression.

The results presented in Table 3.4 show that the GPMs and UPMs have very similar performances in this high-frequency scenario, both for motion with 1-axis and multi-axis rotations. This shrink in accuracy gap is a consequence of the very high upsampling frequency done when computing the UPMs (300 Hz). At such frequencies, the numerical integration noise of the UPMs become extremely small. The original preintegration method, whether it is via PM-IMU or PM-HF, underperforms other techniques by at least a factor 20. Note that, while displaying inferior accuracy performance, the LPMs show precision in the same order of magnitude as the GPMs and UPMs.

3.6.2.2 Computation time

In this subsection, we discuss the computation time of the previous set-up. The 300 kHz preintegrated-measurement frequency over a time window of 0.3 s corresponds to 90 k

Computation time in seconds for 90 k inference operations		
Method	Motion type	
	1-axis rotations	Multi-axis rotations
PM-IMU [2]	0.059 ± 0.009	0.057 ± 0.002
PM-HF	6.33 ± 0.48	6.38 ± 0.10
LPM (linear)	2.51 ± 0.19	3.67 ± 0.15
UPM [19]	8.26 ± 0.72	8.18 ± 0.17
GPM (proposed)	5.74 ± 0.57	6.56 ± 0.21

TABLE 3.5: Average (over 100 trials) computation time for 90 k inferrences over a time window of 0.3 s given IMU measurements at 100 Hz.

inference operations. Table 3.5 reports the computation time required by the different benchmarked methods to generate this large amount of preintegrated measurements.

It shows that the GPMs outperform the UPMs not only in terms of accuracy but also computation-wise. As in the low-frequency scenario, the standard preintegration [2], while less accurate, offers faster computation. For time-constrained systems, the LPMs offer a decent compromise between accuracy and execution time.

Here, the evaluations are based on single-threaded Matlab code. The multi-axis-case C++ implementation used in the following chapters is not optimised for computation time, but leverages a parallelisation over three CPU-cores. The GPM inference time is reduced by around a factor three (≈ 2 s) when the LPM computation time is divided by approximately twenty-four (≈ 0.15 s). While a Graphic Processing Unit (GPU) implementation of the GPM could match real-time constraints for high-frequency inference, one can consider using LPMs for CPU-based real-time applications at the expense of a slight loss of accuracy.

3.7 Conclusion

This chapter presents a novel theory for preintegration based on a continuous and probabilistic representation of inertial data. The proposed preintegrated measurements, called GPMs, leverage GP models of the IMU measurements and apply linear operators to the covariance kernels to analytically infer the integrals of the signals over any time interval. In combination with a first-order expansion for postintegration bias and inter-sensor time-shift corrections, GPMs are especially suited for inertial-aided estimation frameworks.

Our experiments show that GPMs outperform both UPMs [19] and PMs [2] in terms of accuracy. The GPMs calculation suffers from the cubic computational complexity of GPs but is still suitable for close-to-real-time operations in the context of low-frequency inference operations (like in VIO frameworks). In the following chapters of this thesis, we propose two frameworks designed to leverage GPMs: a lidar-inertial localisation and mapping framework (IN2LAAMA), and an event-based odometry method (IDOL).

On the theory side, future work includes relaxing the assumption of constant bias during preintegration and developing an analytical approximation for the integration of gyroscope measurements in the multi-axis-rotation case. In terms of the current implementation, for real-time operations, integration intervals above approximately 0.5 s (actual duration subject to computer performance and code optimisation) do not allow the hyperparameter training from scratch for each integration interval. While one can use heuristics, pretraining or sporadic training (every N integration intervals), a more appealing approach is the use of filtering methods to update the hyperparameters online. Thus, the computation cost per integration interval is substantially reduced, while the hyperparameters are tracked over time. Additionally, the use of GPU can significantly reduce the computation time. In the context of low-frequency query, some matrix operations as well as the computation of the kernel matrices and vectors' components can be highly parallelised. For the high-frequency inference, the parallelisation can be implemented at a higher level as each inference can be performed independently.

Chapter 4

IN2LAAMA: INertial Lidar Localisation Autocalibration And MApping

4.1 Introduction

This chapter presents a LiDAR-inertial framework for simultaneous localisation, mapping, and inter-sensor extrinsic calibration. This framework is called IN2LAAMA and leverages the GPMs introduced in the previous chapter. This work corresponds to the contributions presented in [111].

Unlike global-shutter cameras that take snapshots of the environment, most of today's LiDARs collect range information by sweeping the surrounding space with one or multiple lasers. A popular LiDAR design, such as the iconic *Velodyne HDL-64* (Fig. 4.1), consists of stacking N lasers at different inclination angles, and making them spin around a fixed axis. Commonly, the measurements collected during each revolution are grouped into 3D point clouds denoted *scans*.

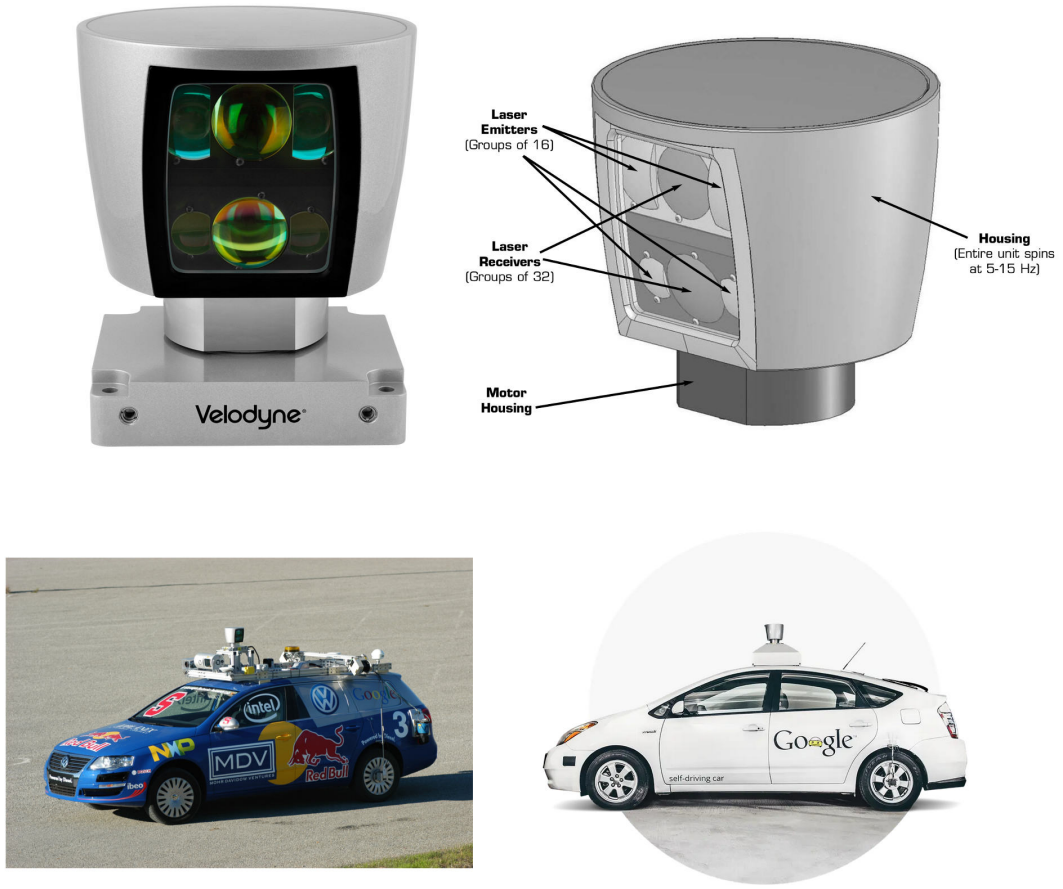


FIGURE 4.1: Example of rotative LiDAR: Velodyne HDL-64. Top row: photo of the sensor and schematic with the different major components (image source: velodynelidar.com). Bottom row: on the right is one of the first application of the Velodyne HDL-64 during the DARPA Grand Challenge 2007 (image source: cs.stanford.edu/group/roadrunner/). On the right the sensor mounted on the "Google car" (erroneously) considered as being the first self-driving car by the general public (image source: waymo.com).

Regardless of the sweeping mechanism, be it mechanical, optical or electronic, the time needed to scan the environment is not instantaneous. Consequently, each individual measurement of a scan is collected at a different time, and therefore from a different position if the system is moving. Without accurate knowledge of the sensor's pose at each of the points' timestamp, the individual 3D points cannot be registered precisely, resulting in scans that are affected by *motion distortion* (cf. Fig. 1.3).

In the context of odometry and mapping, traditionally, a system's pose is estimated at discrete timestamps corresponding to the sensor's sampling times. Such a paradigm becomes intractable when considering LiDAR points as independent measurements (bandwidth over

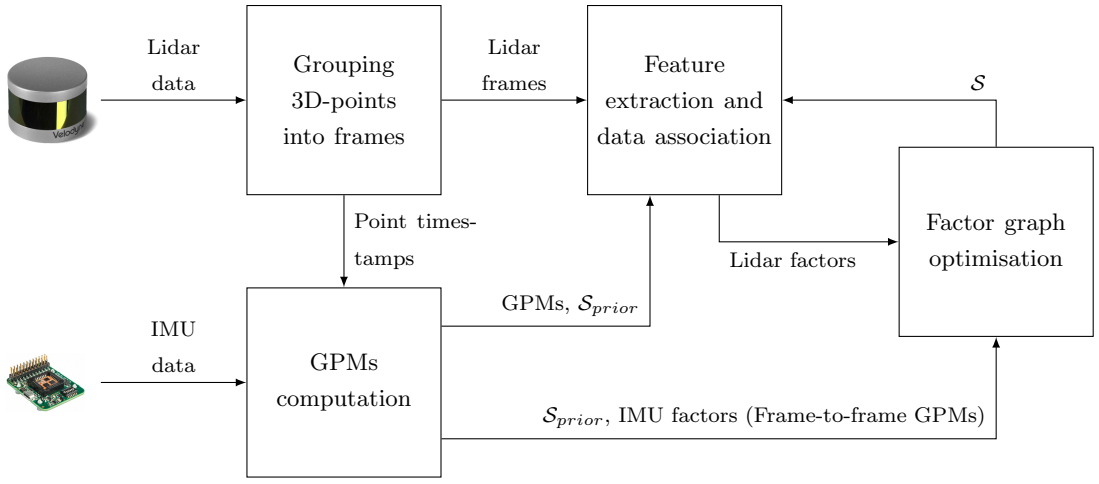


FIGURE 4.2: Overview of IN2LAAMA with $\mathcal{S}$ the estimated state.

hundreds of kilohertz). In order to reduce the complexity of the estimation problem, many methods in the literature assume a parametric motion model to describe the system's motion and estimate the model parameters. While allowing for computational efficiency, it generally leads to restrictive assumptions that do not necessarily represent reality in many real-world scenarios. The method presented in this chapter uses the inertial data in the form of GPMs to characterise the system's trajectory in a continuous manner while relying on a discrete state estimation at a low frequency. As the GPMs do not use any motion model, the proposed method does not make any assumptions about the sensors' dynamics.

Like most localisation and mapping frameworks, the proposed method is constituted of two main modules: a front-end for feature extraction and data association, and a back-end for state estimation through numerical optimisation. Nonetheless, IN2LAAMA differs from most frameworks as per the tight relationship between these two major modules. Reliable geometric feature extraction in LiDAR scans requires the knowledge of the system's trajectory to be unaffected by motion distortion. Accurate knowledge of the system's trajectory relies on the extraction and association of robust features. To address this "chicken-and-egg" problem, the features (front-end) are periodically recomputed according to the last state estimate (back-end), as shown in the block diagram of Fig. 4.2.

4.2 Method overview

4.2.1 Notation and definitions

Let us consider a rigidly mounted 3D LiDAR and a 6-DoF IMU. The LiDAR and IMU reference frames at time t_i are respectively noted $\mathfrak{F}_L^{t_i}$ and $\mathfrak{F}_I^{t_i}$. The rotation matrix $\mathbf{R}_I^L$ and the translation vector $\mathbf{p}_I^L$ characterise the pose of $\mathfrak{F}_L^{t_i}$ in $\mathfrak{F}_I^{t_i}$. Homogeneous transformation will be used for the rest this chapter, therefore rotation matrices and translations/positions will be associated with 4×4 transformation matrices with the same combination of subscripts and superscripts,

$$\mathbf{T}_a^b = \begin{bmatrix} \mathbf{R}_a^b & \mathbf{p}_a^b \\ \mathbf{0}^\top & 1 \end{bmatrix} \text{ and } \mathbf{T}_a^{b\,-1} = \begin{bmatrix} \mathbf{R}_a^{b\,\top} & -\mathbf{R}_a^{b\,\top}\mathbf{p}_a^b \\ \mathbf{0}^\top & 1 \end{bmatrix}. \tag{4.1}$$

The 3D-points $\mathbf{x}_L^i$ provided by the LiDAR at time t_i are projected from $\mathfrak{F}_L^{t_i}$ to $\mathfrak{F}_I^{t_i}$ using

$$\begin{bmatrix} \mathbf{x}_I^i \\ 1 \end{bmatrix} = \mathbf{T}_I^L \begin{bmatrix} \mathbf{x}_L^i \\ 1 \end{bmatrix}. \tag{4.2}$$

In this work, the LiDAR points are grouped into M frames. Note that in the proposed method, a frame corresponds to the data collected in scan greater than 360-degree, as explained in Section 4.4.3. The points that belong to the m^{th} frame form the set $\mathcal{X}^m$. $\mathcal{F}^m$ is a subset of $\mathcal{X}^m$ that represents LiDAR feature-points. A feature is a point belonging to a distinctive type of surface (e.g. plane or edge). The set of feature associations $\mathcal{A}$ contains tuples of 3 or 4 LiDAR feature-points depending on whether they are edges or planes respectively.

The 6-DoF-IMU is the combination of a 3-axis accelerometer and a 3-axis gyroscope. Therefore, the inertial data acquired consists of proper accelerations $\tilde{\mathbf{f}}_i$ and angular velocities $\tilde{\boldsymbol{\omega}}_i$ at time $\mathfrak{t}_i$ ($i = 1, \ldots, Q$). GPMs from Chapter 3 are used to infer inertial information for each of the individual LiDAR points.

The proposed method aims to estimate the IMU orientation $\mathbf{R}_W^{\tau_m}$, position $\mathbf{p}_W^{\tau_m}$ and velocity $\mathbf{v}_W^{\tau_m}$ for each LiDAR frame ($m = 0, \ldots, M$-1), as well as the IMU biases and the time-shifts

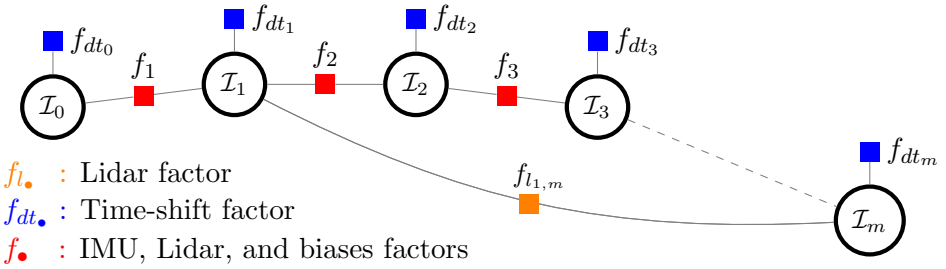


FIGURE 4.3: Factor graph representation of the optimisation problem solved in IN2LAAMA. $\mathcal{I}_m = \{\mathbf{R}_W^{\tau_m}, \mathbf{p}_W^{\tau_m}, \mathbf{v}_W^{\tau_m}, \hat{\mathbf{b}}_f^m, \hat{\mathbf{b}}_\omega^m, \hat{\delta}_{t_m}\}$ represents the IMU pose, velocity, biases and time-shift correction associated to the LiDAR scan $\mathcal{X}^m$ at τ_m. The factor $f_{l_{2,m}}$ represents a loop-closure.

between the two sensors. The subscript $_W$ represents the earth-fixed world reference frame $\mathfrak{F}_W$, and τ_m corresponds to the timestamp at the beginning of the m^{th} LiDAR frame. $\mathfrak{F}_\bullet^{\tau_m}$ refers to the reference frame of the IMU or LiDAR (as $\bullet$ represents in this case L or I) at time τ_m.

In the following, $\mathcal{S}$ indicates the state to be estimated: $\mathcal{S} = (\mathbf{R}_W^{\tau_0}, \cdots, \mathbf{R}_W^{\tau_{M\text{-}1}}, \mathbf{p}_W^{\tau_1}, \cdots, \mathbf{p}_W^{\tau_{M\text{-}1}}, \mathbf{v}_W^{\tau_0}, \cdots, \mathbf{v}_W^{\tau_{M\text{-}1}}, \hat{\mathbf{b}}_f^0, \cdots, \hat{\mathbf{b}}_f^{M\text{-}1}, \hat{\mathbf{b}}_\omega^0, \cdots, \hat{\mathbf{b}}_\omega^{M\text{-}1}, \hat{\delta}_{t_0}, \cdots, \hat{\delta}_{t_{M\text{-}1}})$ with $\hat{\mathbf{b}}_f^m$, $\hat{\mathbf{b}}_\omega^m$, and $\hat{\delta}_{t_m}$ the biases and time-shift corrections associated with the m^{th} LiDAR frame (more details about the biases and time-shift corrections are given in Section 4.3.2). In the case of extrinsic autocalibration, the calibration parameters $\mathbf{T}_I^L$ are also added to $\mathcal{S}$. The calibration procedure is explained in Section 4.5.2.

4.2.2 Cost function

The proposed method does not rely on any trajectory prior but uses a Gaussian distribution to constrain the inter-sensor time-shift. Therefore, the localisation and mapping problem is formulated as a Maximum A Posterior (MAP) estimation:

$$\mathcal{S}^* = \underset{\mathcal{S}}{\operatorname{argmin}} \; -\log(p(\mathcal{Z}|\mathcal{S})p(\mathcal{S})) = \underset{\mathcal{S}}{\operatorname{argmin}}\, C(\mathcal{S}), \tag{4.3}$$

with $\mathcal{Z}$ representing the available measurements and C the optimisation cost function.

Represented as the factor graph in Fig. 4.3, and under the assumption of zero-mean Gaussian noise, the estimation can be solved by minimising geometric distances $d_{\mathfrak{a}}$ associated

with LiDAR features, inertial residuals $\mathbf{r}_I^m$, accelerometer biases residuals $\mathbf{r}_f^m$, gyroscope biases residuals $\mathbf{r}_\omega^m$, and time-shift residuals $r_{\delta_t}^m$. That is,

$$C(\mathcal{S}) = \sum_{\mathfrak{a}\in\mathcal{A}} \|d_\mathfrak{a}\|^2_{\Sigma_{d_\mathfrak{a}}} + \sum_{m=0}^{M-1} \|r_{\delta_t}^m\|^2_{\Sigma_{r_{\delta_t}^m}} + \sum_{m=1}^{M-1} \Big(\|\mathbf{r}_f^m\|^2_{\Sigma_{\mathbf{r}_f^m}} + \|\mathbf{r}_\omega^m\|^2_{\Sigma_{\mathbf{r}_\omega^m}} + \|\mathbf{r}_I^m\|^2_{\Sigma_{\mathbf{r}_I^m}} \Big). \tag{4.4}$$

The different components of $C(\mathcal{S})$ are detailed in Section 4.3. Note that $\Sigma_\bullet$ is the covariance matrix of the variable $\bullet$.

4.3 Back-end

The first two parts of this section present the IMU and bias factors. These are not specific to IN2LAAMA as they only relates to the inertial data. Consequently, the two following subsections are also relevant to IDOL presented in Chapter 5. The Jacobians associated to the different residuals of this section are present in Appendix C.

4.3.1 IMU factors

The IMU factors constitute direct constraints on the IMU poses and velocities. They leverage the GPMs detailed in Chapter 3. The associated residuals $\mathbf{r}_I^m = [\mathbf{r}_{I_r}^m; \mathbf{r}_{I_v}^m; \mathbf{r}_{I_p}^m]$ are obtained directly by manipulating (3.19), (3.20), and (3.21),

$$\begin{aligned}
\mathbf{r}_{I_p}^m =& {\mathbf{R}_W^{\tau_{m\text{-}1}}}^\top \big(\mathbf{p}_W^{\tau_m} - \mathbf{p}_W^{\tau_{m\text{-}1}} - \Delta\tau_m \mathbf{v}_W^{\tau_{m\text{-}1}} - \frac{{\Delta\tau_m}^2}{2}\mathbf{g} \big) - \Delta\mathbf{p}_{\tau_{m\text{-}1}}^{\tau_m} \\
\mathbf{r}_{I_v}^m =& {\mathbf{R}_W^{\tau_{m\text{-}1}}}^\top \big(\mathbf{v}_W^{\tau_m} - \mathbf{v}_W^{\tau_{m\text{-}1}} - \Delta\tau_m \mathbf{g} \big) - \Delta\mathbf{v}_{\tau_{m\text{-}1}}^{\tau_m} \\
\mathbf{r}_{I_r}^m =& \mathrm{Log}\Big({\Delta\mathbf{R}_{\tau_{m\text{-}1}}^{\tau_m}}^\top {\mathbf{R}_W^{\tau_{m\text{-}1}}}^\top \mathbf{R}_W^{\tau_m} \Big),
\end{aligned} \tag{4.5}$$

with $\Delta\tau_m = \tau_m - \tau_{m\text{-}1}$. In (4.4), these residuals are weighted by the information matrix of the GPMs ($\Delta\mathbf{R}_{\tau_{m\text{-}1}}^{\tau_m}$, $\Delta\mathbf{v}_{\tau_{m\text{-}1}}^{\tau_m}$, and $\Delta\mathbf{p}_{\tau_{m\text{-}1}}^{\tau_m}$) that is the inverse of $\Sigma_{\mathbf{r}_I^m}$ with

$$\Sigma_{\mathbf{r}_I^m} = \begin{bmatrix} \Sigma_{\Delta\mathbf{R}_{\tau_{m\text{-}1}}^{\tau_m}} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \Sigma_{\Delta\mathbf{v}_{\tau_{m\text{-}1}}^{\tau_m}} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \Sigma_{\Delta\mathbf{p}_{\tau_{m\text{-}1}}^{\tau_m}} \end{bmatrix}. \tag{4.6}$$

4.3.2 IMU biases and inter-sensor time-shift

As detailed in Chapter 3, the GPMs computation is a function of the accelerometer biases $\mathbf{b}_f$, gyroscope biases $\mathbf{b}_\omega$, and inter-sensor time-shift δ_t. However, these values are not perfectly known at the time of preintegration. In this framework, we model the IMU biases as a Brownian motion as in [93] and the inter-sensor time-shift as a simple Gaussian. We consider the biases and time-shift locally constant during LiDAR frames, and adopt the first-order expansions (3.54), (3.55), and (3.56).

The residuals

$$\mathbf{r}_f^m = \bar{\mathbf{b}}_f^m + \hat{\mathbf{b}}_f^m - \bar{\mathbf{b}}_f^{m\text{-}1} - \hat{\mathbf{b}}_f^{m\text{-}1} \tag{4.7}$$

$$\mathbf{r}_\omega^m = \bar{\mathbf{b}}_\omega^m + \hat{\mathbf{b}}_\omega^m - \bar{\mathbf{b}}_\omega^{m\text{-}1} - \hat{\mathbf{b}}_\omega^{m\text{-}1} \tag{4.8}$$

are used in the biases factors to impose the Brownian motion constraint. The covariance matrices $\Sigma_{\mathbf{r}_f^m}$ and $\Sigma_{\mathbf{r}_\omega^m}$ from (4.4) are defined according to the timestamp difference $\tau_m - \tau_{m\text{-}1}$, and the random-walk characteristics present in the IMU manufacturer specifications.

The time-shift factor residual is simply $r_{\delta_t}^m = \hat{\delta}_{t_m}$ as per the Gaussian noise model. The weighting (inverse of $\Sigma r_{\delta_t}^m$) is arbitrarily set as per the prior knowledge of the inter-sensor time-shift (typically in the order of milliseconds).

4.3.3 Lidar factors

Lidar factors correspond to distance residuals computed between LiDAR feature-points and their corresponding feature-points from other LiDAR frames. As we will explain in the front-end section, the set of feature associations $\mathcal{A}$ contains tuples of 3 (point-to-edge constraints) or 4 feature-points (point-to-plane constraints).

For the LiDAR factors, point-to-line or point-to-plane distances are used. The matched points found in $\mathcal{A}$ are projected in the world frame $\mathfrak{F}_W$ using the calibration parameters, GPMs for each of the points and the current estimates of the IMU poses and velocities (Fig. 4.4). Therefore, a point $\mathbf{x}_L^i \in \mathcal{X}^m$ is projected in $\mathfrak{F}_W$ using (4.2), and the GPMs

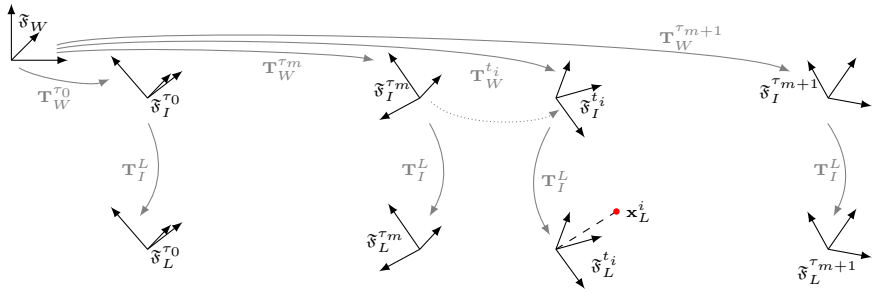


FIGURE 4.4: Frames and frame transformations during a sequence of measurements. $\mathfrak{F}_I^{\tau_m}$ and $\mathfrak{F}_L^{\tau_m}$ respectively represent the IMU and LiDAR frames at time τ_m. The grey continuous line arrows represent the transformations between the different frames. $\mathfrak{F}_W$ is the world fixed frame. The dotted line shows the use of upsampled preintegrated measurements to reproject the point $\mathbf{x}_L^i$.

(3.19) and (3.21):

$$\begin{bmatrix} \mathbf{x}_W^i \\ 1 \end{bmatrix} = \mathbf{T}_W^{t_i} \mathbf{T}_I^L \begin{bmatrix} \mathbf{x}_L^i \\ 1 \end{bmatrix}. \tag{4.9}$$

Let us denote an edge association $\mathfrak{a}_3 \in \mathcal{A}$. $\mathfrak{a}_3 = \{\mathbf{x}_L^i, \mathbf{x}_L^j, \mathbf{x}_L^k\}$ with $\mathbf{x}_L^i \in \mathcal{F}^m$, $\mathbf{x}_L^j \in \mathcal{F}^n$, $\mathbf{x}_L^k \in \mathcal{F}^o$ and $n, o \neq m$. These points are projected in $\mathfrak{F}_W$ via (4.9) to get $\mathbf{x}_W^i$, $\mathbf{x}_W^j$ and $\mathbf{x}_W^k$. The point-to-line distance

$$d_{\mathfrak{a}_3} = \frac{\|(\mathbf{x}_W^i - \mathbf{x}_W^j) \times (\mathbf{x}_W^i - \mathbf{x}_W^k)\|_2}{\|(\mathbf{x}_W^j - \mathbf{x}_W^k)\|_2} \tag{4.10}$$

is used as an edge feature residual.

Let us denote a plane association $\mathfrak{a}_4 \in \mathcal{A}$. $\mathfrak{a}_4 = \{\mathbf{x}_L^i, \mathbf{x}_L^j, \mathbf{x}_L^k, \mathbf{x}_L^l\}$ with $\mathbf{x}_L^i \in \mathcal{F}^m$, $\mathbf{x}_L^j \in \mathcal{F}^n$, $\mathbf{x}_L^k \in \mathcal{F}^o$, $\mathbf{x}_L^l \in \mathcal{F}^p$ and $n, o, p \neq m$. These points are projected in $\mathfrak{F}_W$ via (4.9) to get $\mathbf{x}_W^i$, $\mathbf{x}_W^j$, $\mathbf{x}_W^k$ and $\mathbf{x}_W^l$. The point-to-plane distance

$$d_{\mathfrak{a}_4} = \frac{(\mathbf{x}_W^i - \mathbf{x}_W^j)^\top \Big((\mathbf{x}_W^j - \mathbf{x}_W^k) \times (\mathbf{x}_W^j - \mathbf{x}_W^l)\Big)}{\|(\mathbf{x}_W^j - \mathbf{x}_W^k) \times (\mathbf{x}_W^j - \mathbf{x}_W^l)\|_2} \tag{4.11}$$

is used as a plane feature residual. The variance of the LiDAR residuals requires the knowledge of the state. Therefore, the noise propagation $\mathbf{\Sigma}_{d_\mathfrak{a}} = \mathbf{J}_{d_\mathfrak{a}}(\mathcal{S}) \mathbf{\Sigma}_z (\mathbf{J}_{d_\mathfrak{a}}(\mathcal{S}))^\top$, with $\mathbf{\Sigma}_z$ the covariance of the corresponding LiDAR and GPMs measurements, and $\mathbf{J}_{d_\mathfrak{a}}(\mathcal{S})$ the Jacobian of $d_\mathfrak{a}$ with respect to the sensors measurements evaluated at the current best estimate of the state $\mathcal{S}$ needs to be executed regularly during the optimisation.

4.4 Front-end

The front-end of the proposed method aims at populating the set $\mathcal{A}$ of LiDAR feature-point associations to allow frame-to-frame and loop closure matching.

4.4.1 Feature extraction

The vertical resolution of most of today's LiDARs has driven the design of our feature extraction algorithm toward a channel-by-channel method in a similar way to the one in [10]. The authors of [10] introduced a computationally efficient smoothness score for feature extraction/classification. While robust in weakly structured environments and allowing real-time operations, this score computation is not fully consistent. For example, points belonging to the same planar surface will have different smoothness scores despite the same underlying structure. We propose a feature extraction technique based on linear regression to describe the surface observed by the LiDAR consistently.

Given an N-channel LiDAR, each LiDAR scan $\mathcal{X}^m$ is split into N "lines", $\mathcal{N}_l^m$ ($l = 1, \cdots, N$), according to the elevation of the 3D-points collected. All the points are given a curvature score. The curvature computation aims at fitting lines to two subsets of points adjacent to the point under examination $\mathbf{x}_L^i \in \mathcal{N}_l^m$, and then to retrieve the cosine of the angle between these two lines. The subsets, $\mathcal{L}_i$ and $\mathcal{R}_i$ contain the D previous and following measurements (with respect to $\mathbf{x}_L^i$) in $\mathcal{N}_l^m$.

First, the points need to be reprojected into the LiDAR frame at τ_m ($\mathfrak{F}_L^{\tau_m}$) to remove motion distortion according to the best current estimate of the state $\mathcal{S}$. These reprojected points $\mathbf{x}_{L_m}^i$ are computed as follows:

$$\begin{bmatrix} \mathbf{x}_{L_m}^i \\ 1 \end{bmatrix} = (\mathbf{T}_I^L)^{-1}(\mathbf{T}_W^{\tau_m})^{-1}\mathbf{T}_W^{t_i}\mathbf{T}_I^L \begin{bmatrix} \mathbf{x}_L^i \\ 1 \end{bmatrix}. \tag{4.12}$$

The curvature scores are computed under the approximation that around a certain azimuth the consecutively measured 3D-points belong to the same plane. As shown in Fig. 4.5, and given α^i the new azimuth of $\mathbf{x}_{L_m}^i$, the points in $\mathcal{L}_i$ and $\mathcal{R}_i$ are projected on a plane

FIGURE 4.5: Geometric feature extraction based on linear regression. The points around a given azimuth are assumed to belong to a local plane. On that local plane, linear regressions are performed considering points in $\mathcal{N}_l^m$ on both sides of the i^{th} point $\mathbf{x}_L^i \in \mathcal{N}_l^m$ independently. The curvature score is equal to $\cos(\beta)$ with β the angle between the two fitted lines.

space around α^i

$$x_{P_i}^k = |\mathbf{x}_{L_m}^{i+k}| \sin(\alpha^{i+k} - \alpha^i), \tag{4.13}$$

$$y_{P_i}^k = |\mathbf{x}_{L_m}^{i+k}| \cos(\alpha^{i+k} - \alpha^i), \tag{4.14}$$

with $k = -D, \cdots, D$ ($D = 5$ in our implementation).

$$\mathbf{X}_{\mathcal{L}_i} = \begin{bmatrix} 1 & x_{P_i}^{-D} \\ \vdots & \vdots \\ 1 & x_{P_i}^{0} \end{bmatrix}, \mathbf{X}_{\mathcal{R}_i} = \begin{bmatrix} 1 & x_{P_i}^{0} \\ \vdots & \vdots \\ 1 & x_{P_i}^{D} \end{bmatrix}, \tag{4.15}$$

$$\mathbf{Y}_{\mathcal{L}_i} = \begin{bmatrix} y_{P_i}^{-D} & \cdots & y_{P_i}^{0} \end{bmatrix}^\top \text{ and } \mathbf{Y}_{\mathcal{R}_i} = \begin{bmatrix} y_{P_i}^{0} & \cdots & y_{P_i}^{D} \end{bmatrix}^\top$$

group the projected points coordinates according to the two adjacent subsets $\mathcal{L}_i$ and $\mathcal{R}_i$. In the rest of this section, $\bullet$ represents either $\mathcal{L}_i$ or $\mathcal{R}_i$. A line of slope $s_\bullet$ and y-intercept $q_\bullet$ can be fitted to the subset $\bullet$ with

$$\begin{bmatrix} q_\bullet & s_\bullet \end{bmatrix}^\top = \left(\mathbf{X}_\bullet^\top \mathbf{X}_\bullet\right)^{-1} \mathbf{X}_\bullet^\top \mathbf{Y}_\bullet, \tag{4.16}$$

and an associated unit direction vector can be obtained as

$$\mathbf{v}_\bullet = \begin{bmatrix} \frac{1}{\sqrt{1+s_\bullet^2}} & \frac{s_\bullet}{\sqrt{1+s_\bullet^2}} \end{bmatrix}^\top. \tag{4.17}$$

(a) Outward edge (b) Inward edge

FIGURE 4.6: Edge classification for data association robustness.

The average and maximum regression error values, respectively

$$\bar{e}^i_\bullet = \frac{1}{|\bullet|} \sum_{k|\mathbf{x}^k_L \in \bullet} \left| y^k_{P_i} - q_\bullet - s_\bullet x^k_{P_i} \right| \text{ and} \tag{4.18}$$

$$e^i_\bullet = \max_{k|\mathbf{x}^k_L \in \bullet} \left(\left| y^k_{P_i} - q_\bullet - s_\bullet x^k_{P_i} \right| \right), \tag{4.19}$$

are used to reject points or to detect border of occlusions as per Algorithm 1 The score $c_i = \mathbf{v}^\top_{\mathcal{L}_i} \mathbf{v}_{\mathcal{R}_i}$ represents the cosine of the angle between the two fitted lines. As a consequence, c_i is close to 1 when the underlying surface is planar and decreases with the sharpness of edges.

As in [10], surfaces close to being parallel to the laser beams are rejected as features. We also use a system of bins and a maximum number of features per bin on each laser line to ensure the features are spread over the whole scan. The points with the highest scores in each of the bins of $\mathcal{N}^m_l$ are classified as planar points and the lowest scores as edges according to arbitrarily chosen maximum numbers of features per bin and thresholds on scores. The edge orientation, classified as inward (pointing toward the LiDAR as shown in Fig. 4.6(b)) or outward (pointing away from the LiDAR as shown in Fig. 4.6(a)), can be defined by looking at the values of the regressed lines' parameters. This edge classification brings additional robustness to the later feature association as inward and outward edges cannot be matched together. All the planar features in $\mathcal{N}^m_l$ with $l = 1, \cdots, N$, are grouped into a set $\mathcal{P}^m$, the inward edges in $\mathcal{E}^m_I$ and outward edges in $\mathcal{E}^m_O$. The reader should note that the feature set (from the back-end section of this chapter) $\mathcal{F}^m = \mathcal{P}^m \cup \mathcal{E}^m_I \cup \mathcal{E}^m_O$.

4.4.2 Feature recomputation

The aforementioned process of feature extraction is computationally costly and depends on the last estimate of the state $\mathcal{S}$. IN2LAAMA integrates a way to check the validity of

Algorithm 1 Algorithm used to reject lidar points as potential features according to the linear regression errors.

Input:
$\bar{e}_{th}, e_{th}$: Thresholds on mean and max regression error
$\bar{e}^i_{\mathcal{L}}, \bar{e}^i_{\mathcal{R}}, e^i_{\mathcal{L}}, e^i_{\mathcal{R}}$: Regression errors in $\mathcal{L}_i$ and $\mathcal{R}_i$
Output:
Boolean flag: *Accept_point* or *Reject_point*

function REGRESSIONOK ($\bullet = \mathcal{L}_i$ or $\mathcal{R}_i$)
return $(\bar{e}^i_\bullet < \bar{e}_{th})$ & $(e^i_\bullet < e_{th})$
end function

function OCCLUSION ($\bullet = \mathcal{L}_i$ or $\mathcal{R}_i$)
return $(s_\bullet x^{-1}_{P_i} + q_\bullet < |\mathbf{x}^i_{L_m}|)$
end function

if REGRESSIONOK($\mathcal{L}_i$) & REGRESSIONOK($\mathcal{R}_i$) **then**
  return *Accept_point*
else if !(REGRESSIONOK($\mathcal{L}_i$)) & !(REGRESSIONOK($\mathcal{R}_i$))
  then return *Reject_point*
else
  if REGRESSIONOK($\mathcal{L}_i$) **then**
    Remove $\mathbf{x}^i_L$ from $\mathcal{R}_i$ and recompute regression
    if !(REGRESSIONOK($\mathcal{R}_i$)) ‖ OCCLUSION($\mathcal{R}_i$) **then**
      return *Reject_point*
    else
      $s_{\mathcal{R}_i} \leftarrow (y^1_{P_i} - y^0_{P_i})/(x^1_{P_i} - x^0_{P_i})$, recompute $\mathbf{v}_{\mathcal{R}_i}$
      return *Accept_point*
    end if
  else if REGRESSIONOK($\mathcal{R}_i$) **then**
    Remove $\mathbf{x}^i_L$ from $\mathcal{L}_i$ and recompute regression
    if !(REGRESSIONOK($\mathcal{L}_i$)) ‖ OCCLUSION($\mathcal{L}_i$) **then**
      return *Reject_point*
    else
      $s_{\mathcal{L}_i} \leftarrow (y^0_{P_i} - y^{-1}_{P_i})/(x^0_{P_i} - x^{-1}_{P_i})$, recompute $\mathbf{v}_{\mathcal{L}_i}$
      return *Accept_point*
    end if
  end if
end if

features without the need to recompute all the linear regressions.

For the moment, let us consider planar features only and define N_f as the maximum number of planar features selected per bin during the feature extraction performed on the m^{th} LiDAR frame. The set of planar features in the k^{th} bin of the m^{th} LiDAR frame is denoted $\mathcal{B}^j_{m,k}$ with $j > 0$ corresponding to the j^{th} time the features of frame m have been computed. Considering the case $j = 1$, the scores c_i are computed for all points in $\mathcal{X}^m$.

The points are then sorted according to their score in a decreasing order. Starting from the highest score, points are added to $\mathcal{B}^j_{m,k}$ if their score is above a threshold and as long as $|\mathcal{B}^j_{m,k}| < N_f$. The algorithm also stores the N_f next candidates (even if they do not match the threshold) in the set $\mathcal{C}^j_{m,k}$. Note that the set union of the planar feature bins $\mathcal{B}^j_{m,k}$ is $\mathcal{P}^m$.

In the case of $j > 1$, typically after an optimisation iteration of the factor graph, the state $\mathcal{S}$ changes. The features potentially need to be recomputed. The score of the points in $\mathcal{B}^{j-1}_{m,k}$ and $\mathcal{C}^{j-1}_{m,k}$ are recomputed and sorted by decreasing order. The point selection for the bins is done as if $j = 1$, but using point scores from $\mathcal{B}^{j-1}_{m,k} \cup \mathcal{C}^{j-1}_{m,k}$ and not from $\mathcal{X}^m$. An overlap ratio of number of features is computed as

$$\Theta_{j,m} = \frac{\left|\left(\bigcup_k \mathcal{B}^j_{m,k}\right) \cap \left(\bigcup_k \mathcal{B}^{j-1}_{m,k}\right)\right|}{\left|\left(\bigcup_k \mathcal{B}^{j-1}_{m,k}\right)\right|}. \tag{4.20}$$

If $\Theta_{j,m}$ is close to one, then $\mathcal{B}^j_{m,k} \leftarrow \mathcal{B}^{j-1}_{m,k}$ and $\mathcal{C}^j_{m,k} \leftarrow \mathcal{C}^{j-1}_{m,k}$. Otherwise, $\mathcal{B}^j_{m,k}$ and $\mathcal{C}^j_{m,k}$ are recomputed from $\mathcal{X}^m$ as per the case $j = 1$. Similar process is used for edge features.

4.4.3 Data association

The proposed scan registration method requires matching features from frame-to-frame. Feature matching is usually prone to outliers. Thus, a robust process for data association is needed. This section describes the different processes used in IN2LAAMA for matching and outlier rejection.

4.4.3.1 Feature matching

Given a pair of LiDAR frames i and m reprojected into $\mathfrak{F}_W$, the method looks for the 3 nearest neighbours of each point from $\mathcal{P}^i$ in $\mathcal{P}^m$. For points in $\mathcal{E}^i_I$ and $\mathcal{E}^i_O$, only the 2 nearest neighbours are searched in $\mathcal{E}^m_I$ and $\mathcal{E}^m_O$ respectively. In both cases, to limit the impact of the measurements' noise on the point-to-line and point-to-plane distances used as LiDAR residuals, the $n = \{2, 3\}$ closest points need to be spatially spread over some minimum

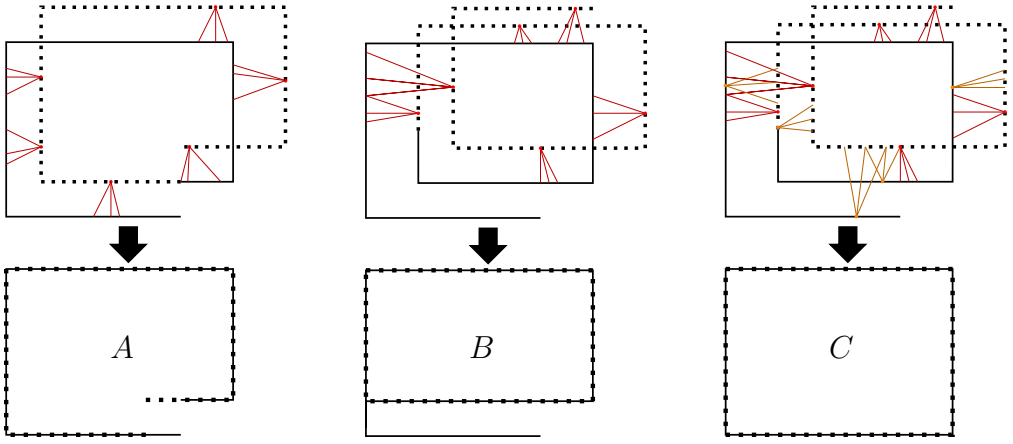


FIGURE 4.7: Different data association strategies between a scan $\mathcal{X}^m$ (dashed line) and its previous scan $\mathcal{X}^{m\text{-}1}$ (plain line). The top row represents the data association. The bottom row shows the potential results after minimising point-to-plane distances. A uses 360° scans with back-association. B uses scans greater than 360°, with back-association. C extends B with back-and-forth-association. C ensure consistency of the LiDAR scans, whereas A and B do not.

distances. The n closest points cannot belong to a single LiDAR channel. If the n closest points do not satisfy these conditions, the subsequent closest points are considered. For planar feature associations, the collinearity of the 3 points from $\mathcal{P}^m$ is checked. Kd-trees [112] are used for efficient nearest neighbour searches. The data associations are included in $\mathcal{A}$ as tuples of 3 or 4 as per the type of feature.

To enforce LiDAR scans' consistency without additional constraints, the proposed method considers scans greater than 360° (520° in our implementation) and does the data association both from i to m, and from m to i. The idea behind these choices is illustrated in Fig. 4.7 through a 2D example, in which the system moves in a rectangular room detecting only planar features and leveraging only LiDAR factors. In scenario A, the data association between scans of 360° (or less) does not necessarily allow for the correction of the motion distortion in the scans. Individually the LiDAR residuals do not robustly constrain the motion distortion present in each frame, and the sensor noise can worsen this situation even more. Without constraints on pose continuity between the last point of $\mathcal{X}^{m\text{-}1}$ and the first one of $\mathcal{X}^m$, the registration is unlikely to correct the distorted scans properly. In scenario B, the scans are greater than 360°, but the lower wall of the first scan does not appear in any data association. Consequently, under some particular circumstances, the

registration can still contain some motion distortion in $\mathcal{X}^{m\text{-}1}$. The scenario C fully constrains the scan consistency correcting the motion distortion by having both scans greater than 360° combined with back-and-forth data association.

Intuitively, the greater the angle swept by a scan, the better the scan consistency, but there is a trade-off with the execution time. The GPMs' GP regressions are computed from the IMU readings that are collected during a LiDAR scan. Therefore, larger LiDAR scans imply cubically longer inference time as per the $\mathcal{O}(n^3)$ complexity of GP interpolation.

4.4.3.2 Outliers rejection

To remove outliers from $\mathcal{A}$, matching points spread over too large areas are disregarded. For planar features, the outlier detection additionally analyses the patch around the matched feature-points. Considering a planar point $\mathbf{x}_W^i$ associated with $\mathbf{x}_W^j$, $\mathbf{x}_W^k$, and $\mathbf{x}_W^l$, the line-neighbours of each of the 3 matched points from $\mathcal{P}^m$ ($\mathcal{L}_j$, $\mathcal{R}_j$, $\mathcal{L}_k$, $\mathcal{R}_k$, $\mathcal{L}_l$, and $\mathcal{R}_l$) are placed in a set $\mathcal{U}$. If the points in $\mathcal{U}$ do not belong to the plane described by $\mathbf{x}_W^j$, $\mathbf{x}_W^k$, and $\mathbf{x}_W^l$, the association is rejected. Formally, an association is valid if

$$\max_{\mathbf{x}_W^u \in \mathcal{U}} \left(\frac{\left(\mathbf{x}_W^u - \mathbf{x}_W^j\right)^\top \left(\left(\mathbf{x}_W^j - \mathbf{x}_W^k\right) \times \left(\mathbf{x}_W^j - \mathbf{x}_W^l\right) \right)}{\| \left(\mathbf{x}_W^j - \mathbf{x}_W^k\right) \times \left(\mathbf{x}_W^j - \mathbf{x}_W^l\right) \|_2} \right) \tag{4.21}$$

complies with the LiDAR range noise.

For edge features, the following two nearest neighbours in $\mathcal{E}_I^m$ or $\mathcal{E}_O^m$ (depending on the edge orientation) are queried. If not all the four neighbours lie on the same line (with compliance to the LiDAR range noise), the feature association is rejected.

4.4.4 Loop-closure detection

Loop-closures allow localisation and mapping algorithms to correct the accumulated drift inherent to frame-to-frame trajectory estimation. The proposed method does not address large drift scenarios (such as the kidnapped robot scenario). It is out of the scope of this chapter and part of the future work. Here, a simple geometric loop-closure detection based on estimated poses proximity has been implemented. In other words, if two poses

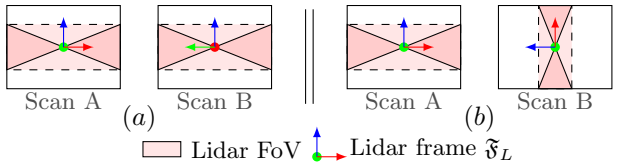


FIGURE 4.8: Illustration of two loop-closure scenarios with different orientation gaps between the LiDAR scans. The sensing system is moved inside a room represented by the outer rectangle. In (a), between scans A and B, the LiDAR is rotated by 90° around its spinning axis. In (b), it is also rotated by 90° but around another axis. In case (a) there is a lot of geometric overlap to register the two frames, whereas in (b) the registration is challenging because of the risk of poor data association due the small overlap.

are spatially close enough given the estimated state, a LiDAR factor is built between these two poses (performing feature matching and adding a set of residuals to the cost functions). As an intuition, for indoor scenarios, it aims at detecting loop closures when the drift is smaller than the dimensions of the rooms. An optional ICP test is conducted to validate or reject a loop closure candidate. In the proposed method, loop-closures are modelled with additional LiDAR factors, as shown in Fig. 4.3.

Commonly used LiDARs have a 360° FoV around the spinning axis (azimuth) but have a narrower angular range on the other axis (elevation). The nature of that setup results in big overlaps between scans that have been collected while the LiDAR rotates around its spinning axis. On the other hand, if the LiDAR rotates around other axes, the geometric data overlap decreases, making the registration between two scans more challenging. This is illustrated in Fig. 4.8. Therefore, the direct angle between two orientations cannot be used as part of the proximity metric. The 360° "horizontal" field-of-view of the LiDAR must be taken into account.

Let us consider a spinning LiDAR that sweeps the environment around the z-axis of its reference frame. The origin of the frame coincides with the LiDAR optical centre. The different metrics used to define the closeness between two LiDAR frames $\mathfrak{F}_L^{\tau_m}$ and $\mathfrak{F}_L^{\tau_i}$ are as follows:

- d_r is the radial distance of the origin of $\mathfrak{F}_L^{\tau_i}$ regarding the z-axis of $\mathfrak{F}_L^{\tau_m}$.
- d_h is the point-to-plane distance between the origin of $\mathfrak{F}_L^{\tau_i}$ and the plane formed by the x and y axes of $\mathfrak{F}_L^{\tau_m}$.

- d_α is the angle between the z-axes $\mathfrak{F}_L^{\tau_i}$ and $\mathfrak{F}_L^{\tau_m}$ when their origins coincide.

More formally,

$$d_r = \sqrt{{x_m^i}^2 + {y_m^i}^2} \tag{4.22}$$

$$d_h = |z_m^i| \tag{4.23}$$

$$\cos(d_\alpha) = {\mathbf{u}_z}^\top {\mathbf{R}_I^L}^\top {\mathbf{R}_W^{\tau_m}}^\top \mathbf{R}_W^{\tau_i} \mathbf{R}_I^L \mathbf{u}_z, \tag{4.24}$$

with $\begin{bmatrix} x_m^i & y_m^i & z_m^i & 1 \end{bmatrix}^\top = (\mathbf{T}_I^L)^{-1} (\mathbf{T}_W^{\tau_m})^{-1} \mathbf{T}_W^{\tau_i} \begin{bmatrix} \mathbf{p}_I^L \\ 1 \end{bmatrix}$

and $\mathbf{u}_z = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^\top$.

To limit the number of redundant loop-closures, IN2LAAMA sets a minimum time between two consecutive loop-closures, as well as a minimum gap time between the two frames used for closures. The algorithm looks for loop-closures every time a new frame is added to the factor graph. The metrics d_r, d_h, and $|\cos(d_\alpha)|$ are computed between this new frame and the previous frames that satisfy the aforementioned time conditions, by order of increasing timestamp. The first frame that complies with thresholds on the above metrics is considered as a valid loop-closure candidate. The threshold on d_r is dynamically computed for each frame and is equal to the upper 1-sigma bound (mean plus standard deviation) of the range measurements of that frame. The threshold on d_h is user-defined according to the type of trajectory and environment, and the one on $|\cos(d_\alpha)|$ is set according to the LiDAR vertical FoV (2/3 of the vertical FoV in our implementation). Optionally, in exchange for an additional computational cost, a standard ICP [34] is conducted between the new frame and frames contained in a time window around the loop-closure candidate. In this case, the loop-closure is validated only if the ICP fitness score is below a given value. The validation of a loop-closure leads to the addition of a new LiDAR factor in the factor graph as it can be seen in Fig. 4.3.

4.5 On the factor graph and implementation

4.5.1 Factor graph for localisation and mapping

In the absence of trajectory and velocity priors (*no* Global Position System (GPS), *no* odometry, etc.), the factor graph used for state estimation is built iteratively and optimised as new factors are added to the cost function. Algorithm 2 shows the proposed strategy. Intuitively, the first frames need particular attention because the initial state is completely unknown when the system is switched on. Therefore, during the initialisation step, the integration of any single new frame triggers both the optimisation of the state $\mathcal{S}$, and the feature recomputation and data association for *every* frame already in the graph. Once this initialisation step is finished, motion-distortion is removed from the corresponding LiDAR scans. Feature recomputation in these frames is not needed later in the process (the features are reliable as computed on distortion-free LiDAR scans). Only the latest frames have their features and data association recomputed as the state $\mathcal{S}$ changes. Note that the method still considers motion distortion in *all* the frames at all times because GPMs are used in the LiDAR residuals, and the full trajectory is part of the state $\mathcal{S}$. Integrating IN2LAAMA into a more complex system that provides reliable prior information (e.g., a robotic platform with odometry, GPS in outdoor scenarios, etc.), could significantly reduce the number of iterations needed to build the factor graph (potentially in one go). The execution time would be greatly reduced.

4.5.2 Factor graph for autocalibration, localisation, and mapping

The localisation and mapping procedure relies on a good knowledge of $\mathbf{T}_I^L$. Using inaccurate calibration parameters can lead to contradicting information in the factor graph. As a consequence, using the integration of inertial data from the last estimate of the state provides a prior that drift rapidly. The proposed method allows for the extrinsic calibration of the LiDAR with respect to the LiDAR. This feature is denoted *autocalibration* as no additional factor nor calibration target is required.

Algorithm 2 Algorithm describing the factor graph construction and optimisation procedure.

Input:
M: Number of frames in the dataset
N_g: Number of frames to initialise initial conditions
N_e: Number of frames added between each optimisation
Calib: Activate calibration parameters estimation
Output:
$\mathcal{S}$: State estimate

// Initialisation
$F \leftarrow$ Create empty factor graph
for $n = 0 : N_g - 1$ **do**
    Add frame n and associated factors to F
    repeat
        $\mathcal{S} \leftarrow Optimise(F)$
        Check/recompute features in frames 0 to n
    until Reach nb. iterations ‖ State converges
end for
if *Calib* **then**
    $N \leftarrow 1$
else
    $N \leftarrow N_e$
end if
// End initialisation

for $n = N_g : M - 1$ **do**
    Add frame n and associated factors to F
    if $n \mod N = 0 \parallel n = M - 1$ **then**
        repeat
            $\mathcal{S} \leftarrow Optimise(F)$
            Check/recompute features in frames $n - N_e$ to n
        until Reach nb. iterations ‖ State converges
    end if
    Check loop-closure. If loop detected: $\mathcal{S} \leftarrow Optimise(F)$
end for
if *Calib* **then**
    Add $\mathbf{T}_I^L$ to $\mathcal{S}$
    repeat
        $\mathcal{S} \leftarrow Optimise(F)$
        Check/recompute features in frames 0 to $M - 1$
    until Reach nb. iterations ‖ State converges
end if

For the autocalibration procedure, IN2LAAMA runs the localisation and mapping procedure based on any prior knowledge of $\mathbf{T}_I^L$, while performing the optimisation step every time a new frame is added to the factor graph (cf. $N \leftarrow 1$ in the algorithm of Fig. 2). Once the full trajectory is estimated based on the inaccurate calibration parameters, the calibration parameters $\mathbf{T}_I^L$ are included as part of the state $\mathcal{S}$ to be estimated along with the different poses, velocities, and bias and time-shift corrections already in $\mathcal{S}$. Given this new $\mathcal{S}$, the proposed method iteratively optimises the factor graph and recomputes features until the estimate converges.

4.5.3 Robustness of state estimation

A major challenge for robotics state estimation is the management of outliers. In a localisation and mapping framework, like the one presented here, outliers can originate through different phenomena. One is simply wrong data association. Despite conservative LiDAR feature-association rules, in cluttered environments, it is still likely that outliers pass the rejection tests mentioned in Subsection 4.4.3.2. To address this issue, bisquare weights [113] are applied to each individual LiDAR residual.

Another source of outliers is the "quality" of the sensor models used (one can interpret it the other way around as "the quality of the sensor data"). By definition, a mathematical model is an approximation (more or less accurate) used to describe a real-world system. IN2LAAMA uses common models for the sensor readings (additive zero-mean Gaussian noise) and the IMU biases (Brownian motion). These considerations might not capture reality accurately. In a multi-sensor estimation framework, sensor data that do not correspond exactly to the models employed create contradicting information in the estimation process. We propose to overcome this issue by applying Cauchy loss functions on each of the LiDAR and IMU factors to attenuate these outliers. The experiment in Subsection 4.6.1.3 shows the robustness gain in the presence of erroneously modelled IMU measurements.

4.5.4 Bias observability

The inertial navigation literature have previously studied the observability of IMU biases in different estimation frameworks [114–116]. It has been proven that in the presence of inertial data, the biases of the accelerometer are observable only if the attitude of the system is perfectly known or if the trajectory contains rotations [110]. If none of these conditions is satisfied, the estimated state $\mathcal{S}$ is not unique. In other words, in the presence of translation-only motions, there is an infinity of orientation-biases combinations that correspond to a given trajectory.

In the case of LiDAR-inertial fusion, a LiDAR alone cannot provide an accurate global attitude without relying on strong heuristics that are not desired in generic localisation and mapping frameworks (e.g. Manhattan world with walls aligned with gravity vector). Therefore, an extra constraint is needed to tackle the problem of translation-only trajectories. The proposed method overcomes this problem by integrating a simple factor that penalises the distance between the estimated accelerometer biases and the null vector.

This additional factor on $\hat{\mathbf{b}}_f^0$ is added to the factor graph upon creation. Once the final frame's factors are added to the factor graph, the observability constraint is released (weight null), and the cost function is minimised. If the magnitude of the accelerometer biases is far from zero, the constraint is re-established and the optimisation run again. This strategy covers the scenarios where the estimation ambiguity is present only at the start of the trajectory. Note that in the non-observable cases (translation-only motion), the estimated biases and global orientation are inaccurate, but the trajectory, therefore the map, is still consistent (but not gravity aligned).

4.5.5 GPMs and memory

The main attribute of the GPMs is to make precise inertial data available for each of the points collected by the LiDAR and therefore allow for the precise motion distortion correction. The drawbacks of these measurements are the computation time (due to the GP interpolations and the numerical integration) and memory usage. Because the GPMs are relatively slow to compute, storing them is essential to limit the global execution

time. On the other hand, GPMs need a significant amount of memory as each GPM is stored on at least 150 floating-point numbers (preintegrated measurements, covariance matrix, Jacobians for bias and time-shift corrections). Based on the Velodyne VLP-16 and double-precision floating-point numbers, it represents a memory consumption of more than 340MB per second of data solely to store the GPMs. To reduce the memory footprint and make IN2LAAMA executable on standard computers, only the GPMs associated to the last N_{GPM} frames are stored. As per the localisation and mapping procedure shown in the algorithm of Fig. 2, choosing N_{GPM} equal to $\max(N_g, N_e)$ does not impact the estimation time. However, this strategy requires the recomputation of all the GPMs to export the dense map when the estimation process terminates.

4.6 Experiments and results

The proposed framework has been evaluated in simulation and on real data from our platform and a public dataset. Our real-world platform is a self-contained LiDAR and IMU sensor suite;

- Velodyne VLP-16, 16-channel ($\pm 15\,^\circ$) LiDAR rotating at 10 Hz with a density of 300k point per second and noise of ± 3 cm.
- Xsens MTi-3, 3-axis accelerometer and 3-axis gyroscope sampling at 100Hz with noise of $0.02\,\mathrm{m/s^2}$ and $0.097\,^\circ/\mathrm{s}$.

The simulated datasets have been generated to match the characteristics of the above-mentioned system moving in a virtual room constituted of 7 planes. Both the back-end and front-end of the proposed method are tested and evaluated in our simulated experiments. In the rest of this section, the A-LOAM implementation[1] of [10] is used to benchmark the proposed method. This last technique has been chosen for its top performance with LiDAR systems in the KITTI odometry benchmark [11]. Our implementation of IN2LAAMA is built upon the non-linear least-square solver Ceres[2] with analytical Jacobians of the cost function.

[1] https://github.com/HKUST-Aerial-Robotics/A-LOAM
[2] http://ceres-solver.org

4.6.1 Simulation - localisation and mapping

This set-up aims at evaluating different configurations of the proposed framework for localisation and mapping. The trajectories of the simulated sensor suite have been generated from sine functions with random frequencies and amplitudes. The extrinsic calibration between sensors is randomly generated for each trajectory. The results are evaluated over 50-run Monte Carlo simulations.

4.6.1.1 Odometry

First, we want to evaluate the advantages of using IMU factors for odometry-like localisation and mapping by comparing the accuracy of IN2LAAMA against our preliminary work [15] (which did not integrate IMU factors, and [10]. We evaluate the localisation accuracy of the three frameworks on three sets of trajectories that have different levels of angular velocities. The loop closure detection is deactivated for these experiments.

Table 4.1 reports the trajectories' parameters as well as the localisation errors against ground truth for [10], [15], and IN2LAAMA. The poor performance of [10] is easily explained by the fact that the motion starts at the very beginning of the simulation. Consequently, even the first LiDAR frame contains motion distortion. This cannot be properly corrected by [10], and leads to unrecoverably large localisation error.

In the case of low angular velocities (row "Slow"), both [15] and IN2LAAMA succeed in estimating the system pose. The integration of IMU factors improves the trajectory estimate slightly. Note that the presence of higher angular velocities leads to smaller overlap between LiDAR scans. Thus, the "Moderate" and "Fast" trajectories contain cases where the overlap between consecutive scans is not enough for the LiDAR factors to fully constrain the frame-to-frame motion. These "degenerated" trajectories trigger estimation failure of [15], but are correctly handled by the integration of frame-to-frame IMU factors. Note that the accuracy metrics of Table 4.1 are computed on successful runs only, therefore advantaging [15] in the "Fast" scenario.

Slow motion (angular velocity in ∘/s: avg 14.7, max 22.1)			
Framework	[10]	[15] (No IMU factors)	IN2LAAMA
Num. fails	**0**	**0**	**0**
Final position error (m)	5.67 ± 2.63	0.11 ± 0.06	**0.06** ± 0.03
Final orientation error (°)	27.9 ± 13.6	0.80 ± 0.42	**0.12** ± 0.07
Relative position error (m)	0.47 ± 0.14	0.01 ± 0.002	**0.003** ± 3e-4
Relative orientation error (°)	1.46 ± 0.51	0.03 ± 0.002	**0.005** ± 1e-4
RMSE position error (m)	5.62 ± 1.72	0.08 ± 0.03	**0.04** ± 0.02
RMSE orientation error (°)	29.2 ± 8.98	0.56 ± 0.24	**0.09** ± 0.04

Moderate motion (angular velocity in ∘/s: avg 49.0, max 78.2)			
Framework	[10]	[15] (No IMU factors)	IN2LAAMA
Num. fails	**0**	1	**0**
Final position error (m)	7.03 ± 3.41	0.34 ± 0.26	**0.30** ± 0.57
Final orientation error (°)	57.2 ± 26.2	2.17 ± 1.51	**1.37** ± 6.31
Relative position error (m)	0.48 ± 0.17	0.02 ± 0.004	**0.004** ± 0.002
Relative orientation error (°)	4.85 ± 1.84	0.06 ± 0.009	**0.009** ± 0.018
RMSE position error (m)	6.29 ± 2.22	0.24 ± 0.12	**0.19** ± 0.38
RMSE orientation error (°)	56.1 ± 17.9	1.70 ± 0.86	**0.81** ± 3.63

Fast motion (angular velocity in ∘/s: avg 125, max 198)			
Framework	[10]	[15] (No IMU factors)	IN2LAAMA
Num. fails	**0**	37	**0**
Final position error (m)	16.2 ± 5.60	**0.39** ± 0.14	0.96 ± 0.61
Final orientation error (°)	119 ± 38.2	3.39 ± 1.65	**0.59** ± 3.00
Relative position error (m)	0.53 ± 0.12	0.04 ± 0.01	**0.007** ± 0.003
Relative orientation error (°)	13.0 ± 3.45	0.10 ± 0.02	**0.007** ± 0.009
RMSE position error (m)	11.1 ± 2.71	**0.28** ± 0.09	0.57 ± 0.36
RMSE orientation error (°)	90.7 ± 20.6	2.38 ± 0.92	**0.36** ± 1.71

TABLE 4.1: Quantitative results of the odometry set-up in simulated environment (50-run Monte Carlo simulation). The trajectories have an average length of 288.7 m, an average velocity of 4.85 m/s, and a maximum velocity of 7.35 m/s. The different tables correspond to different levels of angular velocity during the trajectory. The errors displayed are computed only on the successful runs against the ground truth. The RMSE errors are computed all along each trajectory estimates. The relative errors correspond to frame-to-frame registration errors.

Loop-closure	Final pos. error (m)	Final rot. error (°)	RMSE pos. error (m)	RMSE rot. error (°)
Without	0.110 ± 0.043	0.30 ± 0.19	0.051 ± 0.021	0.17 ± 0.11
With	**0.011** ± 0.011	**0.15** ± 0.12	**0.019** ± 0.007	**0.10** ± 0.07

TABLE 4.2: Quantitative results of trajectory estimation when integrating loop closures constraints in the factor graph. The set of trajectories has the following characteristics: mean distance = 210 m, mean velocity = 3.53 m/s, mean angular velocity = 8.16 °/s. The relative errors correspond to frame-to-frame registration errors.

4.6.1.2 Loop-closure

This set-up aims at demonstrating the ability of the proposed method to perform simultaneous localisation and mapping (SLAM) by integrating loop closures in the batch optimisation. A set of simulated trajectories is generated so that the first and last poses coincide. Table 4.2 shows the localisation results with and without the proposed loop closure detection method. The numbers show that loop closures help to reduce both the final pose error as well as the overall localisation error all along the trajectory.

4.6.1.3 Robustness to inaccurate sensor model

This set-up has been designed to demonstrate the gain of robustness brought by the use of Cauchy loss functions on LiDAR and IMU factors. As mentioned in Section 4.5.3, a sensor model is only an approximation of reality. In real data, the sensors do not always behave as per the manufacturer specifications or noise models. This is the scenario that we are trying to emulate in this set-up.

To simulate a "mismatch" between model and reality, we perturb the IMU sensitivity by simply multiplying the inertial data by a constant. By looking at the localisation accuracy shown in Table 4.3, both in terms of error and number of failure cases, it is clear that the loss functions make IN2LAAMA more robust to discrepancies between the IMU model and the actual readings.

Cauchy loss	IMU data multiplier 1.01	1.03	1.05
Without	0.23 ± 0.09 (0)	0.70 ± 0.27 (12)	1.03 ± 0.31 (34)
With	**0.21** ± 0.14 (0)	**0.51** ± 0.19 (0)	**0.80** ± 0.29 (0)

TABLE 4.3: Analysis of the robustness gained by using Cauchy loss functions on LiDAR and IMU factors. The results represent the RMSE position error, in meters, computed upon a 50-run Monte Carlo simulation. The digits in parenthesis correspond to the number of failure cases. The trajectories are 47.2 m-long on average and have the "Fast" velocity (linear and angular) profile from Table 4.1.

	Num. fails	RMSE pos. (m)	RMSE rot. (°)
Constant vel.	18	2.67 ± 4.84	19.9 ± 24.7
IN2LAAMA (no motion model)	**0**	**0.087** ± 0.041	**0.088** ± 0.267

TABLE 4.4: Comparison of IN2LAAMA with a modified version that assumes constant angular and linear velocities, over a 50-run Monte Carlo simulation. The trajectories are 95.2 m-long on average and have the "Fast" velocity (linear and angular) profile from Table 4.1.

4.6.1.4 No motion model

Using individual preintegrated measurements for each LiDAR point, while leveraging non-parametric interpolation, allows the proposed framework to alleviate the constraints of an explicit motion model. This set-up aims at demonstrating the importance of not imposing a motion model to the state estimation. We compare IN2LAAMA with a modified version of it built on the assumption of constant angular and linear velocities during the LiDAR frames. Table 4.4 displays the pose error of both frameworks over a 50-run simulation. The results clearly demonstrate the accuracy gain of alleviating the use of such a restrictive motion model.

4.6.2 Simulation - front-end

This subsection discusses the proposed feature extraction, and compares it with the front-end of [10]. We show that our method has a consistent behaviour with respect to the observed surface by using a scoring system robust to the lidar viewpoint. In this regard, we have computed the feature score for each of the points of simulated scans. Our method computes a score that represents the cosine of the angle present in a patch. The higher the score (with 1 being the maximum value), the more planar the patch. In [10], the

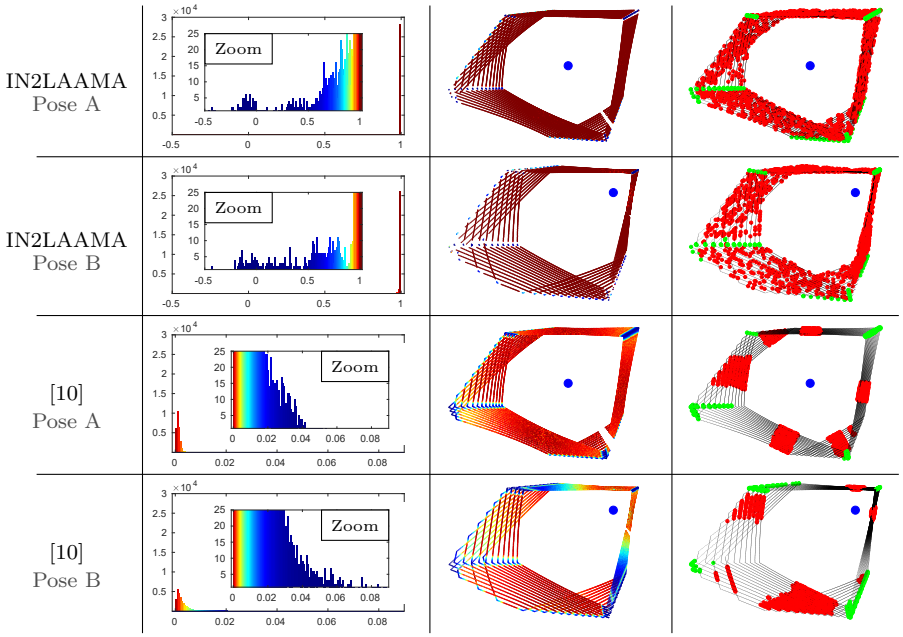


FIGURE 4.9: Feature score comparison between IN2LAAMA and [10]. (a) Histograms of the points' score. (b): Spatial visualisation of the score (same colours as histograms). (c): Features selected with the 100 "most planar" points of each lidar channel in red and the 15 sharpest edge points in green. IN2LAAMA's score exposes a physical value (here, edges have a score $\cos(\beta) < \cos(45°)$, cf. Fig. 4.5). The blue dots are the lidar position.

point score gives an evaluation of smoothness in a patch but does not correspond to any particular physical measurement of the actual geometry. Nonetheless, this score tends to be low in planar patches and high in edge-like patches.

Fig. 4.9 (a) shows the histograms of the scores in simulated environment. The histograms illustrate that the scores computed by IN2LAAMA are consistent across the scan (the bin around 1 dominates largely the histogram). The zoomed-in plots show the modes associated with the different edges of the environment. However, with [10]'s scoring system (inverse to ours), the distinction of the different surface types is ambiguous. Fig. 4.9 (b) spatially represents the points' score with colours, and Fig. 4.9 (c) shows the subsequent feature selection. One can see that IN2LAAMA's score is consistent with the observed surface despite changes in the lidar's pose. Having features spread all across the scene leads to better estimation stability.

To evaluate this last point, we compare the proposed method (front-end and back-end) against a hybrid method that uses our back-end in association with the front-end of [10] in

Avg. error initial guess	Estimated translation error (m)	Estimated rotation error (°)
0.17 m, 1.74 °	10.5e-3 ± 6.34e-3	0.035 ± 0.037
0.52 m, 3.47 °	11.2e-3 ± 6.94e-3	0.047± 0.053
0.86 m, 8.55 °	10.8e-3 ± 6.92e-3	0.062 ± 0.142
1.22 m, 25.1 °	16.3e-3 ± 18.7e-3	0.125 ± 0.326

TABLE 4.5: Quantitative results for extrinsic calibration in a 50-run Monte Carlo simulation. The different rows correspond to different levels of error on the initial guess used to estimate the LiDAR-IMU geometric transformation. The trajectories used last 19.6 s (95.2 m-long on average), and have the same velocity characteristics as the "Fast" trajectories of our odometry set-up (Table 4.1).

simulated environment. The proposed method leads to an RMSE pose error of 0.087 m and 0.088 °, while the hybrid method resulted in an error of 0.431 m and 0.220 °. These numbers have been computed over a 50-run Monte Carlo simulation. The average trajectory length is 95.2 m, the average velocity 4.86 m/s, and the average angular velocity 125 °/s.

4.6.3 Simulation - calibration

This set-up aims to evaluate the accuracy of the proposed method quantitatively when used for extrinsic calibration between a LiDAR and an IMU. These series of experiments were run with different error levels of initial guess. Table 4.5 shows the error of the calibration estimates. We can see that the estimates' errors stay small despite an increasingly bad initial guess. The scenario with the worst initial guess (fourth row of Table 4.5) leads to an error slightly bigger than in the three other scenarios. Nonetheless, in such a case, it is possible to run a second iteration using the first iteration's calibration estimate as the initial guess. Doing so, the worst initial guess scenario leads to final calibration errors of 10.8e-3 m, and 0.031 ° after the second iteration of IN2LAAMA. Experiments have also been conducted using longer data recordings, 39.2 s, with the same initial guess as in the first row of Table 4.5. The average errors over 50 runs are reduced to 7.6e-3 m, and 0.024 °. These results demonstrate similar accuracy as our preliminary work [19] that relies on observing a calibration target made of at least three non-coplanar planes.

4.6.4 Real-data - Localisation and mapping

Multiple datasets have been collected with our sensor suite inside the facilities of the University of Technology Sydney. These datasets contain per-point timestamps, and have been made publicly available[3]. Moreover, to demonstrate the versatility of the proposed method, we have applied IN2LAAMA to the MC2SLAM dataset [50]. This dataset has been selected because it contains timestamps for every single LiDAR point.

As mentioned above, our sensor suite comprises a *Velodyne VLP-16* and a low-cost *Xsens MTi-3* IMU. The *snark* driver[4] and the ROS Xsens driver[5] were used to collect the LiDAR and IMU data, respectively. Lidar points and IMU measurements were logged with their associated timestamps. There is no explicit hardware or software mechanism for synchronisation between LiDAR and IMU data.

For quantitative comparisons, we chose to use metrics related to the planarity of planes in the environment as per none of the datasets used contains ground truth. When available, we overlay the map generated with the blueprints of the building or satellite images of the area. In the presence of loop closure, we compute the amount of drift without the loop-closure.

4.6.4.1 Indoors

This set-up aims to benchmark IN2LAAMA against our preliminary work [15] (no IMU factors) and the method in [10]. Real indoor datasets from two different locations have been used for this evaluation: a lab environment and a staircase between floors. In both cases, we show that the proposed method outperforms both [15] and [10].

For the lab environment, Fig. 4.10 shows the map estimated by IN2LAAMA and Table 4.6, first three rows, shows the quantitative comparison between the maps obtained from the different methods. The metric used is the RMS point-to-plane distance between the 3D-LiDAR points belonging to a plane and the corresponding plane. The planes are

[3]https://github.com/UTS-CAS/in2laama_datasets
[4]https://github.com/acfr/snark
[5]http://wiki.ros.org/xsens_driver

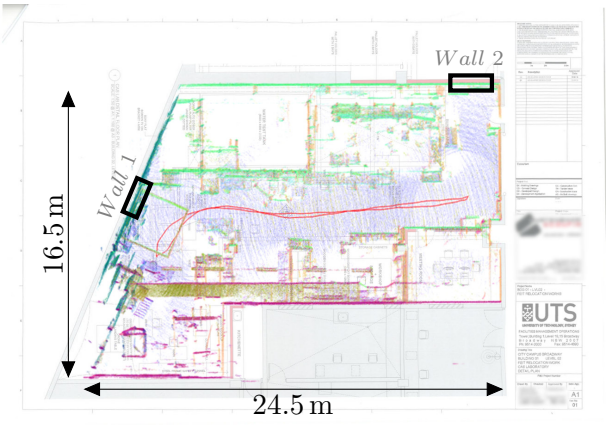


FIGURE 4.10: Lab environment map generated with IN2LAAMA overlaid over digitalised pre-construction blueprints (the blueprint does not capture reality accurately due to structural differences with the original plans and furniture/equipment present in the lab). The estimated trajectory, in red, starts and ends on the left-hand side of the map. Note that the trajectory is 6-DoF and the map is 3D. The points are coloured with the values of the post-processed normals based on the 100 closest neighbours.

Dataset (plane used)	RMS point-to-plane distance (mm)		
	[10]	[15]	IN2LAAMA
Lab (floor)	24	19	**16**
Lab (wall 1)	20	13	**11**
Lab (wall 2)	16	**15**	16
Staircase (wall)	81	31	**10**

TABLE 4.6: Quantitative comparison on real data. The values shown correspond to RMS point-to-plane distance between map points and the corresponding plane. Note that the IN2LAAMA and [15] are offline frameworks, whereas [10] operates in real-time.

estimated by running a principal component analysis on the manually segmented points. The estimated trajectory is 40.4 m-long and has an average velocity of 1.02 m/s. Without loop-closure, the proposed method accumulates a drift of 0.11 m and 0.61 °. In this first dataset, the motion is not aggressive, and the use of IMU factors does not impact the final estimate significantly. All the benchmarked methods perform similarly and lead to RMS point-to-plane distances inferior to the LiDAR noise specification (±3 cm).

The staircase dataset is more challenging because of the nature of the motion (dynamic with strong rotations) and the weak geometric information contained in some of the collected LiDAR scans. The different estimated maps and trajectories are displayed in Fig. 4.11. Irrespectively, whether it is only to help motion distortion correction as in [15] or to also fully constrain the pose-graph optimisation (IN2LAAMA approach), tightly

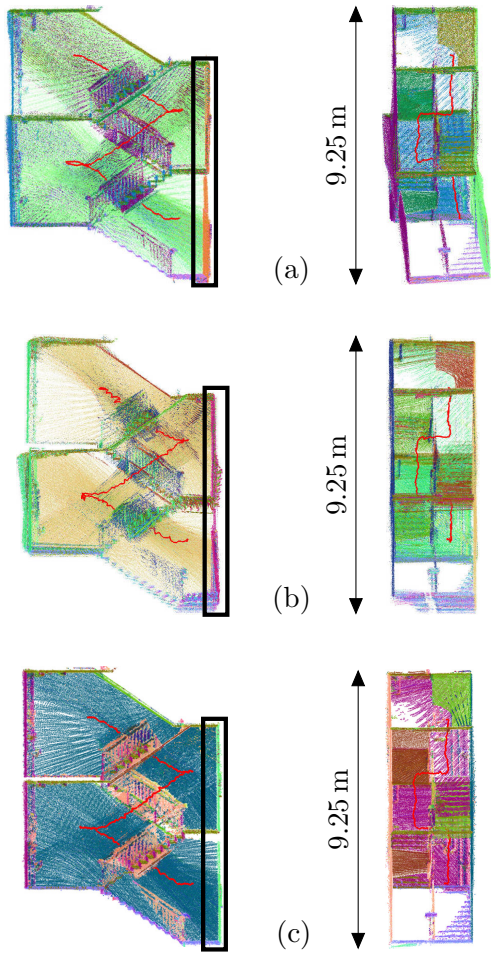


FIGURE 4.11: Staircase maps and trajectories generated with: [10] in (a), [15] in (b), and IN2LAAMA in (c). The black rectangles highlight the wall used for the quantitative comparison. The points are coloured with the values of the post-processed normals based on the 100 closest neighbours.

integrating inertial information in the trajectory estimation leads to greater mapping accuracy than LiDAR-only techniques. The constant velocity motion assumption used in [10] reaches its limits in this kind of scenarios. To provide a quantitative evaluation of the maps, point-to-plane distances are computed for points belonging to the same wall across the different floor levels (black rectangles in Fig. 4.11). The results are shown in the last row of Table 4.6.

Note that our framework uses extra information (IMU readings) in comparison to [10], and the incrementally built batch optimisation is not running in real time, thus the comparison is not totally fair. Table 4.7 shows the computation time and memory usage for the real-data experiments. We differentiate between the amount of memory used by the non-linear

Dataset (length)	N_e	Mem. data/prog.	Mem. optimiser	Exec. time
Lab (41 s)	100	4.70 GiB	0.59 GiB	950 s
Staircase (35 s)	10	4.20 GiB	0.58 GiB	1306 s
Outdoor (85 s)	100	6.45 GiB	5.77 GiB	4699 s

TABLE 4.7: Memory consumption and execution time. Not that the outdoor dataset is collected with a Velodyne HDL-32 that produce around four times as much data as the VLP-16 used in the other experiments. For the outdoor dataset, the optional ICP tests to validate/reject loop-closures have been activated. It represents 1588 s of the overall execution time.

least-square optimiser and the rest of the memory consumption (data, GPMs, program, etc.). While Ceres provides smart implementations of commonly used iterative solvers, its design is not optimised memory-wise for very large numbers of residuals as the evaluation of the system's Hessian matrix requires considerable memory allocations. A potential solution to greatly reduce the memory load is to directly compute the Hessian matrix as done in [31]. With further engineering efforts, the execution time can be substantially reduced as well; parallelisation on GPU could be applied to different operations (e.g. feature extraction, inertial data upsampling, residual and Jacobian computations, etc.). Note that the proposed framework introduces a principled and accurate full-batch optimisation that leverages the full dataset at the cost of computation time and memory usage. Nonetheless, we believe that, at the cost of accuracy, simple approximations of our framework associated with the aforementioned effective implementation techniques would enable real-time and efficient localisation and mapping methods.

4.6.4.2 Outdoors

As mentioned above, the front-end of the proposed method has been designed for structured geometry. While the geometric features used are largely present in indoor environments, outdoor scenarios can represent a challenge for our feature extraction algorithm.

We have chosen the MC2SLAM dataset [50] to show the performance of the proposed approach in an outdoor environment as it provides per-LiDAR-point timestamps. The data have been acquired by a *Velodyne HDL-32* LiDAR and its built-in IMU mounted on top of a car that is driven around a University campus (sequence "campus_drive" of [50]). Fig. 4.12 shows the map generated by IN2LAAMA with loop-closure. As no ground

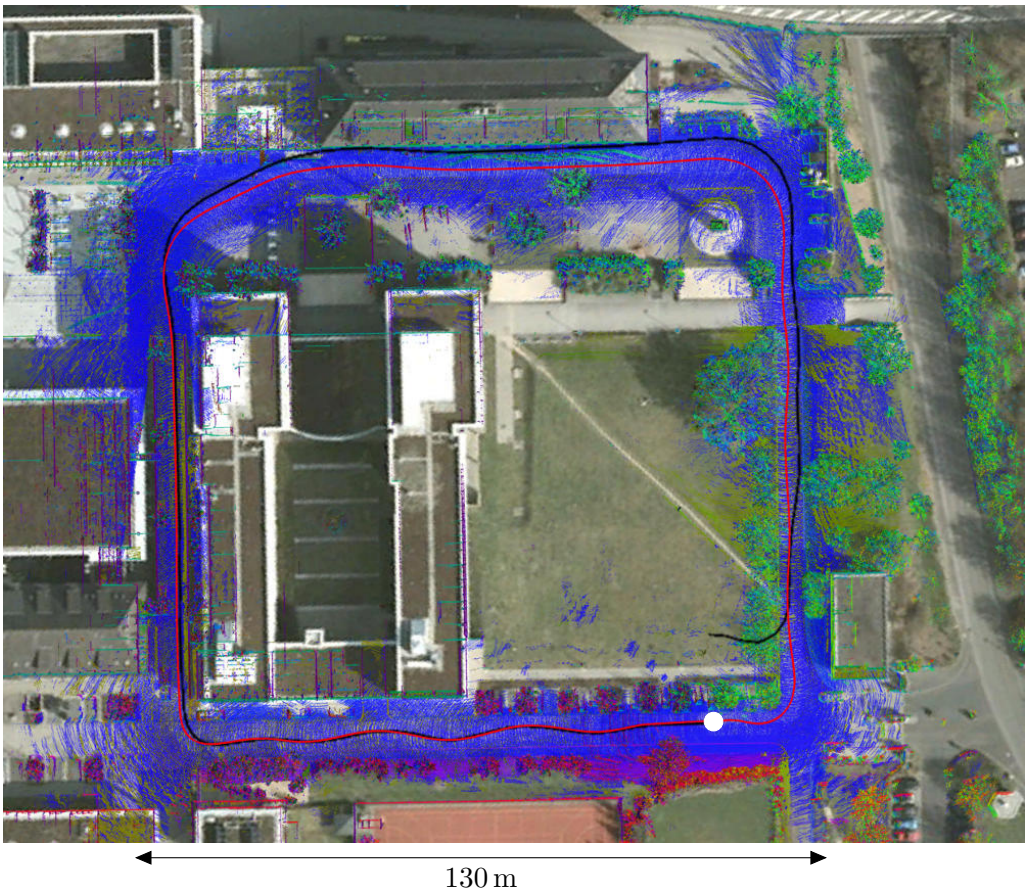


FIGURE 4.12: Map generated by IN2LAAMA in outdoor environments (MC2SLAM dataset [50], sequence "campus_drive") superposed over the corresponding Google Earth image. The trajectory in red is estimated with IN2LAAMA and the one in black with A-LOAM. The white dot represents the start of the trajectories. The points are coloured with the values of the post-processed normals based on the 100 closest neighbours.

truth is given with the dataset, we overlay the map onto the corresponding Google Earth[6] image. The estimated trajectory is 409 m long and lasts 85.4 s. Without loop closure, the accumulated drift of the proposed method is 2.45 m (mostly on the vertical axis: 2.34 m), and 1.12 °. The proposed method outperforms [10] that accumulates a significant drift of 9.78 m and 33.4 ° along the recording.

At the start of the dataset, due to the translation-only trajectory, the accelerometer biases are not observable; the car is driven in a straight line for few meters (before starting a series of turns). Without any additional constraint on the accelerometer biases (Section 4.5.4) and given wrong initial orientation ($\mathbf{R}_W^{\tau_0}$ arbitrarily flipped up-side-down), the estimated

[6] https://www.google.com/earth/

state converges toward wrong biases values (magnitude of $2g$). The extra factor on $\hat{\mathbf{b}}_f^0$ provides the constraint required to contain the estimation error on the accelerometer biases while the lack of motion variations prevents the estimation from converging toward the true value of the state.

4.6.5 Real-data - Calibration

Finally, this set-up aims to evaluate the accuracy of the extrinsic calibration performed by the proposed framework. To do so, we benchmark our method against a "chained calibration" using an extra sensor, an RGB camera (Intel Realsense D435). The "chained calibration" computes IMU-camera and camera-LiDAR extrinsic calibrations and compounds them together to obtain the IMU-LiDAR geometric transformation. The intrinsic camera calibration has been performed with *RADOCC* [96]. The IMU-camera geometric transformation has been estimated with *Kalibr* [93] by moving the sensor suite in front of a static calibration pattern. The camera-LiDAR extrinsic calibration is the result of the minimisation of point-to-plane distances of LiDAR points belonging to a checkerboard itself characterised by plane equations in the camera frame. The data used to estimate this last "link" are static to avoid the issues of motion distortion and time synchronisation.

The two calibration pipelines are executed independently. Fig. 4.13 displays the map estimated during the IN2LAAMA calibration procedure. Then, the evaluation is conducted by running the proposed method for localisation and mapping (on another dataset) based on the calibration parameters obtained from the two calibration pipelines.

The aggressive nature of the trajectory (RMS linear and angular velocities of 0.24 m/s and 81.3 °/s) emphasises the need for good calibration to estimate an accurate map. In the resulting maps, average (over six non-coplanar planes) RMS point-to-plane distances are computed between 3D points and their associated planes in the scene. The chained calibration leads to a mean RMS point-to-plane distance of 58 mm while IN2LAAMA's map displayed more crispness with an average RMS point-to-plane distance of 27 mm.

The quality of the calibration results depends on the quality of the IMU readings as well as the environment and trajectory used for calibration. With IN2LAAMA's front-end

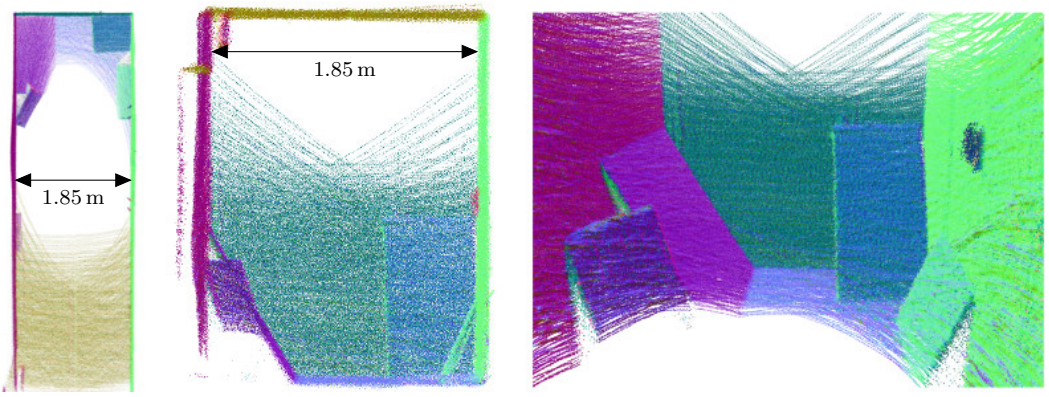


FIGURE 4.13: Map generated by IN2LAAMA during the calibration of the sensor suite. From left to right: top view, side view, first-person view. The points are coloured with the values of the post-processed normals based on the 100 closest neighbours.

being built upon planar an edge features the trajectory of the LiDAR need to allow the frame-to-frame registration of minimum three non-coplanar planes or non-colinear edges to constrain the LiDAR pose estimation properly. As demonstrated in [117], some trajectory types do not lead to observable calibration parameters. The ideal trajectories are random paths that stimulate the 6-DoF of the IMU.

4.7 Conclusion

This chapter introduced INertial Lidar Localisation Autocalibration And MApping; a probabilistic framework for LiDAR-inertial localisation, mapping, and extrinsic calibration. The proposed method aims to deal with the motion distortion present in LiDAR scans without the need for an explicit motion model. The key idea is to use GPMs to allow precise characterisation of the system's motion during each LiDAR scan. The frame-to-frame scan registration is performed with a full batch on-manifold optimisation based on point-to-plane, point-to-line, and inertial residuals. The integration of IMU factors allows us to add robustness to highly dynamic motion. Extensive experiments have been conducted to demonstrate the performances of IN2LAAMA both on simulated and real-world data. A comparison with the state-of-the-art LiDAR localisation and mapping algorithm shows that our method performs better in diverse environments. While providing more accurate results, the current implementation of IN2LAAMA does not allow real-time operations.

Many applications (such as 3D mapping as a service) can leverage such computational intensive framework as it is. However, the proposed framework has the potential to be the baseline to develop more efficient LiDAR-inertial estimation frameworks for real-time operations. To this end, one can easily think about mechanisms such as local maps [118], sliding window optimisation [119], graph sparsification with marginalisation, parallel computation on GPU, etc.

While this application only leverages one supplementary sensor (a LiDAR in addition to an IMU), the asynchronous and continuous nature of the GPMs associated with the factor-graph formulation of IN2LAAMA allows for the fusion of any number of sensors that can provide information (directly or indirectly) about the systems pose or dynamics.

The related future work includes:

- The exploration of different strategies and simplifications to make the method computationally efficient.
- The integration of a more robust and efficient loop closure detection mechanism.
- The addition of other modalities in this localisation and mapping framework.

Alternative map representations such as using surfels could also be investigated to integrate frame-to-model constraints in the optimisation and to improve the front-end robustness in weakly structured environments.

Chapter 5

IDOL: IMU-DVS Odometry using Lines

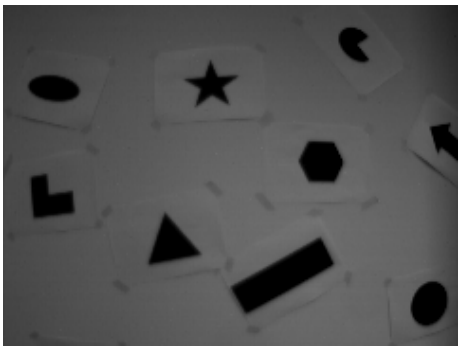
5.1 Introduction

This chapter presents a framework for event-based visual-inertial odometry based on the use of the GPMs presented in Chapter 3. This work corresponds to the contributions presented in [120].

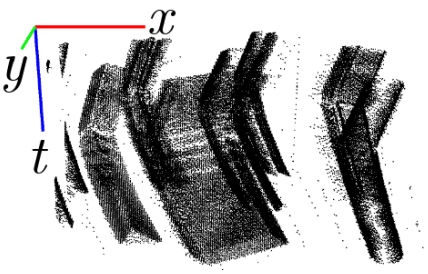
Event-cameras (also called DVS's) and traditional cameras differ in the nature of the data produced despite looking alike from the outside (Fig. 5.1). Traditional frame-cameras output images at a fixed frequency. Each image consists of an array of pixels containing the intensity information about the light coming from the observed scene. Event-cameras are bio-inspired sensors that generate asynchronous streams of events instead of regular images (Fig. 5.2. Individually, the pixels of a DVS trigger events upon changes of intensity according to a given threshold. Consequently, in a static environment, events are generated only when the camera is moving. Note that additionally to the position in the image and a timestamp, each event is characterised by a *polarity* which is a boolean value that encode the direction of the intensity change.

While this novel type of sensor has great advantages (high dynamic range, low latency, high bandwidth data, etc., cf. Fig. 5.3), it comes with new challenges for the vision-based

FIGURE 5.1: Pictures of DVS sensors. On the right is the DAVIS240 (image source: inivation.com). On the right is a propototype as shown in [66].



(a) Greyscale image of environment [121] (not used in IDOL)



(b) Spatio-temporal view of raw stream of events

FIGURE 5.2: Example of data obtained with an event camera (from `shapes` of [121]).

state estimation community. Particularly, traditional estimation frameworks are generally not designed for asynchronous and high-bandwidth sensory information. The proposed method aims at estimating the ego-motion of an event-based visual-inertial system. To this end, as shown in Fig. 5.4, line segments are detected in the event data provided by the camera, and both the system's trajectory and the position of the 3D lines are simultaneously estimated. Addressing the asynchronicity of the event stream, GPMs are used to associate each event with inertial measurements. The state is then estimated by means of a discrete-state batch on-manifold optimisation that accounts for the events individually.

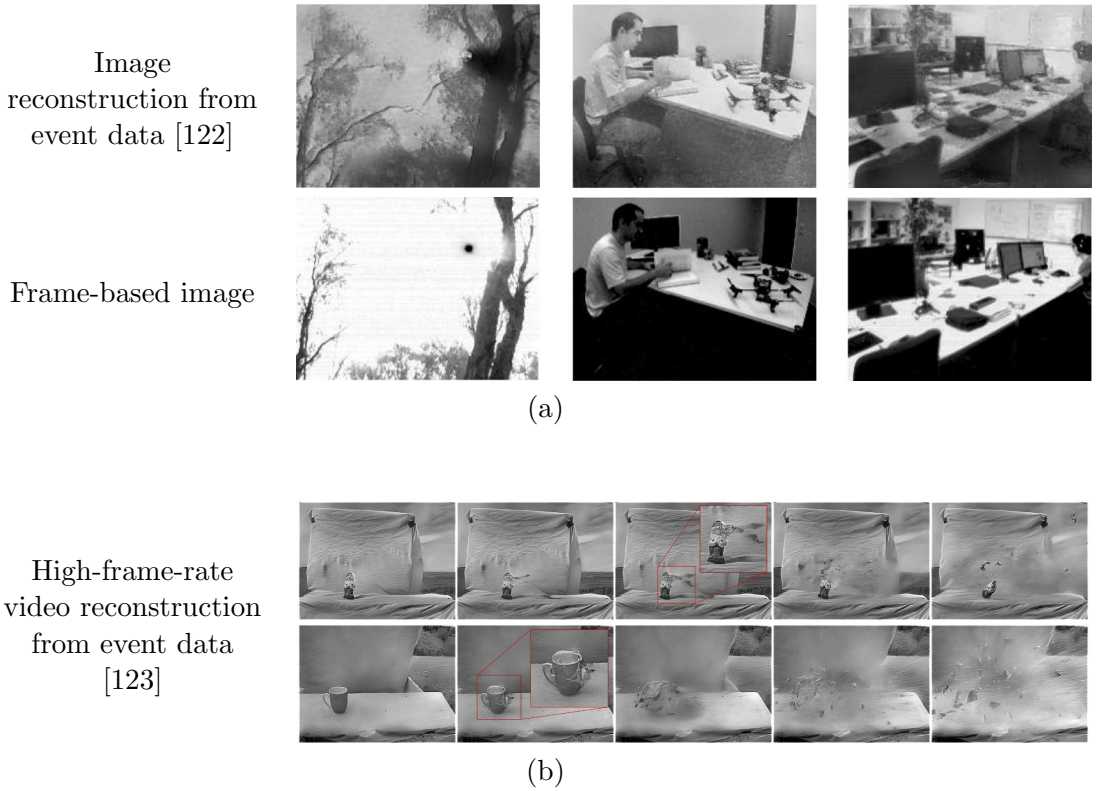


FIGURE 5.3: Illustration of applications that leverage the advantages of event cameras (image sourced from the corresponding papers). In (a) we can see the image reconstruction performed with [122]. This method uses convolutional gated recurrent unit [124] in a neural network named FireNet. Thanks to the HDR abilities of event-cameras, the reconstructed images contain details invisible in the corresponding frame-based images (e.g., the areas under the desks in the two lest-most images). (b) shows different frames from high-frame-rate videos of bullets shot through various objects reconstructed from event data with [123]. This approach is essentially a U-shaped neural network [125] that leverages long short-term memory convolutional layers [126]. The reconstructed videos display details that cannot be acquired with standard frame-based cameras.

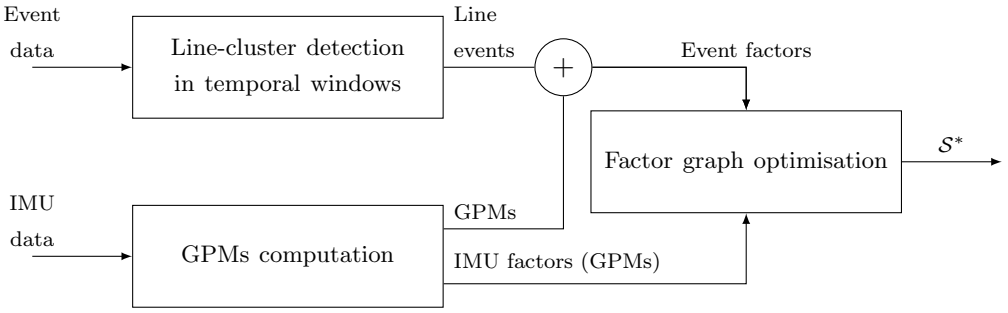


FIGURE 5.4: Overview of IDOL with $\mathcal{S}^*$ the estimated state. The "event factors" correspond to diverse residuals detailed in this chapter: event-to-line, attraction force, repulsion force.

5.2 Method overview

Let us consider a rigidly mounted event-camera and a 6-DoF IMU. The camera and IMU reference frames at time t_i are respectively noted $\mathfrak{F}_C^{t_i}$ and $\mathfrak{F}_I^{t_i}$. The rotation matrix $\mathbf{R}_I^C$ and the translation vector $\mathbf{p}_I^C$ characterise the pose of $\mathfrak{F}_C^{t_i}$ in $\mathfrak{F}_I^{t_i}$. Homogeneous transformations, as introduced in Subsection 4.2.1, will be used in this chapter.

As for IN2LAAMA (Chapter 4), the 6-DoF IMU acquires proper acceleration $\tilde{\mathbf{f}}_{\mathfrak{t}_q}$ and angular velocity $\tilde{\boldsymbol{\omega}}_{\mathfrak{t}_q}$ measurements at time $\mathfrak{t}_q$ $(q = 1, \cdots, Q)$. These readings are combined together into GPMs according to Chapter 3. The event-camera data is collected as an asynchronous stream of events $\mathbf{e}^i = \begin{bmatrix} e_x^i & e_y^i \end{bmatrix}$ at time t_i, with e_x^i and e_y^i being the pixel coordinates in the image space. Note that in this work, we do not consider the polarity of the events. This stream is arbitrarily organised in M windows of N events. A 3D point is projected into the image space with the function $\pi(\bullet)$ according to a pinhole camera-model:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \pi\left(\begin{bmatrix} x & y & z \end{bmatrix}^\top\right) = \begin{bmatrix} \frac{u'}{w'} \\ \frac{v'}{w'} \end{bmatrix} \qquad \text{with} \qquad \begin{bmatrix} u' \\ v' \\ w' \end{bmatrix} = \mathbf{K} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \tag{5.1}$$

and $\mathbf{K}$ being the 3×4 camera matrix.

The system's IMU orientation $\mathbf{R}_W^m$, position $\mathbf{p}_W^m$, and velocity $\mathbf{v}_W^m$ are estimated at the timestamp τ_m of the first event of each window $(m = 1, \cdots, M)$ with respect to the fixed world frame $\mathfrak{F}_W$. The proposed method also estimates corrections, $\hat{\mathbf{b}}_f^m$ and $\hat{\mathbf{b}}_\omega^m$, to the bias priors used during the preintegration.

Along the sensor's trajectory, events are clustered into segments that correspond to 3D lines in the environment $\mathbf{L}_W^l$ (front-end). The accumulation of these event-line associations $\alpha^i = \{\mathbf{e}^i, t_i, \mathbf{L}_W^l\}$ form the set $\mathcal{A}$. The positions of the lines are estimated simultaneously to the IMU pose and velocities mentioned above. Each line is parameterised by its extremities

in the world frame $\mathfrak{F}_W$ with two 3D points ${}^{l}\mathbf{l}_{aW}$ and ${}^{l}\mathbf{l}_{bW}$ as

$$\mathbf{L}_W^l = \{{}^{l}\mathbf{l}_{aW},\ {}^{l}\mathbf{l}_{bW}\}, \tag{5.2}$$

with $l = 1, \cdots, L$. While this representation over-parameterises 3D lines, it allows the proposed method to fit the lines' extremities to the actual line segments through a mechanism of attraction/repulsion detailed in Section 5.3.2.

The proposed method estimates the state $\mathcal{S} = (\mathbf{R}_W^{\tau_1}, \cdots, \mathbf{R}_W^{\tau_M}, \mathbf{p}_W^{\tau_2}, \cdots, \mathbf{p}_W^{\tau_M}, \mathbf{v}_W^{\tau_2}, \cdots, \mathbf{v}_W^{\tau_M}, \hat{\mathbf{b}}_f^{\tau_1}, \cdots, \hat{\mathbf{b}}_f^{\tau_M}, \hat{\mathbf{b}}_\omega^{\tau_1}, \cdots, \hat{\mathbf{b}}_\omega^{\tau_M}, \mathbf{L}_W^1, \cdots, \mathbf{L}_W^L)$ using maximum likelihood estimation that corresponds to the minimisation of the cost function C:

$$\begin{aligned} \mathcal{S}^* &= \underset{\mathcal{S}}{\operatorname{argmin}}\ C(\mathcal{S}), \\ C(\mathcal{S}) &= \sum_{\alpha^i \in \mathcal{A}} \left(\|r_l^{\alpha^i}\|^2_{\Sigma_{\mathbf{r}_l^{\alpha^i}}} + \|r_s^{\alpha^i}\|^2_{\Sigma_{\mathbf{r}_s^{\alpha^i}}} \right) + \sum_{l=1}^{L} \|r_a^l\|^2_{\Sigma_{\mathbf{r}_a^l}} + \\ &\quad \sum_{m=1}^{M-1} \left(\|\mathbf{r}_f^m\|^2_{\Sigma_{\mathbf{r}_f^m}} + \|\mathbf{r}_\omega^m\|^2_{\Sigma_{\mathbf{r}_\omega^m}} + \|\mathbf{r}_I^m\|^2_{\Sigma_{\mathbf{r}_I^m}} \right), \end{aligned} \tag{5.3}$$

with r_l being event-to-line distances for each event-line association in $\mathcal{A}$, r_s and r_a being repulsion and attraction forces between each of the lines' extremities, respectively, $\mathbf{r}_f$ and $\mathbf{r}_\omega$ constraints on the IMU biases random-walk, and $\mathbf{r}_I$ being direct pose and velocity constraints between two consecutive timestamps of the estimated trajectory based on the IMU readings. These factors (back-end) are detailed in Section 5.3. Note that $\Sigma_\bullet$ is the covariance matrix of the variable $\bullet$.

5.3 Back-end

This section describes the event-related and line-related elements of the cost function $C(\mathcal{S})$ presented in (5.3). The IMU and biases factors are common with IN2LAAMA (Chapter 4). These are detailed in Subsections 4.3.1 and 4.3.2. The Jacobians associated to the different residuals of this section are present in Appendix D.

5.3.1 Event-to-line factors

The event-to-line factors correspond to the point-to-line distances between events in the image space and the image projection of the associated 3D lines. Let us consider an event-line association $\alpha^i = \{\mathbf{e}^i, t_i, \mathbf{L}_W^l\}$. The projections $\mathbf{d}_{a_l}^{t_i}$ and $\mathbf{d}_{b_l}^{t_i}$ of the line extremities ${\mathbf{l}_a}_W^l$ and ${\mathbf{l}_b}_W^l$ into the camera image at t_i are obtained using the extrinsic calibration $\mathbf{T}_I^C$, and the GPMs (3.19) and (3.21):

$$\mathbf{d}_{a_l}^{t_i} = \pi({\mathbf{T}_I^C}^\top {\mathbf{T}_W^{t_i}}^\top {\mathbf{l}_a}_W^l) \text{ and } \mathbf{d}_{b_l}^{t_i} = \pi({\mathbf{T}_I^C}^\top {\mathbf{T}_W^{t_i}}^\top {\mathbf{l}_b}_W^l). \tag{5.4}$$

The point-to-line distance residual $r_l^{\alpha^i}$ is then equal to

$$r_l^{\alpha^i} = \frac{\|(\mathbf{e}^i - \mathbf{d}_{a_l}^{t_i}) \times (\mathbf{d}_{b_l}^{t_i} - \mathbf{d}_{a_l}^{t_i})\|}{\|\mathbf{d}_{b_l}^{t_i} - \mathbf{d}_{a_l}^{t_i}\|}. \tag{5.5}$$

5.3.2 Line attraction and repulsion factors

The two-3D-point representation (5.2) is an over-parameterisation of an infinite 3D line and the event-to-line factors do not fully define the position of the line points. In other words, without additional constraints, there is ambiguity on the estimated state as an infinite amount of point pairs can characterise the same line. To address this issue, we introduce an attraction/repulsion strategy that fits the line extremities to the cluster of events, therefore constraining the extra DoFs of the two-point line parameterisation. Intuitively, the two components of this mechanism can be thought of as, on the one side, an inherent attraction force between the line extremities, and on the other, a set of forces generated by the events pushing the extremities apart as illustrated in Fig. 5.5.

Formally, the attraction component is implemented as a residual equal to the square-root of the pixel-distance between the line extremities after projection into the image space at τ_m (the timestamp of the window in which the line has first been observed):

$$r_a^l = \sqrt{\|\mathbf{d}_{b_l}^{\tau_m} - \mathbf{d}_{a_l}^{\tau_m}\|}. \tag{5.6}$$

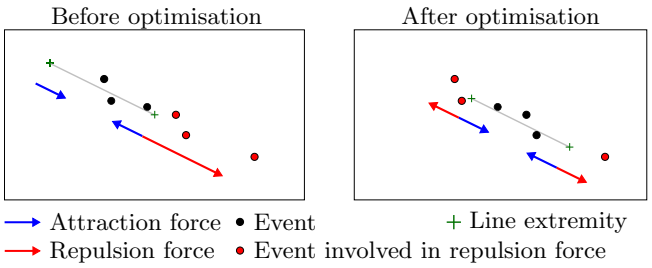


FIGURE 5.5: Illustration of the attraction/repulsion mechanism used to fit the line extremities to the observed segment, as well as to constrain the over-parameterised two-point line representation. The estimated 3D lines extremities are projected into the image space. The extremities are subject to a constant attraction force toward one another. The events that are "outside" the line projection induce a repulsion force that push the extremities apart. After optimisation, the estimated line fits the actual segments.

The repulsion component is generated by the points around the extremities of the line. Given an event-line association α^i, the position d^i of $\mathbf{e}^i$ along the line $\mathbf{d}_{a_l}^{t_i}/\mathbf{d}_{b_l}^{t_i}$ is computed as

$$d^i = \frac{(\mathbf{e}^i - \mathbf{d}_{a_l}^{t_i})^\top (\mathbf{d}_{b_l}^{t_i} - \mathbf{d}_{a_l}^{t_i})}{\|\mathbf{d}_{b_l}^{t_i} - \mathbf{d}_{a_l}^{t_i}\|}. \tag{5.7}$$

The events that are "outside" the line lead to residuals equal to the distance along the line to the closest extremity:

$$r_s^i = \begin{cases} d^i & \text{if } d^i < 0 \\ \|\mathbf{d}_{b_l}^{t_i} - \mathbf{d}_{a_l}^{t_i}\| - d^i & \text{if } d^i > \|\mathbf{d}_{b_l}^{t_i} - \mathbf{d}_{a_l}^{t_i}\| \\ 0 & \text{otherwise.} \end{cases} \tag{5.8}$$

Note that this approach does not require the front-end to extract line extremities among the event data. The events that are projected outside the estimated line segments automatically generate repulsion forces. Therefore, the positions of the line extremities are best estimated when solely a small number of events generate repulsion forces. As repulsion forces are invariant to the lines' length but correlated to the number of events involved, the attraction forces also need to be somewhat length-invariant to prevent the need for any additional balancing mechanism between the forces. Intuitively, in least-square optimisation problems, the Jacobians of the cost function represent forces that constrain the state estimate. Consequently, the attraction forces are made length-invariant by the use of the

square root in r_a^m, making the magnitude of the attraction force "constant" (preventing long line estimates to be "squashed" if Euclidean norms were directly used).

5.4 Front-end

Similarly to [86], the proposed method considers the stream of events as 3D information, where the first two components are the events' coordinates in the image space, and the third coordinate is the events' timestamps arbitrarily normalised: $\mathbf{x}_E^i = \begin{bmatrix} e_x^i & e_y^i & t_i/c \end{bmatrix}^\top$. The value of c is chosen according to the average level of texture in the scene. The front-end consists of clustering events that are triggered by the same physical line in the environment according to the premise that 3D lines translate to locally planar patches in the 3D spatio-temporal representation of the event stream. At the moment, the presented approach does not aim for real-time operation and thus it uses windows of N events to perform the event clustering. The event data in each window can be seen as a point cloud with the 3D-points' coordinates defined as $\mathbf{x}_E^i$. Normal vectors are estimated for each of the points based on the eigenvectors of the covariance matrix built with the neighbouring points. Points are considered neighbours if their Euclidean distance in the 3D spatio-temporal space is under a certain given threshold. Note that the proposed method does not fit planes throughout the windows. The x and y components of the x-y normals are normalised to unit vectors $\mathbf{n}^i$.

The actual clustering is applied in a region growing fashion inspired by the connected component segmentation implemented in [127]. Colinearity of the normal vectors ($\mathbf{n}^{i\top}\mathbf{n}^j > n_{\text{thr}}$) and the point-to-line distance in the image space ($|\mathbf{n}^{i\top}(\mathbf{e}^j - \mathbf{e}^i)| < e_{\text{thr}}$) are used as the similarity criteria to assert that two neighbouring points belong to the same cluster. Fig. 5.4 and 5.6 show examples of event clusters in the three types of datasets used in Section 5.5. By considering only the x-y normal components over local patches with a region growing algorithm, the proposed method allows for line clustering in a large variety of scenarios that are not restricted by the nature of the system's motion (e.g. constant velocity).

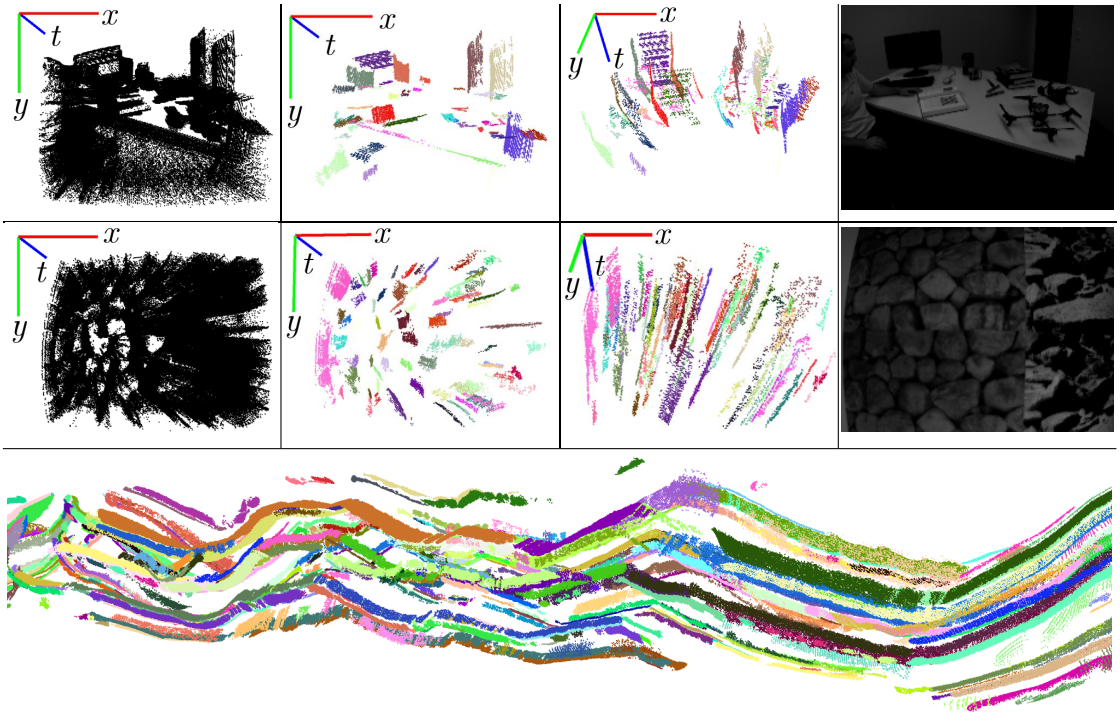


FIGURE 5.6: Examples of line clusters extracted from different dataset. The first two rows (raw event data, line clusters viewpoint A, and B, corresponding greyscale image) correspond to windows of 200k events (value used in our experiments). The bottom row is the cluster extraction on a very large window in `shapes_6dof`.

The segment association between two consecutive windows is conducted by appending the last events of a window to the beginning of the following one, and using the same similarity criteria as described for the in-window clustering. Each connected segment is attached to a line $\mathbf{L}_W^l$ and the event-line associations $\alpha^i = \{\mathbf{e}^i, t_i, \mathbf{L}_W^l\}$ are pushed to the set $\mathcal{A}$. Note that, in the current implementation, the event-to-line association is only performed at the front-end level. There is no additional strategy to associate new segments to existing line estimates. This would greatly improve the accuracy and robustness of the proposed method in scenarios where the same line results in multiple clusters that cannot be matched via the aforementioned procedures, or when lines reappear in the field of view after having left it.

5.5 Experiments

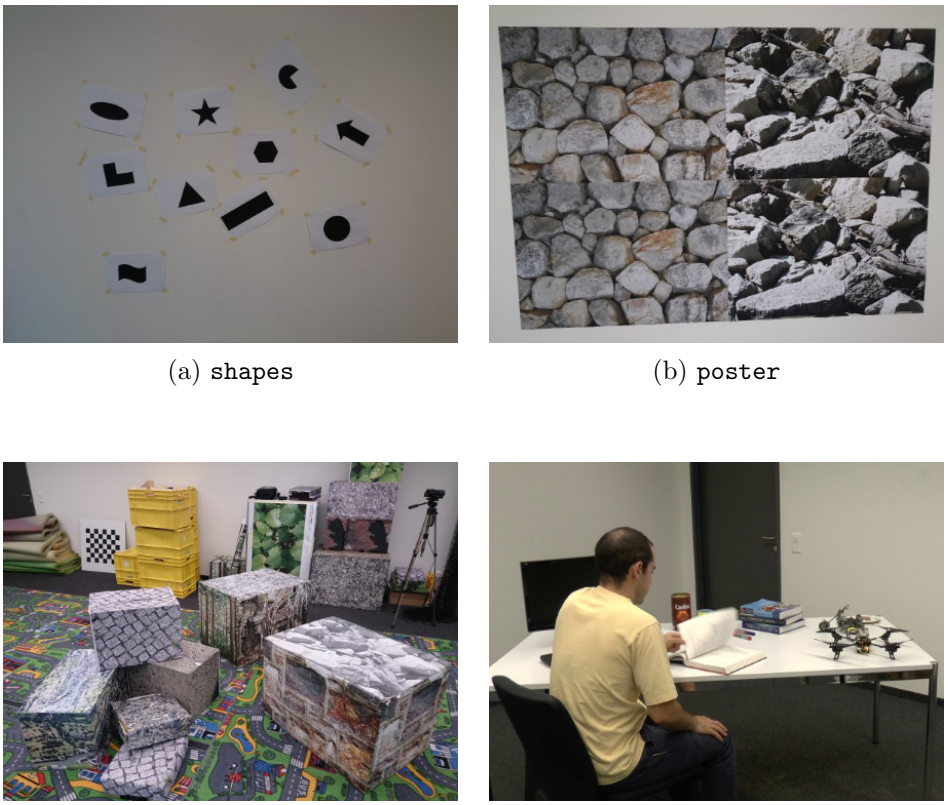
Our implementation uses *Ceres*[1] for the non-linear least-square optimisation (5.3) as in the case of IN2LAAMA (Chapter 4). At the current stage of development, IDOL is computationally expensive as the full batch optimisation is conducted every time a new event-window is processed. The rapidly growing number of residuals leads to prohibiting estimation time of the system's Hessian. We arbitrarily chose to compute the full batch optimisation until $t = 24\,\mathrm{s}$. Intuitively, the longer the full-batch duration, the more robustness to front-end failures. Then we switch to a sliding window optimisation over the last 30 event-windows with the last marginalized pose estimate fixed.

Due to the lack of publicly available event-based VIO algorithms, ROVIO [128] was chosen as a comparison. ROVIO is a light-weight VIO algorithm that operates on traditional intensity images and is built upon an EKF back-end. The front-end of ROVIO extracts patch-based features that are matched based on a photometric error resulting in a semi-dense approach. The system has been shown to be very robust even under very aggressive motions [128].
Since IDOL, in contrast to ROVIO, performs batch optimisation, for a fair comparison we also include results obtained with maplab [129]. It builds a Visual-Inertial (VI) pose-graph using ROVIO and jointly optimises the trajectory and sparse BRISK landmarks [130] using the maplab toolbox.

5.5.1 Datasets and Evaluations

In order to test IDOL in challenging conditions and evaluate the performance compared to state-of-the-art in robust traditional VO, tests on multiple real-world datasets including aggressive motions of the Event Dataset and Simulator [121] were performed (Fig. 5.7). The Event Dataset contains sensor data from a DAVIS240 [66] including the event data, traditional grey-scale images and IMU measurements. In particular, the indoors scenarios `shapes`, `poster` and `dynamic` were chosen as they include 6-DoF ground-truth from an indoor positioning system. Both translation-only and full 6-DoF versions of these datasets

[1] http://ceres-solver.org/



(a) `shapes` (b) `poster`

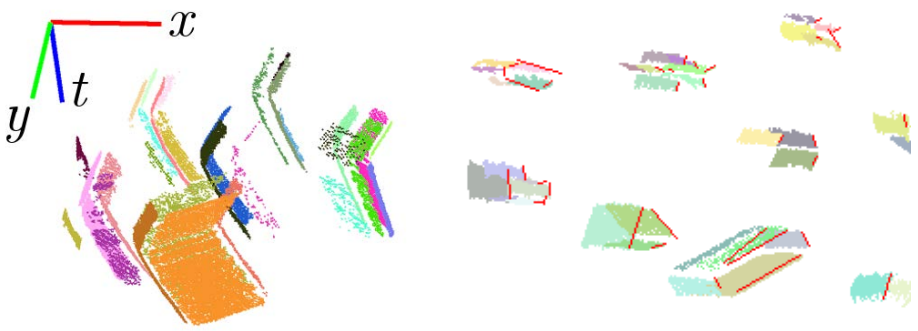
(c) `boxes` (d) `dynamic`

FIGURE 5.7: Subset of the different environments present in the Event Dataset and Simulator [121].

have been used. Rotation-only datasets were omitted since translation is necessary for a good observability of the scene depth [71]. In its current state, our front-end does not offer enough robustness to address the `boxes` scenarios of [121] because of their very high level of texture in the scene. In addition to reporting qualitative results of the state progression and RMSEs of aligned trajectories, the evaluation is also performed using a trajectory-segment based approach [131] with segment lengths corresponding to $\{10, 20, 30, 40, 50\}$ % of the trajectory length. Fig. 5.8 shows an example of the line segmentation and estimated line projection in the image space.

5.5.2 Results

Fig. 5.9-5.11 depict the state progression of the ground-truth and estimations of ROVIO, ROVIO+maplab on `shapes_6dof`, `poster_translation` and `dynamic_translation`, and



(a) Event clustering of line segments

(b) Estimated lines, in red, projected in the image space

FIGURE 5.8: Example projection of estimated lines in the image space (`shapes` dataset of [121], same as Fig. 5.2).

IDOL. As shown in these figures, even though there is visible drift in the translation estimation, IDOL is able to estimate the camera pose and velocity, and especially the camera rotation, with a high accuracy. After the initial stages of each of the experiments, we observe that the proposed VO pipeline's translation estimate tends to diverge, which can be explained as a combination of different factors. One can see that the state variables are generally well estimated during the full-batch optimisation that takes places at the initial instants of each experiment and only start drifting seconds after switching to the sliding-window mode. Our implementation could benefit from leveraging the uncertainty of the estimated pose across consecutive windows of optimisation, properly marginalising previous states. We must also consider that, in its current form, the front-end described in Section 5.4 produces rather short line-tracks. Fig. 5.12 depicts the average track length along the datasets. One can observe the correlation between the drop of the track length around $t = 27\,\mathrm{s}$ and $t = 24\,\mathrm{s}$ in `shapes_6dof` and `poster_translation`, respectively, and the sudden increase of the translation errors. Short track lengths and the absence of a strategy to re-detect previously observed lines do not account for a good depth estimation due to the low parallax. Consequently, the translation estimates are notably affected whereas the rotation estimates, not as dependent on the depth estimate of the scene, still perform in a competitive range for small increments. Note that despite a growing drift of the translation estimates likely due to our front-end's weaknesses, IDOL performs accurate velocity estimation all along the trajectory, validating its effectiveness for VIO. Fig. 5.12 also shows that our front-end does not perform equally well across the different datasets

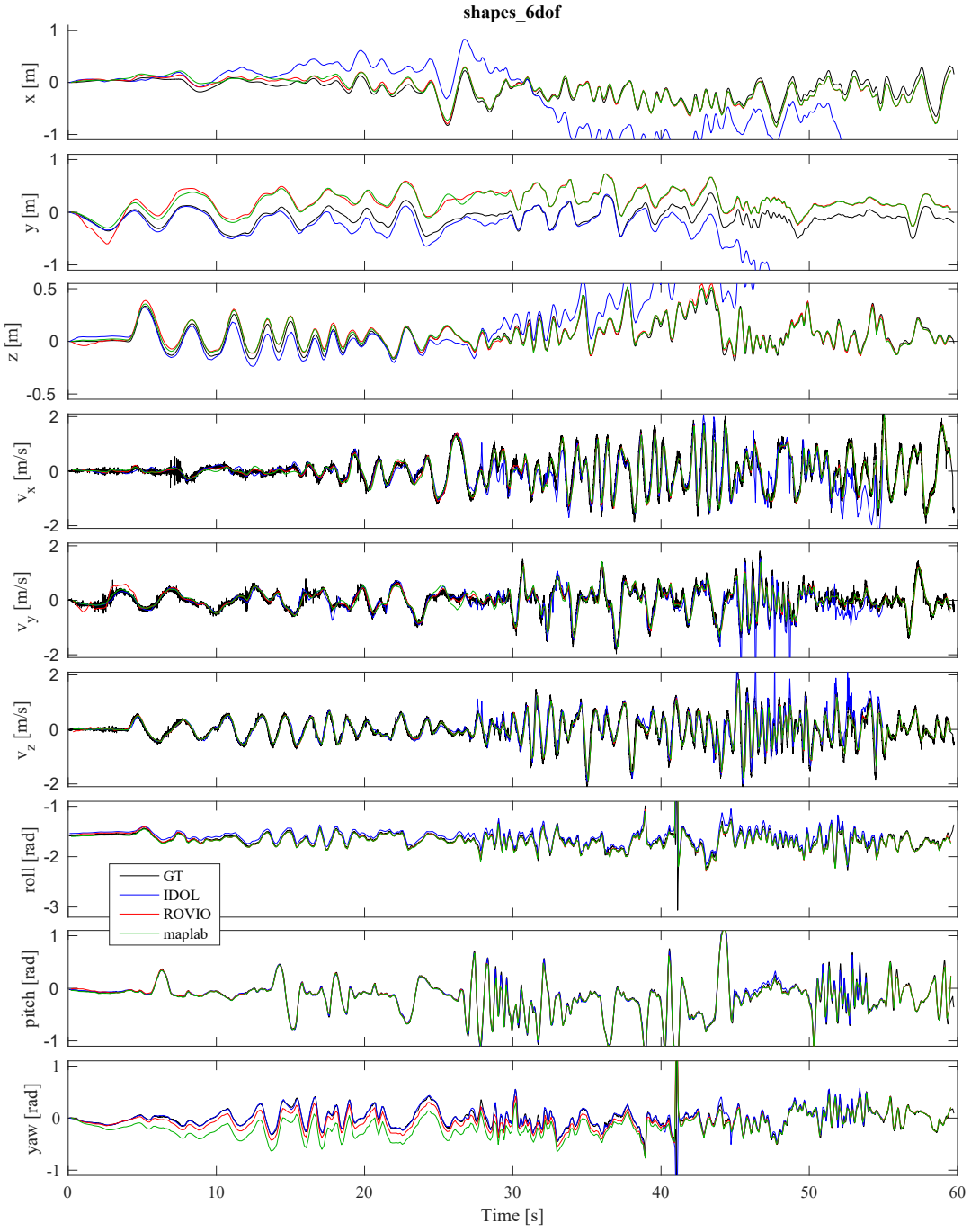


FIGURE 5.9: Pose and velocity estimates' progression of the different algorithms on the `shapes_6dof` dataset.

due to the different levels of noise in event data (due to high texture scenes, cf. Fig. 5.7) and the absence of noise filtering strategy.

Table 5.1 reports RMSE of translation and rotation estimation of all algorithms on each of the datasets. The metric is highly dependent on the translation estimate, as

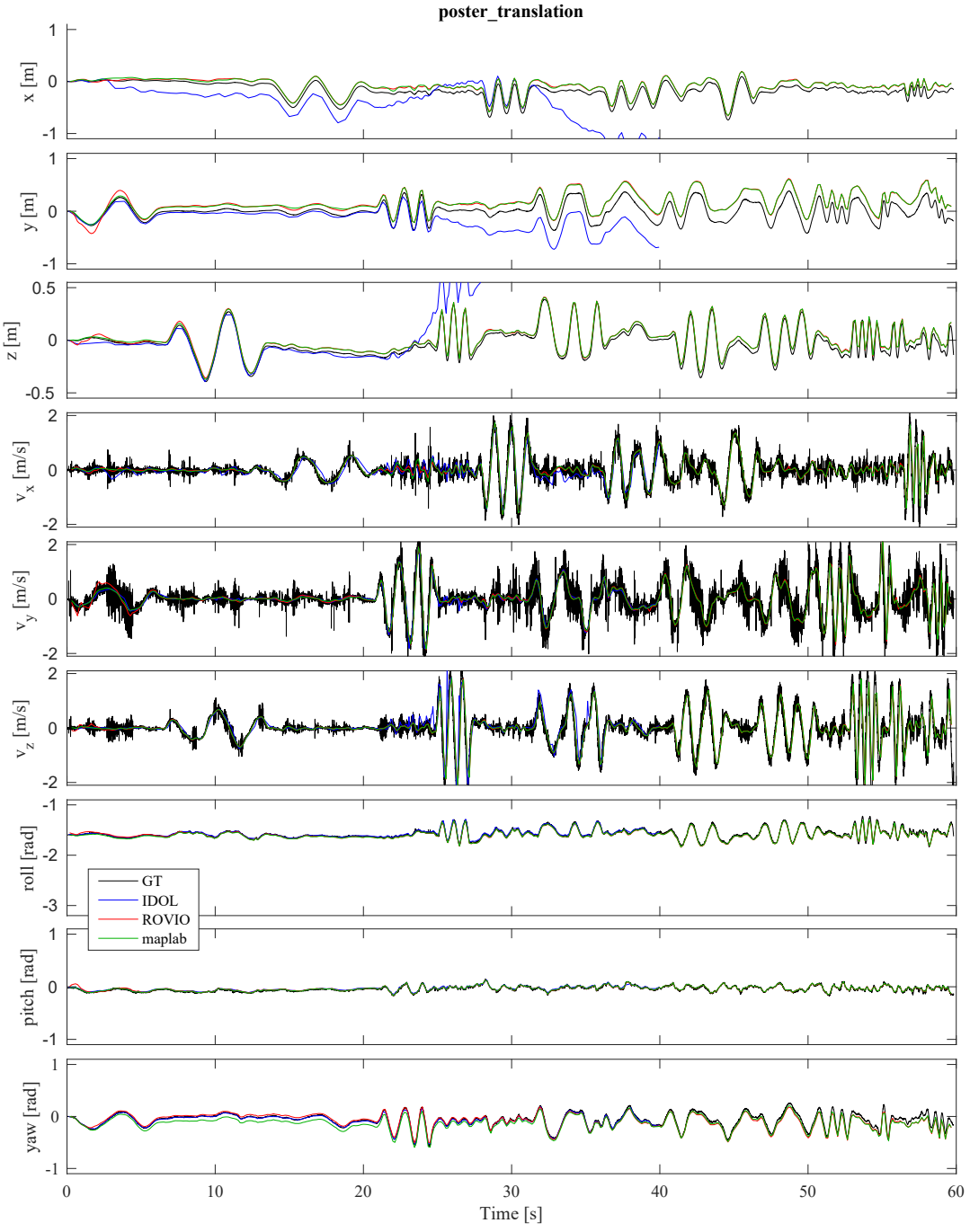


FIGURE 5.10: Pose and velocity estimates' progression of the different algorithms on the `poster_translation` dataset.

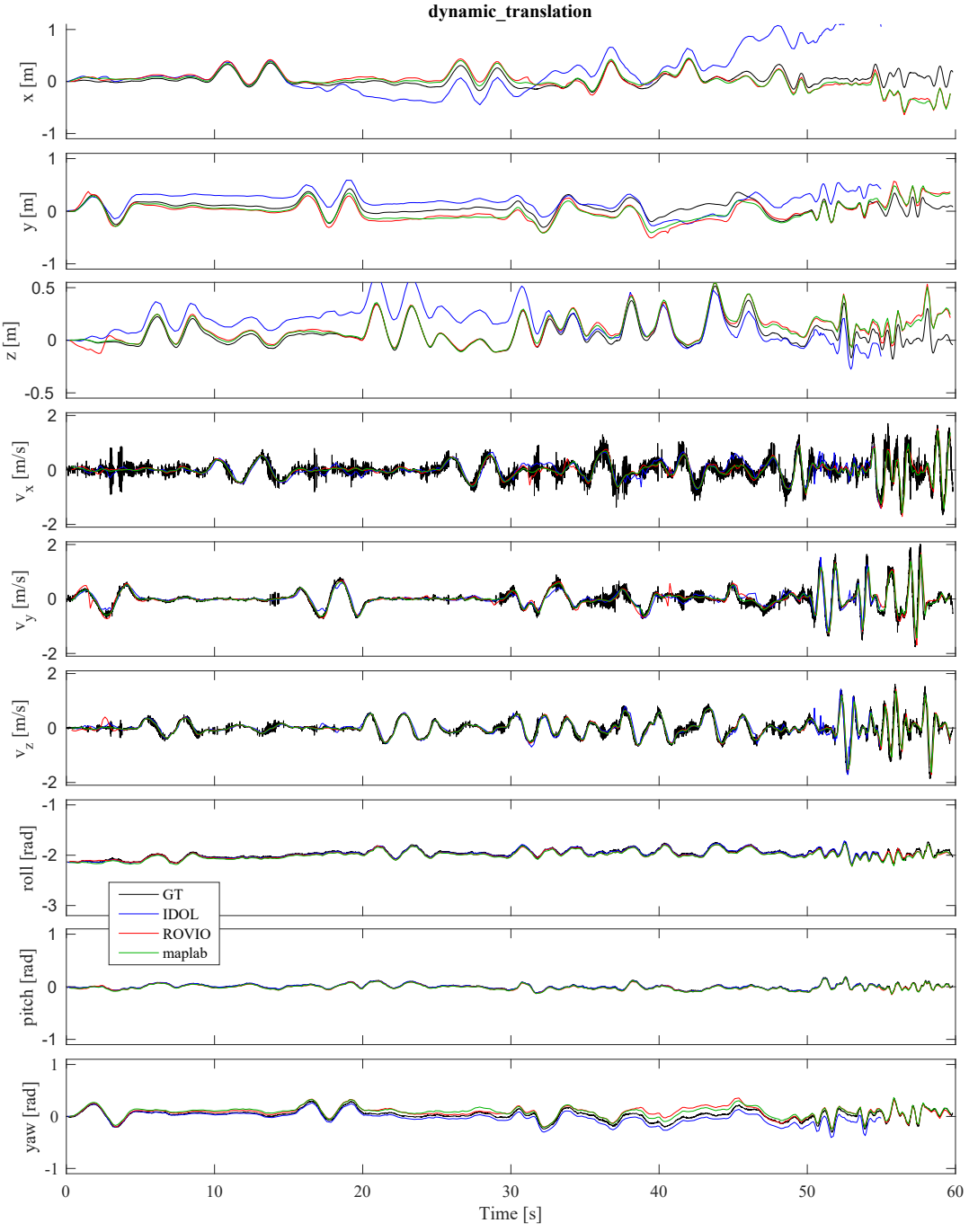


FIGURE 5.11: Pose and velocity estimates' progression of the different algorithms on the `dynamic_translation` dataset.

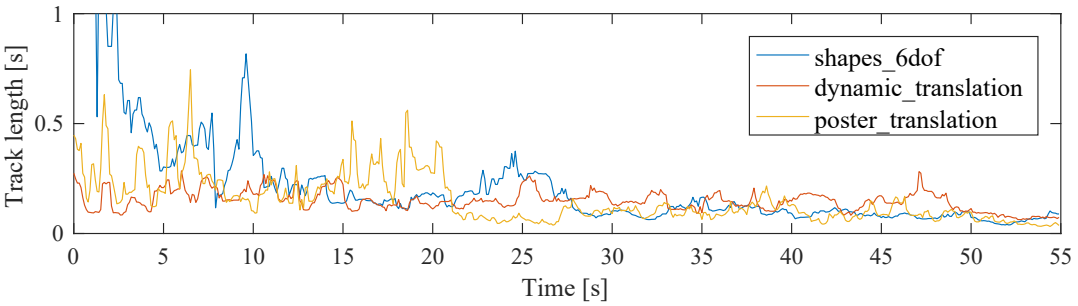
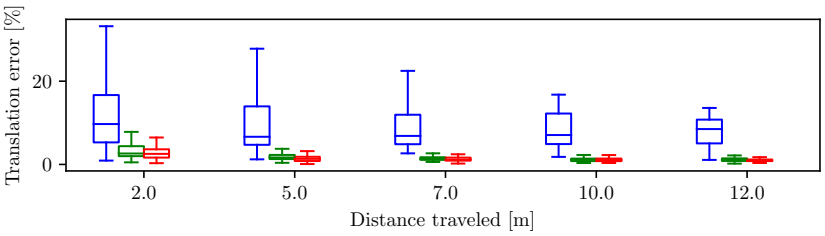
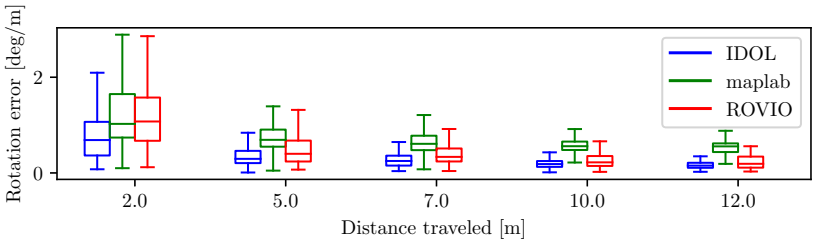


FIGURE 5.12: Average line-track lengths (moving average) across different datasets.

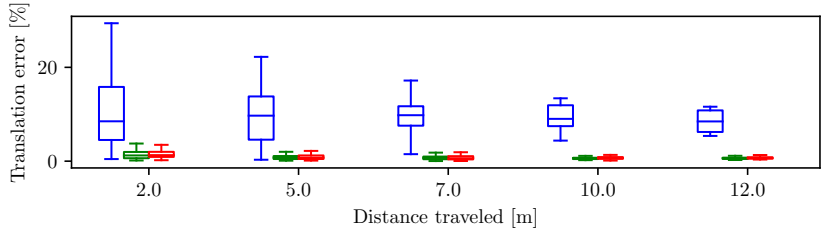
the alignment method used for VIO minimizes the translation error to find a position and yaw offset for the whole trajectory. Accordingly, even if only one component of the translation estimate diverges, the orientation RMSE is highly influenced by the poor trajectory alignment. Consequently, the results depicted in Table 5.1 are obtained by aligning only the 50 first poses of the estimation and only taking the first 40 s into account. Typically, IDOL performs worse than ROVIO and ROVIO+maplab in terms of translation estimation, but still on the same order of magnitude, especially on `shapes_6dof` and `dynamic_translation`. Moreover, competitive results in orientation estimation can be achieved for most datasets. Only `shapes_translation` and `poster_translation` show a worse rotation estimation compared to ROVIO, mainly due to diverged translation estimation (see Fig. 5.9-5.11). More insights into the alignment issue can be found in the segment based evaluation shown in Fig. 5.13. It becomes clear that, in its current state, IDOL can outperform both ROVIO and ROVIO+maplab in incremental rotation estimation, but mainly lacks in translation estimation.

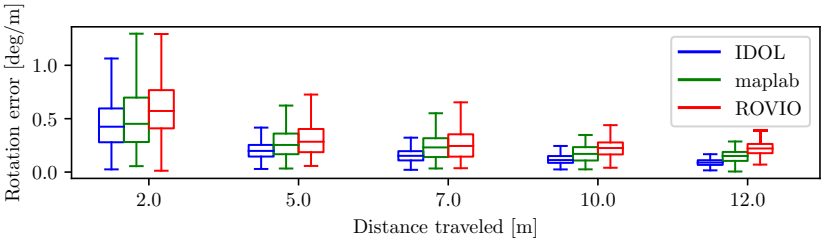
The results displayed above demonstrate the ability of IDOL to perform VO in various real-world scenarios. Overall, the proposed method performs in the same order of magnitude as the compared frame-based methods according to the presented results. However, note that, at the time of writing this thesis, we are using a unrefined implementation of the event-based front-end, in which no noise filtering or re-detection scheme is applied. For instance, the knowledge of the line extremities' position could help the data association in scenarios in which the camera re-observes previously mapped areas. Additionally, no mechanism against outliers have been placed in either the front-end or in the back-end. In this regard, substantial improvements in performance are to be expected from the adoption



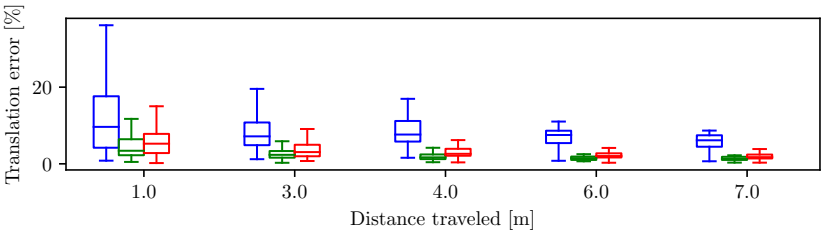


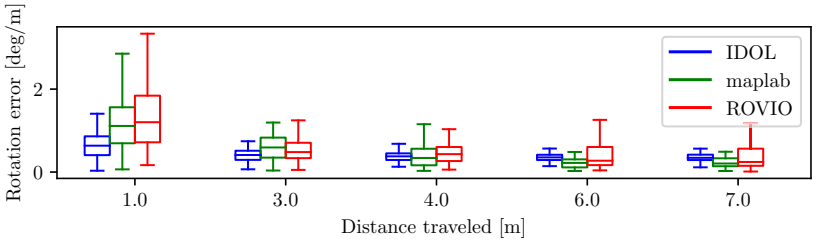
(A) `shapes_6dof`





(B) `poster_translation`





(C) `dynamic_translation`

FIGURE 5.13: Translation and orientation error of the first 40 s using the different algorithms upon different segment lengths.

Method	Dataset `shapes`			
	`6dof`		`translation`	
	e_t	e_r	e_t	e_r
IDOL	0.52	8.35	0.51	18.62
ROVIO	0.34	10.24	0.05	1.56
ROVIO + maplab	0.25	12.91	0.05	2.25

Method	Dataset `poster`			
	`6dof`		`translation`	
	e_t	e_r	e_t	e_r
IDOL	0.62	6.15	0.70	10.82
ROVIO	0.15	8.21	0.13	2.09
ROVIO + maplab	0.12	3.53	0.09	1.92

Method	Dataset `dynamic`			
	`6dof`		`translation`	
	e_t	e_r	e_t	e_r
IDOL	0.54	6.08	0.25	4.92
ROVIO	0.16	10.27	0.12	6.19
ROVIO + maplab	0.08	6.09	0.09	3.15

TABLE 5.1: Overall RMSE of translation estimation e_t [m] and orientation estimation e_r [deg] for IDOL, ROVIO and ROVIO+maplab on different datasets ($0 - 40$ s, aligning first 50 poses).

of more robust strategies from the event-based and frame-based VIO literature with, for example, the adoption of a robust loss function.

5.6 Conclusions

This Chapter introduced IDOL, a novel pipeline for event-based VIO using lines as features. Unlike most of the event-based methods in the literature, the proposed method does not aggregate events into frames that are later used in a traditional frame-based VO pipeline. Here, the events are considered individually as part of a batch optimisation that

estimates the position of 3D lines alongside the system's pose and velocity. This framework leverages the GPMs to characterise the system's trajectory based on a continuous representation of the inertial data.
We demonstrated the feasibility of event-based VIO with lines in different real-world scenarios. While the presented system does not have real-time capabilities at the time of writing, IDOL results show the potential to become an efficient, robust, and accurate event-based VIO method. Across quantitative benchmarking against state-of-the-art frame-based VIO algorithms, IDOL demonstrated accuracy in the same order of magnitude for translation, and competitive results for orientation.

Future work includes the exploration of different optimisation strategies and the implementation of a probabilistic marginalisation of the past state variables to substitute for the current growing full-batch optimisation. The computational cost of IDOL can be addressed at different levels. For example, one could optimise the implementation by using GPU computation as most of the operations are highly parallelizable (normal estimations for each of the events in their spatio-temporal representation, computation of the residuals and Jacobians, inference of the per-event GPMs, etc.). At a higher level, reducing the size of the optimisation problem would greatly benefit the method's efficiency. We will investigate different strategies to reduce the number of residuals while not sacrificing the amount of information used in the state estimation. Finally, combining our line-based features with corner-based features is likely to improve both robustness and accuracy.

Chapter 6

Conclusions and future work

6.1 Conclusions

In this thesis, we presented a novel preintegration method that generates probabilistic and continuous pseudo-measurements denoted GPMs. This technique furthers the classic preintegration method introduced in [1]. It leverages GPs regression and the application of linear operators to the kernel covariance functions to preform analytical integration of an IMU's inertial data. Additionally, the seamless integration of the GPMs into any inertial-aided navigation system is made possible by the derivation of postintegration mechanisms for IMU biases and time-shift corrections. This last feature, associated with the continuous nature of the method, addresses the issue of general inter-sensor asynchronism in multi-modal platforms.

In Chapter 4, we presented a first application of the GPMs as part of a LiDAR-inertial framework for localisation, mapping, and extrinsic calibration. The resulting framework is called IN2LAAMA and addresses thoroughly the issue of motion-distortion in LiDAR scans. The probabilistic formulation relies on the use of GPMs to characterise the system's trajectory between two estimated poses, hence providing inertial information for each of the LiDAR points. In the front-end, feature points that belong to planar patches or edges are extracted and matched across consecutive frames. The back-end's cost function aims at minimising point-to-plane and point-to-line distances (that correspond to the

features registration), and inertial constraints between frames. The projection of the feature points from a frame to another leverages the GPMs. In this way, the motion distortion is considered all along the estimation process via a full batch optimisation. As the extraction of good features needs a good correction of the motion distortion, and good motion distortion correction needs good features, the proposed method implements a tight integration between front-end and back-end. We compared IN2LAAMA with the current state-of-the-art LiDAR odometry framework that relies on a constant velocity motion model. In fast motion scenarios, this assumption reaches its limits, while our motion-model-less method can produce dense and accurate maps.

A second GPM application is introduced in Chapter 5. The proposed method is called IDOL and performs event-based visual-inertial odometry using 3D lines to represent the environment. Event-cameras brought new challenges to the robotics community as they output a new type of data characterised by their high bandwidth and asynchronism. While most of the methods in the literature rely on the accumulation of events into temporal windows before making use of traditional image-based techniques, the proposed method accounts individually for the events in the optimisation cost function. This novel paradigm is enabled by the use of GPMs to asynchronously characterise the system's pose at each of the events' time-stamps. The front-end extracts clusters of events that correspond physical lines leveraging the fact that lines appear as locally-planar patches in the spatio-temporal representation of the event stream. The extracted clusters do not convey any explicit information about where the line extremities are in the camera frame. Nonetheless, atop point-to-line and inertial constraints, new attraction/repulsion factors are introduced to estimate the 3D lines' end-points in the environment alongside the system's pose. In its present state, our method does not benefit from any robustness mechanisms that are present in every mature visual-inertial odometry pipeline. Yet, it still performs in the same order of magnitude as the current frame-based state-of-the-art methods. We believe that our approach can be the baseline for further developments toward assumption-less event-based visual odometry frameworks.

Throughout this thesis, Gaussian preintegration performances are demonstrated across a wide range of experiments whether it is on its own in Chapter 3, or as part of inertial-aided systems in Chapters 4 and 5. While the different methods presented in this thesis

can be computationally intensive, they are rigorous formulations of inertial-aided state estimation without motion models. Certain applications (e.g., mapping as a service) can leverage them as they are. However, the current implementations can greatly benefit from common engineering techniques such as parallelisation on GPUs. And we believe that our work can serve as a baseline for developments of real-time localisation and mapping methods. Details about the potential improvements are given in the respective conclusion sections of each chapter.

6.2 Future work and associated developments

6.2.1 Semantic understanding of the scene

The localisation frameworks presented in this thesis only consider static environments and are based on purely geometric approaches. Over the last few years, the fields of computer vision and machine learning have seen unprecedented advancements. It is now possible for a system to access high-level information about a scene. For example, semantic segmentation algorithms can give class labels to every pixel in an image to determine which type of object they belong (e.g. road, car, pedestrian, tree, etc.). The authors in [60], for instance, use semantic information over clusters of LiDAR points to perform mapping with high-level features.

In the context of IN2LAAMA, semantic information can seamlessly be used to segregate LiDAR points that belong to dynamic objects in order to allow operations in non-static environments. As illustrated in Fig. 6.1, the object detection method we presented in [132] upsamples sparse LiDAR data into Red-Green-Blue (RGB)-like images before performing transfer learning leveraging a pre-trained deep neural network [133]. Other works like [134] directly use 3D convolutions on geometric data to infer 3D bounding boxes around the objects of interest.

Looking further than the task of mapping, combining semantic and geometric information is a key element to enable high-level path planning of autonomous systems. To “fetch a glass from the kitchen’s table” a robot needs to know where the table is in space as

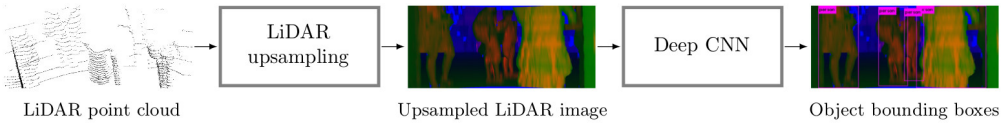
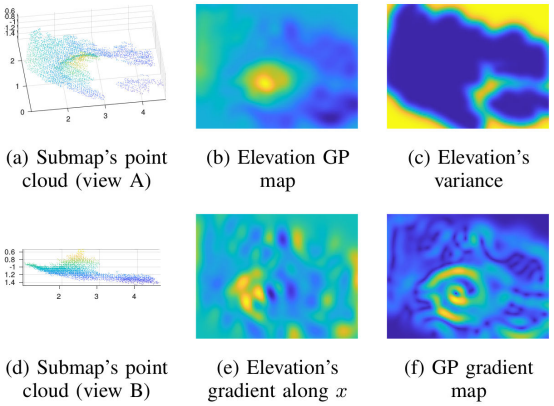


FIGURE 6.1: Overview of the laser-based object detection pipeline presented in [132].

well as any other obstacle on the way. Embedding semantic information in the construction of IN2LAAMA's maps could allow autonomous systems to update their semantic understanding of the environment automatically without further guidance from the user.

6.2.2 Loop closure detection

The loop-closure detection mechanism presented in Chapter 4 is probably the part of IN2LAAMA that would benefit the most from further developments. Loop closure detections help to correct the drift accumulated along any open-loop pose estimation. While semantic information can provide crucial information for high-level loop-closure detection, most of the frameworks rely on geometry or appearance-based methods. For visual data, a conventional approach is to use a bag-of-words over extracted features. This technique is presented in [135] and improved in [136]. Most LiDARs provide sparser data than visual images. Therefore, the extraction of spatial features, and associating them together, is more challenging. In [137], the authors summarise the information contained in 3D-LiDAR scans with surface-descriptors (Normal Distribution Transform (NDT)) histograms. The loop closures are detected if a similarity metric between two scans is above a certain threshold. A similar approach is presented in [138]. The authors of [139] claim to significantly outperform [137] by training an AdaBoost classifier [140] over rotation-invariant engineered features. In [141], denser laser data are used. Representing the 3D point clouds as range images, the method computes features and uses pre-trained bags of words to calculate the possible transformations (and their associated confidence scores) between scan pairs. In [58], a learnt random-forest classifier is used to determine if segments extracted in the scene have already been observed. While also particularly adapted to urban ground navigation, the method [142] introduces a different approach built on a novel global descriptor and a cosine similarity metric to find loop-closures in point cloud maps.



(a) Submap's point cloud (view A) (b) Elevation GP map (c) Elevation's variance

(d) Submap's point cloud (view B) (e) Elevation's gradient along x (f) GP gradient map

FIGURE 6.2: Example of Gaussian gradient map for loop-closure detection in unstructured planetary-exploration scenarios

Most of the aforementioned methods can seamlessly be integrated into IN2LAAMA, but loop-closure detection in 3D point clouds is still an open research topic as each of these methods have different strengths and weaknesses. As part of a collaboration with the German Aerospace Centre (DLR), we addressed the issue of loop-closure detection and robot re-localisation based on 3D data in unstructured planetary-exploration scenarios [143]. The high similarity of the terrain makes place recognition extremely challenging in such environments. Using GP regression and linear operator on covariance kernels, our method transforms 3D-point cloud maps into Gaussian gradient maps that represent the terrain change of elevation (Fig. 6.2). The method assumes overhanging structures are non-existent in the robot vicinity. The novel maps can be sampled into image-like data and process with traditional computer vision tools for image-based place recognition and registration. Through different real-world experiments using a stereo-visual-inertial system on Mount Etna and in the Moroccan desert, we demonstrate that the proposed method outperforms the current vision-based state-of-the-art.

6.2.3 Calibration trajectories

In the LiDAR-IMU framework presented in Chapter 4, we also perform extrinsic calibration of the sensors. Intuitively, given the nature of the IMU readings, a good calibration necessitates the systems to "move in every direction". In [117], a formal observability

analysis is conducted to determine the effectiveness of different trajectory types for inertial-aided system extrinsic calibration. Previous works [144, 145] have explored the issue of informative path planning for self-calibration. Nonetheless, no method has yet been proposed to address the issue of online path-planning for simultaneous self-calibration and mapping.

In [146], we explore the use of a Rapidly Exploring Random Belief Trees (RRBT) planner in association with IN2LAAMA to determine the optimal calibration trajectory (Fig. 6.3). One of the challenges originates from the fact that IN2LAAMA's computations are currently executed offline through a full batch optimisation. The proposed pipeline uses the iterative uncertainty propagation step of a standard EKF (based on the Jacobians associated to the marginal additions of simulated poses into IN2LAAMA's factor graph) to evaluate and compare the amount of "calibration-information" present in each RRBT candidate. Our preliminary results demonstrate that the EKF-style covariance propagation provides a good proxy for the true calibration covariance estimated with the full batch optimisation.

6.2.4 Event-based visual-lidar-inertial localisation and mapping

Chapters 4 and 5 of this thesis presented localisation and mapping frameworks based on LiDAR-IMU and event-camera-IMU systems, respectively. While both methods are promising on their own, they have some drawbacks that are inherent to the different modalities. For example, the position of a LiDAR in a tunnel-like environment is not observable due to the lack of geometric constraints along the tunnel axis. Equally, a camera (frame-based or event-based) becomes ineffective in texture-less environments. In such scenarios, the system's pose estimate is subject to the unconstrained and ineluctable drift of the IMU data integration.

Traditional vision and LiDAR data are complementary, as demonstrated in [52, 54–56]. However, despite publicly available datasets and simulators [147, 148], no event-based visual-lidar-inertial localisation and mapping framework has yet been proposed in the literature to our knowledge at the time of writing. We believe that IDOL and IN2LAAMA constitute straightforward building blocks to create such a framework. The use of GPMs as

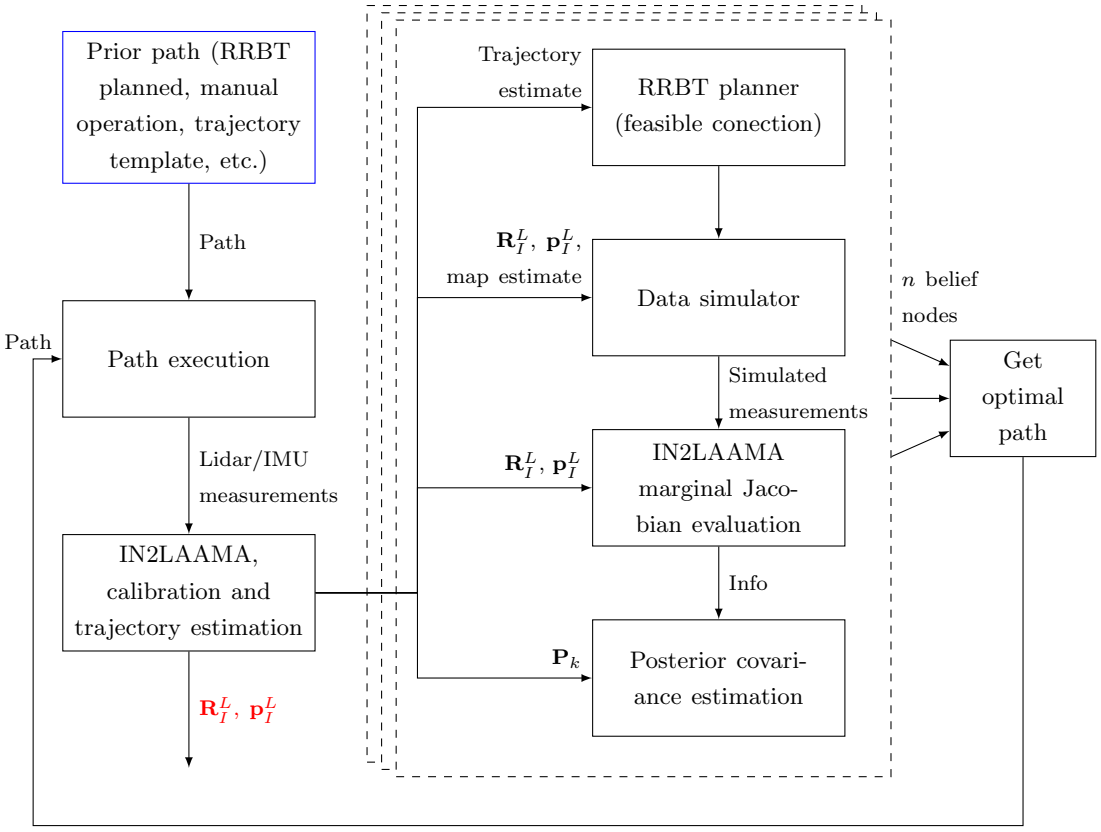


FIGURE 6.3: Overview of the proposed informative path planing method for optimal LiDAR-IMU extrinsic calibration.

the system's backbone for pose estimation enables multi-sensor asynchronism and accurate continuous motion characterisation.

Appendix A

Overview of the Upsampled Preintegration method

This appendix provide an overview of the algorithm used in [19] and [15] to generate the UPMs from 6-DoF IMU data. Conceptually, this method performs the standard preintegration [17] over "virtual" IMU measurements.

As shown in Fig. A.1, the method first upsamples the IMU signals using GP regression (equations present in Subsection 3.3.1). More than sampling the signal at higher frequency, the use of a non-parametric interpolation method allows us to obtain measurements at the time-stamps of interest as displayed in Fig. A.2. The upsampled IMU measurements also benefit from the probabilistic nature of the GPs. From the high-frequency signals, the UPMs $\Delta\mathbf{R}_{t_1}^{t_2}$, $\Delta\mathbf{v}_{t_1}^{t_2}$, and $\Delta\mathbf{p}_{t_1}^{t_2}$ from Subsection 3.2.2 are computed as per Algorithm 3.

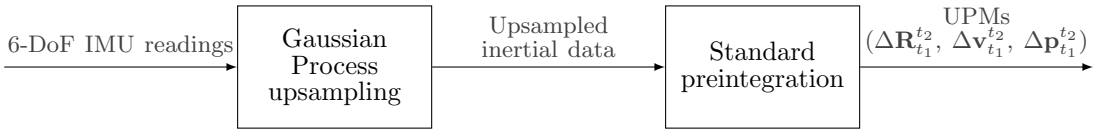


FIGURE A.1: High-level block diagram of the UPM method

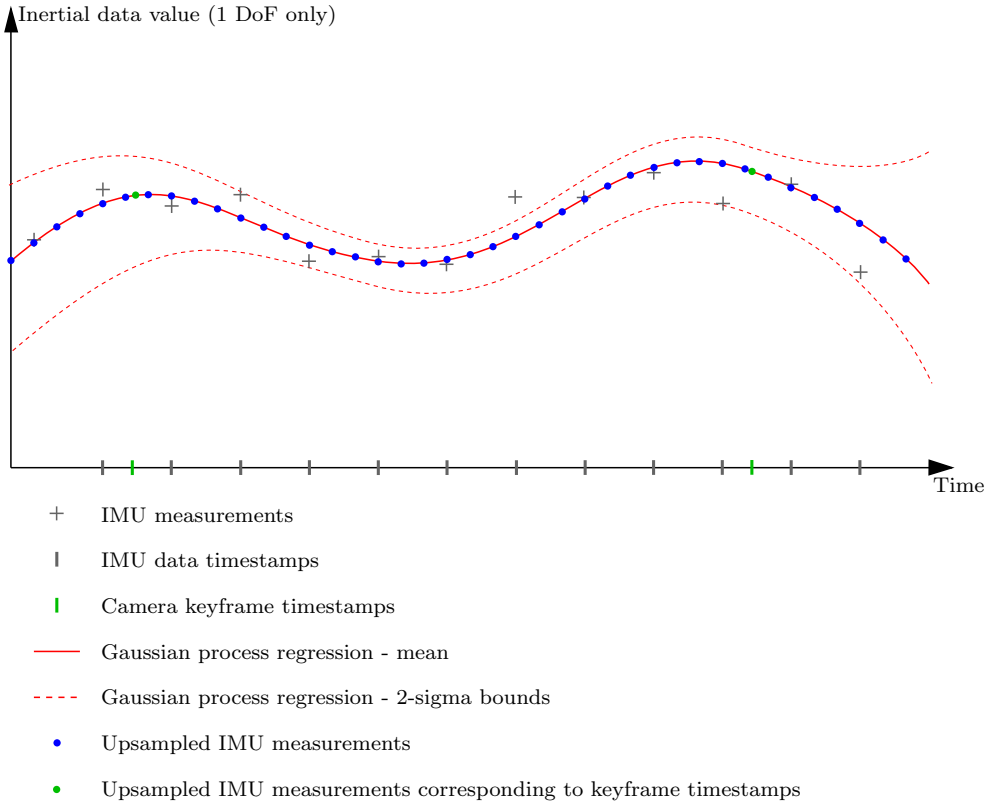


FIGURE A.2: Illustration of the GP upsampling for one of the six IMU DoF in the context of visual-inertial fusion.

Algorithm 3 Algorithm to perform standard preintegration of upsampled IMU signals.

Input:

$\tilde{\boldsymbol{\omega}}^*(t_{v_i})$: Upsampled measurements from a 3D gyroscope
$\tilde{\mathbf{f}}^*(t_{v_i})$: Upsampled measurements from a 3D accelerometer
$\bar{\mathbf{b}}_\omega(t_{v_i})$: Prior knowledge of the gyroscope biases
$\bar{\mathbf{b}}_f(t_{v_i})$: Prior knowledge of the accelerometer biases

Output:

$\Delta\mathbf{R}_{t_1}^{t_2}$, $\Delta\mathbf{v}_{t_1}^{t_2}$, $\Delta\mathbf{p}_{t_1}^{t_2}$: Upsampled Preintegrated Measurements (UPMs)

$\Delta\mathbf{R} \leftarrow \mathbf{0}_{3\times3}$
$\Delta\mathbf{v} \leftarrow \mathbf{0}_{3\times1}$
$\Delta\mathbf{p} \leftarrow \mathbf{0}_{3\times1}$
for i such that $t_{v_i} \in [t_1, t_2[$ **do**
    $\Delta\mathbf{p} \leftarrow \Delta\mathbf{p} + \Delta\mathbf{v}(t_{v_{i+1}} - t_{v_i}) + \Delta\mathbf{R}\big(\tilde{\mathbf{f}}^*(t_{v_i}) - \bar{\mathbf{b}}_f(t_{v_i})\big)\frac{(t_{v_{i+1}} - t_{v_i})^2}{2}$
    $\Delta\mathbf{v} \leftarrow \Delta\mathbf{v} + \Delta\mathbf{R}\big(\tilde{\mathbf{f}}^*(t_{v_i}) - \bar{\mathbf{b}}_f(t_{v_i})\big)(t_{v_{i+1}} - t_{v_i})$
    $\Delta\mathbf{R} \leftarrow \Delta\mathbf{R}\exp\Big(\Big(\big(\tilde{\boldsymbol{\omega}}^*(t_{v_i}) - \bar{\mathbf{b}}_\omega(t_{v_i})\big)(t_{v_{i+1}} - t_{v_i})\Big)^\wedge\Big)$
end for
$\Delta\mathbf{R}_{t_1}^{t_2} \leftarrow \Delta\mathbf{R}$
$\Delta\mathbf{v}_{t_1}^{t_2} \leftarrow \Delta\mathbf{v}$
$\Delta\mathbf{p}_{t_1}^{t_2} \leftarrow \Delta\mathbf{p}$
return

Appendix B

Derivation of the bias jacobians for GPM postintegration correction

This appendix exposes the derivation to obtain $\frac{\partial \mathbf{f}_j}{\partial \mathbf{b}_f}$ and $\frac{\partial \mathbf{f}_j}{\partial \mathbf{b}_\omega}$ used in (3.61). To do so, additional notion is employed:

- $[\mathbf{M}]_{(:)^T}$ the operator that transforms a r-by-c matrix $\mathbf{M}$ into a rc-by-1 vector.
- $[\mathbf{M}]_{(j,:)}$ the operator that isolates the j^{th} row of $\mathbf{M}$.

Examples:

$$\mathbf{M} = \begin{bmatrix} m_1 & m_4 & m_7 \\ m_2 & m_5 & m_8 \\ m_3 & m_6 & m_9 \end{bmatrix} \quad \text{then} \quad [\mathbf{M}]_{(:)^T} = \begin{bmatrix} m_1 & m_2 & m_3 & m_4 & m_5 & m_6 & m_7 & m_8 & m_9 \end{bmatrix}, \tag{B.1}$$

and

$$\mathbf{M} = \begin{bmatrix} m_1 & m_4 & m_7 \\ m_2 & m_5 & m_8 \\ m_3 & m_6 & m_9 \end{bmatrix} \quad \text{then} \quad [\mathbf{M}]_{(1,:)} = \begin{bmatrix} m_1 & m_4 & m_7 \end{bmatrix} \tag{B.2}$$

B.1 Accelerometer bias

As presented in (3.1), the accelerometer readings are modelled with an additive bias $\mathbf{b}_f$. Consequently, $\frac{\partial \tilde{\mathbf{f}}(t)}{\partial \mathbf{b}_f(t)} = \mathbf{I}$. The stack of rotated acceleration training values $\mathbf{f}_j$ corresponds to

$$\mathbf{f}_j = \begin{bmatrix} \left(\Delta\mathbf{R}_{t_1}^{\mathrm{t}_1}(\mathrm{t}_1)(\tilde{\mathbf{f}}(\mathrm{t}_1) - \bar{\mathbf{b}}_f(\mathrm{t}_1))\right)_j \\ \cdots \\ \left(\Delta\mathbf{R}_{t_1}^{\mathrm{t}_Q}(\mathrm{t}_Q)(\tilde{\mathbf{f}}(\mathrm{t}_Q) - \bar{\mathbf{b}}_f(\mathrm{t}_Q))\right)_j \end{bmatrix}. \tag{B.3}$$

Therefore

$$\frac{\partial \mathbf{f}_j}{\partial \mathbf{b}_f} = \begin{bmatrix} [\Delta\mathbf{R}_{t_1}^{\mathrm{t}_1}(\mathrm{t}_1)]_{(j,:)} \\ \cdots \\ [\Delta\mathbf{R}_{t_1}^{\mathrm{t}_Q}(\mathrm{t}_Q)]_{(j,:)}. \end{bmatrix} \tag{B.4}$$

B.2 Gyroscope bias

The Jacobian of the rotated acceleration training values in (3.61) is defined as

$$\frac{\partial \mathbf{f}_j}{\partial \mathbf{b}_\omega} = \begin{bmatrix} \frac{\partial\left(\Delta\mathbf{R}_{t_1}^{\mathrm{t}_1}(\mathbf{b}_\omega,\delta_t)(\tilde{\mathbf{f}}(\mathrm{t}_1)-\bar{\mathbf{b}}_f(\mathrm{t}_1))\right)_j}{\partial \mathbf{b}_\omega} \\ \cdots \\ \frac{\partial\left(\Delta\mathbf{R}_{t_1}^{\mathrm{t}_Q}(\mathbf{b}_\omega,\delta_t)(\tilde{\mathbf{f}}(\mathrm{t}_Q)-\bar{\mathbf{b}}_f(\mathrm{t}_Q))\right)_j}{\partial \mathbf{b}_\omega} \end{bmatrix} \tag{B.5}$$

$$= \begin{bmatrix} \frac{\partial\left(\Delta\mathbf{R}_{t_1}^{\mathrm{t}_1}(\bar{\mathbf{b}}_\omega,\bar{\delta}_t)\exp\left(\left(\frac{\partial\Delta\mathbf{R}_{t_1}^{\mathrm{t}_1}(t)}{\partial \mathbf{b}_\omega}\hat{\mathbf{b}}_\omega\right)^\wedge\right)(\tilde{\mathbf{f}}(\mathrm{t}_1)-\bar{\mathbf{b}}_f(\mathrm{t}_1))\right)_j}{\partial \mathbf{b}_\omega} \\ \cdots \\ \frac{\partial\left(\Delta\mathbf{R}_{t_1}^{\mathrm{t}_Q}(\bar{\mathbf{b}}_\omega,\bar{\delta}_t)\exp\left(\left(\frac{\partial\Delta\mathbf{R}_{t_1}^{\mathrm{t}_Q}(t)}{\partial \mathbf{b}_\omega}\hat{\mathbf{b}}_\omega\right)^\wedge\right)(\tilde{\mathbf{f}}(\mathrm{t}_Q)-\bar{\mathbf{b}}_f(\mathrm{t}_Q))\right)_j}{\partial \mathbf{b}_\omega} \end{bmatrix}. \tag{B.6}$$

The first-order Taylor expansion (3.54) is used to go from (B.5) to (B.6).

Let us define $\mathbf{q}_1 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^\top$, $\mathbf{q}_2 = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^\top$, and $\mathbf{q}_3 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^\top$:

$$\frac{\partial \left(\Delta\mathbf{R}_{t_1}^{\mathbf{t}_i}(\bar{\mathbf{b}}_\omega, \bar{\delta}_t) \exp\left(\left(\frac{\partial \Delta\mathbf{R}_{t_1}^{\mathbf{t}_i}(t)}{\partial \mathbf{b}_\omega} \hat{\mathbf{b}}_\omega \right)^\wedge \right) \tilde{\mathbf{f}}(\mathbf{t}_i) - \bar{\mathbf{b}}_f(\mathbf{t}_i)) \right)_j}{\partial \mathbf{b}_\omega} \tag{B.7}$$

$$= \frac{\partial \mathbf{q}_j^\top \Delta\mathbf{R}_{t_1}^{\mathbf{t}_i}(\bar{\mathbf{b}}_\omega, \bar{\delta}_t) \exp\left(\left(\frac{\partial \Delta\mathbf{R}_{t_1}^{\mathbf{t}_i}(t)}{\partial \mathbf{b}_\omega} \hat{\mathbf{b}}_\omega \right)^\wedge \right) (\tilde{\mathbf{f}}(\mathbf{t}_i) - \bar{\mathbf{b}}_f(\mathbf{t}_i))}{\partial \mathbf{b}_\omega} \tag{B.8}$$

$$= \left[\left(\Delta\mathbf{R}_{t_1}^{\mathbf{t}_i}(\bar{\mathbf{b}}_\omega, \bar{\delta}_t) \right)^\top \mathbf{q}_j \left(\tilde{\mathbf{f}}(\mathbf{t}_i) - \bar{\mathbf{b}}_f(\mathbf{t}_i) \right)^\top \right]_{(:)^T} \frac{\partial [\exp\left((\mathbf{a})^\wedge \right)]_{(:)^T}}{\partial \mathbf{a}} \bigg|_{\mathbf{a}=\mathbf{0}} \frac{\partial \Delta\mathbf{R}_{t_1}^{\mathbf{t}_i}(t)}{\partial \mathbf{b}_\omega} \tag{B.9}$$

To go from (B.8) to (B.9), we use the chain rule, the fact that $\frac{\partial \mathbf{a}^\top \mathbf{X} \mathbf{b}}{\partial \mathbf{X}} = \mathbf{a}\mathbf{b}^\top$, $\hat{\mathbf{b}}_\omega$ is close to zero, and $\frac{\partial \mathbf{X}\mathbf{a}}{\partial \mathbf{a}} = \mathbf{X}$. The Jacobian of the exponential mapping around zero is defined as

$$\frac{\partial [\exp\left((\mathbf{a})^\wedge \right)]_{(:)^T}}{\partial \mathbf{a}} \bigg|_{\mathbf{a}=\mathbf{0}} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{B.10}$$

Appendix C

IN2LAAMA Jacobians

This appendix provides the reader with the different Jacobians involved in IN2LAAMA's optimisation. Our implementation uses Ceres and its *local parameterisation* feature. Consequently, the Jacobians in this section are derived with respect to the over-parameterised SO(3) variables. Note that many computations could be further optimised as many matrices are quite sparse.

The following notation is employed:

- $[\mathbf{M}]_{(:)^T}$ the operator that transforms a r-by-c matrix $\mathbf{M}$ into a rc-by-1 vector.
- $[\mathbf{M}]_{(j,:)}$ the operator that isolates the j^{th} row of $\mathbf{M}$.
- $[\mathbf{M}]_{(:,j)}$ the operator that isolates the j^{th} column of $\mathbf{M}$.

Examples:

$$\mathbf{M} = \begin{bmatrix} m_1 & m_4 & m_7 \\ m_2 & m_5 & m_8 \\ m_3 & m_6 & m_9 \end{bmatrix} \quad \text{then} \quad [\mathbf{M}]_{(:)^T} = \begin{bmatrix} m_1 & m_2 & m_3 & m_4 & m_5 & m_6 & m_7 & m_8 & m_9 \end{bmatrix}, \tag{C.1}$$

$$\mathbf{M} = \begin{bmatrix} m_1 & m_4 & m_7 \\ m_2 & m_5 & m_8 \\ m_3 & m_6 & m_9 \end{bmatrix} \quad \text{then} \quad [\mathbf{M}]_{(1,:)} = \begin{bmatrix} m_1 & m_4 & m_7 \end{bmatrix}, \tag{C.2}$$

and

$$\mathbf{M} = \begin{bmatrix} m_1 & m_4 & m_7 \\ m_2 & m_5 & m_8 \\ m_3 & m_6 & m_9 \end{bmatrix} \quad \text{then} \quad [\mathbf{M}]_{(:,1)} = \begin{bmatrix} m_1 \\ m_2 \\ m_3 \end{bmatrix} \tag{C.3}$$

Additionally, we define $\Delta t_m^i = t_i - \tau_m$, $= \tau_m - \tau_{m\text{-}1}$, $\mathrm{Log}(\bullet) = \log(\bullet^\vee)$, $\mathrm{Exp}(\bullet) = \exp(\bullet^\wedge)$, $\mathbf{q}_1 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^\top$, $\mathbf{q}_2 = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^\top$, and $\mathbf{q}_3 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^\top$.

C.1 IMU factors

$$\frac{\partial \mathbf{r}_{I_v}^m}{\partial \mathbf{v}_W^{\tau_m}} = \frac{\partial \mathbf{r}_{I_p}^m}{\partial \mathbf{p}_W^{\tau_m}} = \mathbf{R}_W^{\tau_{m\text{-}1}\top} \tag{C.4}$$

$$\frac{\partial \mathbf{r}_{I_v}^m}{\partial \mathbf{v}_W^{\tau_{m\text{-}1}}} = \frac{\partial \mathbf{r}_{I_p}^m}{\partial \mathbf{p}_W^{\tau_{m\text{-}1}}} = -\mathbf{R}_W^{\tau_{m\text{-}1}\top} \tag{C.5}$$

$$\frac{\partial \mathbf{r}_{I_p}^m}{\partial \mathbf{v}_W^{\tau_{m\text{-}1}}} = -\Delta\tau_{m\text{-}1}^m \mathbf{R}_W^{\tau_{m\text{-}1}\top} \tag{C.6}$$

$$\frac{\partial \mathbf{r}_{I_p}^m}{\partial [\mathbf{R}_W^{\tau_{m\text{-}1}}]_{(:)^T}} = \begin{bmatrix} \mathbf{a}^\top & \mathbf{0}_{1\times 3} & \mathbf{0}_{1\times 3} \\ \mathbf{0}_{1\times 3} & \mathbf{a}^\top & \mathbf{0}_{1\times 3} \\ \mathbf{0}_{1\times 3} & \mathbf{0}_{1\times 3} & \mathbf{a}^\top \end{bmatrix} \tag{C.7}$$

with $\mathbf{a} = \mathbf{p}_W^{\tau_m} - \mathbf{p}_W^{\tau_{m\text{-}1}} - \Delta\tau_{m\text{-}1}^m \mathbf{v}_W^{\tau_{m\text{-}1}} - \frac{1}{2}\Delta\tau_{m\text{-}1}^m{}^2\mathbf{g}$

$$\frac{\partial \mathbf{r}_{I_v}^m}{\partial [\mathbf{R}_W^{\tau_{m\text{-}1}}]_{(:)^T}} = \begin{bmatrix} \mathbf{a}^\top & \mathbf{0}_{1\times 3} & \mathbf{0}_{1\times 3} \\ \mathbf{0}_{1\times 3} & \mathbf{a}^\top & \mathbf{0}_{1\times 3} \\ \mathbf{0}_{1\times 3} & \mathbf{0}_{1\times 3} & \mathbf{a}^\top \end{bmatrix} \tag{C.8}$$

with $\mathbf{a} = \mathbf{v}_W^{\tau_m} - \mathbf{v}_W^{\tau_{m\text{-}1}} - \Delta\tau_{m\text{-}1}^m \mathbf{g}$

$$\frac{\partial \mathbf{r}_{I_r}^m}{\partial [\mathbf{R}_W^{\tau_m}]_{(:)^T}} = \frac{\partial \mathrm{Log}(\Delta\mathbf{R}_{\tau_{m\text{-}1}}^{\tau_m}{}^\top \mathbf{R}_W^{\tau_{m\text{-}1}}{}^\top \mathbf{R}_W^{\tau_m})}{\partial [\Delta\mathbf{R}_{\tau_{m\text{-}1}}^{\tau_m}{}^\top \mathbf{R}_W^{\tau_{m\text{-}1}}{}^\top \mathbf{R}_W^{\tau_m}]_{(:)^T}} \frac{\partial [\Delta\mathbf{R}_{\tau_{m\text{-}1}}^{\tau_m}{}^\top \mathbf{R}_W^{\tau_{m\text{-}1}}{}^\top \mathbf{R}_W^{\tau_m}]_{(:)^T}}{\partial [\mathbf{R}_W^{\tau_m}]_{(:)^T}} \tag{C.9}$$

with

$$\frac{\partial [\Delta\mathbf{R}_{\tau_{m\text{-}1}}^{\tau_m}{}^\top \mathbf{R}_W^{\tau_{m\text{-}1}}{}^\top \mathbf{R}_W^{\tau_m}]_{(:)^T}}{\partial [\mathbf{R}_W^{\tau_m}]_{(:)^T}} = \begin{bmatrix} (\mathbf{R}_W^{\tau_{m\text{-}1}} \Delta\mathbf{R}_{\tau_{m\text{-}1}}^{\tau_m})^\top & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 3} & (\mathbf{R}_W^{\tau_{m\text{-}1}} \Delta\mathbf{R}_{\tau_{m\text{-}1}}^{\tau_m})^\top & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} & (\mathbf{R}_W^{\tau_{m\text{-}1}} \Delta\mathbf{R}_{\tau_{m\text{-}1}}^{\tau_m})^\top \end{bmatrix} \tag{C.10}$$

$$\frac{\partial \mathbf{r}_{I_r}^m}{\partial [\mathbf{R}_W^{\tau_{m\text{-}1}}]_{(:)^T}} = \frac{\partial \mathrm{Log}(\Delta\mathbf{R}_{\tau_{m\text{-}1}}^{\tau_m}{}^\top \mathbf{R}_W^{\tau_{m\text{-}1}}{}^\top \mathbf{R}_W^{\tau_m})}{\partial [\Delta\mathbf{R}_{\tau_{m\text{-}1}}^{\tau_m}{}^\top \mathbf{R}_W^{\tau_{m\text{-}1}}{}^\top \mathbf{R}_W^{\tau_m}]_{(:)^T}} \frac{\partial [\Delta\mathbf{R}_{\tau_{m\text{-}1}}^{\tau_m}{}^\top \mathbf{R}_W^{\tau_{m\text{-}1}}{}^\top \mathbf{R}_W^{\tau_m}]_{(:)^T}}{\partial [\mathbf{R}_W^{\tau_{m\text{-}1}}]_{(:)^T}} \tag{C.11}$$

with

$$\frac{\partial[\Delta\mathbf{R}_{\tau_{m\text{-}1}}^{\tau_m}{}^{\top}\mathbf{R}_W^{\tau_{m\text{-}1}}{}^{\top}\mathbf{R}_W^{\tau_m}]_{(:)^T}}{\partial[\mathbf{R}_W^{\tau_{m\text{-}1}}]_{(:)^T}} = \begin{bmatrix} \left[[\mathbf{R}_W^{\tau_m}]_{(:,1)}\left([\Delta\mathbf{R}_{\tau_{m\text{-}1}}^{\tau_m}]_{(:,1)}\right)^{\top}\right]_{(:)^T} \\ \left[[\mathbf{R}_W^{\tau_m}]_{(:,1)}\left([\Delta\mathbf{R}_{\tau_{m\text{-}1}}^{\tau_m}]_{(:,2)}\right)^{\top}\right]_{(:)^T} \\ \left[[\mathbf{R}_W^{\tau_m}]_{(:,1)}\left([\Delta\mathbf{R}_{\tau_{m\text{-}1}}^{\tau_m}]_{(:,3)}\right)^{\top}\right]_{(:)^T} \\ \left[[\mathbf{R}_W^{\tau_m}]_{(:,2)}\left([\Delta\mathbf{R}_{\tau_{m\text{-}1}}^{\tau_m}]_{(:,1)}\right)^{\top}\right]_{(:)^T} \\ \left[[\mathbf{R}_W^{\tau_m}]_{(:,2)}\left([\Delta\mathbf{R}_{\tau_{m\text{-}1}}^{\tau_m}]_{(:,2)}\right)^{\top}\right]_{(:)^T} \\ \left[[\mathbf{R}_W^{\tau_m}]_{(:,2)}\left([\Delta\mathbf{R}_{\tau_{m\text{-}1}}^{\tau_m}]_{(:,3)}\right)^{\top}\right]_{(:)^T} \\ \left[[\mathbf{R}_W^{\tau_m}]_{(:,3)}\left([\Delta\mathbf{R}_{\tau_{m\text{-}1}}^{\tau_m}]_{(:,1)}\right)^{\top}\right]_{(:)^T} \\ \left[[\mathbf{R}_W^{\tau_m}]_{(:,3)}\left([\Delta\mathbf{R}_{\tau_{m\text{-}1}}^{\tau_m}]_{(:,2)}\right)^{\top}\right]_{(:)^T} \\ \left[[\mathbf{R}_W^{\tau_m}]_{(:,3)}\left([\Delta\mathbf{R}_{\tau_{m\text{-}1}}^{\tau_m}]_{(:,3)}\right)^{\top}\right]_{(:)^T} \end{bmatrix}. \tag{C.12}$$

The analytic expressions of $\frac{\partial \text{Log}(\mathbf{X})}{\partial \mathbf{X}}$ and $\frac{\partial \text{Exp}(\mathbf{X})}{\partial \mathbf{X}}$ are elaborated with Matlab's symbolic toolbox.

$$\frac{\partial \mathbf{r}_{I_p}^m}{\partial \hat{\mathbf{b}}_f^{m\text{-}1}} = -\frac{\partial \Delta\mathbf{p}_{\tau_{m\text{-}1}}^{\tau_m}}{\partial \mathbf{b}_f^{m\text{-}1}} \tag{C.13}$$

$$\frac{\partial \mathbf{r}_{I_p}^m}{\partial \hat{\mathbf{b}}_\omega^{m\text{-}1}} = -\frac{\partial \Delta\mathbf{p}_{\tau_{m\text{-}1}}^{\tau_m}}{\partial \mathbf{b}_\omega^{m\text{-}1}} \tag{C.14}$$

$$\frac{\partial \mathbf{r}_{I_v}^m}{\partial \hat{\mathbf{b}}_f^{m\text{-}1}} = -\frac{\partial \Delta\mathbf{v}_{\tau_{m\text{-}1}}^{\tau_m}}{\partial \mathbf{b}_f^{m\text{-}1}} \tag{C.15}$$

$$\frac{\partial \mathbf{r}_{I_v}^m}{\partial \hat{\mathbf{b}}_\omega^{m\text{-}1}} = -\frac{\partial \Delta\mathbf{v}_{\tau_{m\text{-}1}}^{\tau_m}}{\partial \mathbf{b}_\omega^{m\text{-}1}} \tag{C.16}$$

$$\frac{\partial \mathbf{r}_{I_r}^m}{\partial \hat{\mathbf{b}}_\omega^{m\text{-}1}} = \frac{\partial \mathrm{Log}(\mathrm{Exp}\left(\frac{\partial \Delta\mathbf{R}_{\tau_{m\text{-}1}}^{\tau_m}}{\partial \mathbf{b}_\omega^{m\text{-}1}}\hat{\mathbf{b}}_\omega^{m\text{-}1} + \frac{\partial \Delta\mathbf{R}_{\tau_{m\text{-}1}}^{\tau_m}}{\partial \delta_{t_{m\text{-}1}}}\hat{\delta}_{t_{m\text{-}1}}\right)^\top \Delta\mathbf{R}_{\tau_{m\text{-}1}}^{\tau_m\top}(\bar{\mathbf{b}}_\omega^{m\text{-}1}, \bar{\delta}_{t_{m\text{-}1}})} \mathbf{R}_W^{\tau_{m\text{-}1}\top}\mathbf{R}_W^{\tau_m})}{\partial \mathrm{Exp}\left(\frac{\partial \Delta\mathbf{R}_{\tau_{m\text{-}1}}^{\tau_m}}{\partial \mathbf{b}_\omega^{m\text{-}1}}\hat{\mathbf{b}}_\omega^{m\text{-}1} + \frac{\partial \Delta\mathbf{R}_{\tau_{m\text{-}1}}^{\tau_m}}{\partial \delta_{t_{m\text{-}1}}}\hat{\delta}_{t_{m\text{-}1}}\right)^\top \Delta\mathbf{R}_{\tau_{m\text{-}1}}^{\tau_m\top}(\bar{\mathbf{b}}_\omega^{m\text{-}1}, \bar{\delta}_{t_{m\text{-}1}})} \mathbf{R}_W^{\tau_{m\text{-}1}\top}\mathbf{R}_W^{\tau_m}$$
$$\cdots \frac{\partial \mathrm{Exp}\left(\frac{\partial \Delta\mathbf{R}_{\tau_{m\text{-}1}}^{\tau_m}}{\partial \mathbf{b}_\omega^{m\text{-}1}}\hat{\mathbf{b}}_\omega^{m\text{-}1} + \frac{\partial \Delta\mathbf{R}_{\tau_{m\text{-}1}}^{\tau_m}}{\partial \delta_{t_{m\text{-}1}}}\hat{\delta}_{t_{m\text{-}1}}\right)^\top \Delta\mathbf{R}_{\tau_{m\text{-}1}}^{\tau_m\top}(\bar{\mathbf{b}}_\omega^{m\text{-}1}, \bar{\delta}_{t_{m\text{-}1}})} \mathbf{R}_W^{\tau_{m\text{-}1}\top}\mathbf{R}_W^{\tau_m}}{\partial \mathrm{Exp}\left(\frac{\partial \Delta\mathbf{R}_{\tau_{m\text{-}1}}^{\tau_m}}{\partial \mathbf{b}_\omega^{m\text{-}1}}\hat{\mathbf{b}}_\omega^{m\text{-}1} + \frac{\partial \Delta\mathbf{R}_{\tau_{m\text{-}1}}^{\tau_m}}{\partial \delta_{t_{m\text{-}1}}}\hat{\delta}_{t_{m\text{-}1}}\right)}$$
$$\cdots \frac{\partial \mathrm{Exp}\left(\frac{\partial \Delta\mathbf{R}_{\tau_{m\text{-}1}}^{\tau_m}}{\partial \mathbf{b}_\omega^{m\text{-}1}}\hat{\mathbf{b}}_\omega^{m\text{-}1} + \frac{\partial \Delta\mathbf{R}_{\tau_{m\text{-}1}}^{\tau_m}}{\partial \delta_{t_{m\text{-}1}}}\hat{\delta}_{t_{m\text{-}1}}\right)}{\partial \left(\frac{\partial \Delta\mathbf{R}_{\tau_{m\text{-}1}}^{\tau_m}}{\partial \mathbf{b}_\omega^{m\text{-}1}}\hat{\mathbf{b}}_\omega^{m\text{-}1} + \frac{\partial \Delta\mathbf{R}_{\tau_{m\text{-}1}}^{\tau_m}}{\partial \delta_{t_{m\text{-}1}}}\hat{\delta}_{t_{m\text{-}1}}\right)}$$
$$\cdots \frac{\partial \left(\frac{\partial \Delta\mathbf{R}_{\tau_{m\text{-}1}}^{\tau_m}}{\partial \mathbf{b}_\omega^{m\text{-}1}}\hat{\mathbf{b}}_\omega^{m\text{-}1} + \frac{\partial \Delta\mathbf{R}_{\tau_{m\text{-}1}}^{\tau_m}}{\partial \delta_{t_{m\text{-}1}}}\hat{\delta}_{t_{m\text{-}1}}\right)}{\partial \hat{\mathbf{b}}_\omega^{m\text{-}1}} \tag{C.17}$$

with

$$\frac{\partial \left(\frac{\partial \Delta\mathbf{R}_{\tau_{m\text{-}1}}^{\tau_m}}{\partial \mathbf{b}_\omega^{m\text{-}1}}\hat{\mathbf{b}}_\omega^{m\text{-}1} + \frac{\partial \Delta\mathbf{R}_{\tau_{m\text{-}1}}^{\tau_m}}{\partial \delta_{t_{m\text{-}1}}}\hat{\delta}_{t_{m\text{-}1}}\right)}{\partial \hat{\mathbf{b}}_\omega^{m\text{-}1}} = \frac{\partial \Delta\mathbf{R}_{\tau_{m\text{-}1}}^{\tau_m}}{\partial \mathbf{b}_\omega^{m\text{-}1}} \tag{C.18}$$

and

$$\frac{\partial \mathrm{Exp}\left(\frac{\partial \Delta\mathbf{R}_{\tau_{m\text{-}1}}^{\tau_m}}{\partial \mathbf{b}_\omega^{m\text{-}1}}\hat{\mathbf{b}}_\omega^{m\text{-}1} + \frac{\partial \Delta\mathbf{R}_{\tau_{m\text{-}1}}^{\tau_m}}{\partial \delta_{t_{m\text{-}1}}}\hat{\delta}_{t_{m\text{-}1}}\right)^\top \Delta\mathbf{R}_{\tau_{m\text{-}1}}^{\tau_m\top}(\bar{\mathbf{b}}_\omega^{m\text{-}1}, \bar{\delta}_{t_{m\text{-}1}})} \mathbf{R}_W^{\tau_{m\text{-}1}\top}\mathbf{R}_W^{\tau_m}}{\partial \mathrm{Exp}\left(\frac{\partial \Delta\mathbf{R}_{\tau_{m\text{-}1}}^{\tau_m}}{\partial \mathbf{b}_\omega^{m\text{-}1}}\hat{\mathbf{b}}_\omega^{m\text{-}1} + \frac{\partial \Delta\mathbf{R}_{\tau_{m\text{-}1}}^{\tau_m}}{\partial \delta_{t_{m\text{-}1}}}\hat{\delta}_{t_{m\text{-}1}}\right)} = \begin{bmatrix} [\mathbf{K}]_{(1,:)} & \mathbf{0}_{1\times 3} & \mathbf{0}_{1\times 3} \\ \mathbf{0}_{1\times 3} & [\mathbf{K}]_{(1,:)} & \mathbf{0}_{1\times 3} \\ \mathbf{0}_{1\times 3} & \mathbf{0}_{1\times 3} & [\mathbf{K}]_{(1,:)} \\ [\mathbf{K}]_{(2,:)} & \mathbf{0}_{1\times 3} & \mathbf{0}_{1\times 3} \\ \mathbf{0}_{1\times 3} & [\mathbf{K}]_{(2,:)} & \mathbf{0}_{1\times 3} \\ \mathbf{0}_{1\times 3} & \mathbf{0}_{1\times 3} & [\mathbf{K}]_{(2,:)} \\ [\mathbf{K}]_{(3,:)} & \mathbf{0}_{1\times 3} & \mathbf{0}_{1\times 3} \\ \mathbf{0}_{1\times 3} & [\mathbf{K}]_{(3,:)} & \mathbf{0}_{1\times 3} \\ \mathbf{0}_{1\times 3} & \mathbf{0}_{1\times 3} & [\mathbf{K}]_{(3,:)} \end{bmatrix}, \tag{C.19}$$

with

$$\mathbf{K} = \left(\Delta\mathbf{R}_{\tau_{m\text{-}1}}^{\tau_m\top}(\bar{\mathbf{b}}_\omega^{m\text{-}1}, \bar{\delta}_{t_{m\text{-}1}})\mathbf{R}_W^{\tau_{m\text{-}1}\top}\mathbf{R}_W^{\tau_m}\right)^\top \tag{C.20}$$

$$\frac{\partial \mathbf{r}_{I_r}^m}{\partial \hat{\mathbf{b}}_\omega^{m\text{-}1}} = \frac{\partial \mathrm{Log}(\mathrm{Exp}\left(\frac{\partial \Delta \mathbf{R}_{\tau_{m-1}}^{\tau_m}}{\partial \mathbf{b}_\omega^{m-1}} \hat{\mathbf{b}}_\omega^{m\text{-}1} + \frac{\partial \Delta \mathbf{R}_{\tau_{m-1}}^{\tau_m}}{\partial \delta_{t_{m-1}}} \hat{\delta}_{t_{m-1}}\right)^\top \Delta \mathbf{R}_{\tau_{m-1}}^{\tau_m\ \top}(\bar{\mathbf{b}}_\omega^{m-1}, \bar{\delta}_{t_{m-1}}) \mathbf{R}_W^{\tau_{m-1}\top} \mathbf{R}_W^{\tau_m})}{\partial \mathrm{Exp}\left(\frac{\partial \Delta \mathbf{R}_{\tau_{m-1}}^{\tau_m}}{\partial \mathbf{b}_\omega^{m-1}} \hat{\mathbf{b}}_\omega^{m\text{-}1} + \frac{\partial \Delta \mathbf{R}_{\tau_{m-1}}^{\tau_m}}{\partial \delta_{t_{m-1}}} \hat{\delta}_{t_{m-1}}\right)^\top \Delta \mathbf{R}_{\tau_{m-1}}^{\tau_m\ \top}(\bar{\mathbf{b}}_\omega^{m-1}, \bar{\delta}_{t_{m-1}}) \mathbf{R}_W^{\tau_{m-1}\top} \mathbf{R}_W^{\tau_m}} \cdots \frac{\partial \mathrm{Exp}\left(\frac{\partial \Delta \mathbf{R}_{\tau_{m-1}}^{\tau_m}}{\partial \mathbf{b}_\omega^{m-1}} \hat{\mathbf{b}}_\omega^{m\text{-}1} + \frac{\partial \Delta \mathbf{R}_{\tau_{m-1}}^{\tau_m}}{\partial \delta_{t_{m-1}}} \hat{\delta}_{t_{m-1}}\right)^\top \Delta \mathbf{R}_{\tau_{m-1}}^{\tau_m\ \top}(\bar{\mathbf{b}}_\omega^{m-1}, \bar{\delta}_{t_{m-1}}) \mathbf{R}_W^{\tau_{m-1}\top} \mathbf{R}_W^{\tau_m}}{\partial \mathrm{Exp}\left(\frac{\partial \Delta \mathbf{R}_{\tau_{m-1}}^{\tau_m}}{\partial \mathbf{b}_\omega^{m-1}} \hat{\mathbf{b}}_\omega^{m\text{-}1} + \frac{\partial \Delta \mathbf{R}_{\tau_{m-1}}^{\tau_m}}{\partial \delta_{t_{m-1}}} \hat{\delta}_{t_{m-1}}\right)} \cdots \frac{\partial \mathrm{Exp}\left(\frac{\partial \Delta \mathbf{R}_{\tau_{m-1}}^{\tau_m}}{\partial \mathbf{b}_\omega^{m-1}} \hat{\mathbf{b}}_\omega^{m\text{-}1} + \frac{\partial \Delta \mathbf{R}_{\tau_{m-1}}^{\tau_m}}{\partial \delta_{t_{m-1}}} \hat{\delta}_{t_{m-1}}\right)}{\partial \left(\frac{\partial \Delta \mathbf{R}_{\tau_{m-1}}^{\tau_m}}{\partial \mathbf{b}_\omega^{m-1}} \hat{\mathbf{b}}_\omega^{m\text{-}1} + \frac{\partial \Delta \mathbf{R}_{\tau_{m-1}}^{\tau_m}}{\partial \delta_{t_{m-1}}} \hat{\delta}_{t_{m-1}}\right)} \cdots \frac{\partial \left(\frac{\partial \Delta \mathbf{R}_{\tau_{m-1}}^{\tau_m}}{\partial \mathbf{b}_\omega^{m-1}} \hat{\mathbf{b}}_\omega^{m\text{-}1} + \frac{\partial \Delta \mathbf{R}_{\tau_{m-1}}^{\tau_m}}{\partial \delta_{t_{m-1}}} \hat{\delta}_{t_{m-1}}\right)}{\partial \hat{\delta}_{t_{m-1}}} \tag{C.21}$$

with

$$\frac{\partial \left(\frac{\partial \Delta \mathbf{R}_{\tau_{m-1}}^{\tau_m}}{\partial \mathbf{b}_\omega^{m-1}} \hat{\mathbf{b}}_\omega^{m\text{-}1} + \frac{\partial \Delta \mathbf{R}_{\tau_{m-1}}^{\tau_m}}{\partial \delta_{t_{m-1}}} \hat{\delta}_{t_{m-1}}\right)}{\partial \hat{\delta}_{t_{m-1}}} = \frac{\partial \Delta \mathbf{R}_{\tau_{m-1}}^{\tau_m}}{\partial \delta_{t_{m-1}}} \tag{C.22}$$

$$\frac{\partial \mathbf{r}_{I_p}^m}{\partial \hat{\delta}_{t_{m-1}}} = -\frac{\partial \Delta \mathbf{p}_{\tau_{m-1}}^{\tau_m}}{\partial \delta_{t_{m-1}}} \tag{C.23}$$

$$\frac{\partial \mathbf{r}_{I_v}^m}{\partial \hat{\delta}_{t_{m-1}}} = -\frac{\partial \Delta \mathbf{v}_{\tau_{m-1}}^{\tau_m}}{\partial \delta_{t_{m-1}}} \tag{C.24}$$

C.2 Biases factors

$$\frac{\partial \mathbf{r}_f^m}{\partial \hat{\mathbf{b}}_\omega^{m\text{-}1}} = \mathbf{I}_{3\times 3} \tag{C.25}$$

$$\frac{\partial \mathbf{r}_f^m}{\partial \hat{\mathbf{b}}_\omega^m} = -\mathbf{I}_{3\times 3} \tag{C.26}$$

$$\frac{\partial \mathbf{r}_\omega^m}{\partial \hat{\mathbf{b}}_f^{m\text{-}1}} = \mathbf{I}_{3\times 3} \tag{C.27}$$

$$\frac{\partial \mathbf{r}_\omega^m}{\partial \hat{\mathbf{b}}_f^m} = -\mathbf{I}_{3\times 3} \tag{C.28}$$

C.3 LiDAR factors

C.3.1 Point reprojection

Given $\mathbf{R}_I^L$ and $\mathbf{p}_I^L$ representing the transformation from the IMU frame to the lidar frame (calibration), the reprojection of a point $\mathbf{x}_L^i$ in the corresponding frame is

$$\mathbf{x}_W^i = \mathbf{R}_W^{\tau_m}\left(\Delta\mathbf{R}_{\tau_m}^{t_i}\left(\mathbf{R}_I^L\mathbf{x}_L^i + \mathbf{p}_I^L\right) + \Delta\mathbf{p}_{\tau_m}^{t_i}\right) + \mathbf{p}_W^{\tau_m} + \Delta t_m^i \mathbf{v}_W^{\tau_m} + \frac{1}{2}\Delta {t_m^i}^2 \mathbf{g}. \tag{C.29}$$

Taking into account the IMU bias and the time-shift between the two sensors, $\Delta\mathbf{R}_{\tau_m}^{t_i}$ becomes $\Delta\bar{\mathbf{R}}_{\tau_m}^{t_i}\mathrm{Exp}\left(\frac{\partial\Delta\mathbf{R}_{\tau_m}^{t_i}}{\partial\mathbf{b}_\omega^m}\hat{\mathbf{b}}_\omega^m + \frac{\partial\Delta\mathbf{R}_{\tau_m}^{t_i}}{\partial t}\hat{\delta}_{t_m}\right)$ and $\Delta\mathbf{p}_{\tau_m}^{t_i}$ becomes $\Delta\bar{\mathbf{p}}_{\tau_m}^{t_i} + \frac{\partial\Delta\mathbf{p}_{\tau_m}^{t_i}}{\partial\mathbf{b}_f^m}\hat{\mathbf{b}}_f^m + \frac{\partial\Delta\mathbf{p}_{\tau_m}^{t_i}}{\partial\mathbf{b}_\omega^m}\hat{\mathbf{b}}_\omega^m + \frac{\partial\Delta\mathbf{p}_{\tau_m}^{t_i}}{\partial t}\hat{\delta}_{t_m}$.

$$\begin{aligned}\mathbf{x}_W^i =& \mathbf{R}_W^{\tau_m}\left(\Delta\bar{\mathbf{R}}_{\tau_m}^{t_i}\mathrm{Exp}\left(\frac{\partial\Delta\mathbf{R}_{\tau_m}^{t_i}}{\partial\mathbf{b}_\omega^m}\hat{\mathbf{b}}_\omega^m + \frac{\partial\Delta\mathbf{R}_{\tau_m}^{t_i}}{\partial t}\hat{\delta}_{t_m}\right)\left(\mathbf{R}_I^L\mathbf{x}_L^i + \mathbf{p}_I^L\right) + \right.\\ &\left.\Delta\bar{\mathbf{p}}_{\tau_m}^{t_i} + \frac{\partial\Delta\mathbf{p}_{\tau_m}^{t_i}}{\partial\mathbf{b}_f^m}\hat{\mathbf{b}}_f^m + \frac{\partial\Delta\mathbf{p}_{\tau_m}^{t_i}}{\partial\mathbf{b}_\omega^m}\hat{\mathbf{b}}_\omega^m + \frac{\partial\Delta\mathbf{p}_{\tau_m}^{t_i}}{\partial t}\hat{\delta}_{t_m}\right) + \mathbf{p}_W^{\tau_m} + \Delta t_m^i\mathbf{v}_W^{\tau_m} + \frac{1}{2}\Delta {t_m^i}^2\mathbf{g}.\end{aligned} \tag{C.30}$$

$$\frac{\partial \mathbf{x}_W^i}{\partial [\mathbf{R}_W^{\tau_m}]_{(:)^T}} = \begin{bmatrix} \left[\mathbf{q}_1 \left(\Delta \mathbf{R}_{\tau_m}^{t_i} \left(\mathbf{R}_I^L \mathbf{x}_L^i + \mathbf{p}_I^L\right) + \Delta \mathbf{p}_{\tau_m}^{t_i}\right)^\top\right]_{(:)^T} \\ \left[\mathbf{q}_2 \left(\Delta \mathbf{R}_{\tau_m}^{t_i} \left(\mathbf{R}_I^L \mathbf{x}_L^i + \mathbf{p}_I^L\right) + \Delta \mathbf{p}_{\tau_m}^{t_i}\right)^\top\right]_{(:)^T} \\ \left[\mathbf{q}_3 \left(\Delta \mathbf{R}_{\tau_m}^{t_i} \left(\mathbf{R}_I^L \mathbf{x}_L^i + \mathbf{p}_I^L\right) + \Delta \mathbf{p}_{\tau_m}^{t_i}\right)^\top\right]_{(:)^T} \end{bmatrix} \tag{C.31}$$

$$\frac{\partial \mathbf{x}_W^i}{\partial \mathbf{v}_W^{\tau_m}} = \Delta t^i{}_m \mathbf{I}_{3\times 3} \tag{C.32}$$

$$\frac{\partial \mathbf{x}_W^i}{\partial \mathbf{p}_W^{\tau_m}} = \mathbf{I}_{3\times 3} \tag{C.33}$$

$$\frac{\partial \mathbf{x}_W^i}{\partial \hat{\mathbf{b}}_f^m} = \mathbf{R}_W^{\tau_m} \frac{\partial \Delta \mathbf{p}_{\tau_m}^{t_i}}{\partial \mathbf{b}_f^m} \tag{C.34}$$

$$\frac{\partial \mathbf{x}_W^i}{\partial \hat{\mathbf{b}}_\omega^m} = \begin{bmatrix} \left[\left(\mathbf{q}_1{}^\top \mathbf{R}_W^{t_i} \Delta \bar{\mathbf{R}}_{\tau_m}^{t_i}\right)^\top \left(\mathbf{R}_I^L \mathbf{x}_L^i + \mathbf{p}_I^L\right)^\top\right]_{(:)^T} \\ \left[\left(\mathbf{q}_2{}^\top \mathbf{R}_W^{t_i} \Delta \bar{\mathbf{R}}_{\tau_m}^{t_i}\right)^\top \left(\mathbf{R}_I^L \mathbf{x}_L^i + \mathbf{p}_I^L\right)^\top\right]_{(:)^T} \\ \left[\left(\mathbf{q}_3{}^\top \mathbf{R}_W^{t_i} \Delta \bar{\mathbf{R}}_{\tau_m}^{t_i}\right)^\top \left(\mathbf{R}_I^L \mathbf{x}_L^i + \mathbf{p}_I^L\right)^\top\right]_{(:)^T} \end{bmatrix} \mathbf{J}_{Exp} \frac{\partial \Delta \mathbf{R}_{\tau_m}^{t_i}}{\partial \mathbf{b}_\omega^m} + \mathbf{R}_W^{\tau_m} \frac{\partial \Delta \mathbf{p}_{\tau_m}^{t_i}}{\partial \mathbf{b}_\omega^m} \tag{C.35}$$

$$\frac{\partial \mathbf{x}_W^i}{\partial \hat{\delta}_{t_m}} = \begin{bmatrix} \left[\left(\mathbf{q}_1{}^\top \mathbf{R}_W^{t_i} \Delta \bar{\mathbf{R}}_{\tau_m}^{t_i}\right)^\top \left(\mathbf{R}_I^L \mathbf{x}_L^i + \mathbf{p}_I^L\right)^\top\right]_{(:)^T} \\ \left[\left(\mathbf{q}_2{}^\top \mathbf{R}_W^{t_i} \Delta \bar{\mathbf{R}}_{\tau_m}^{t_i}\right)^\top \left(\mathbf{R}_I^L \mathbf{x}_L^i + \mathbf{p}_I^L\right)^\top\right]_{(:)^T} \\ \left[\left(\mathbf{q}_3{}^\top \mathbf{R}_W^{t_i} \Delta \bar{\mathbf{R}}_{\tau_m}^{t_i}\right)^\top \left(\mathbf{R}_I^L \mathbf{x}_L^i + \mathbf{p}_I^L\right)^\top\right]_{(:)^T} \end{bmatrix} \mathbf{J}_{Exp} \frac{\partial \Delta \mathbf{R}_{\tau_m}^{t_i}}{\partial \delta_t} + \mathbf{R}_W^{\tau_m} \frac{\partial \Delta \mathbf{p}_{\tau_m}^{t_i}}{\partial \delta_t} \tag{C.36}$$

with $\mathbf{J}_{Exp}$ the jacobian of the exponential mapping (matrix 9x3).

C.3.2 Point-to-plane

$$\begin{aligned}\frac{\partial d_p}{\partial \mathbf{x}_W^{i,j,,l}} =& \frac{1}{\|(\mathbf{x}_W^j - \mathbf{x}_W) \times (\mathbf{x}_W^j - \mathbf{x}_W^l)\|_2^2} \\ & \left(\frac{\partial (\mathbf{x}_W^i - \mathbf{x}_W^j)^\top \Big((\mathbf{x}_W^j - \mathbf{x}_W) \times (\mathbf{x}_W^j - \mathbf{x}_W^l) \Big)}{\partial \mathbf{x}_W^{i,j,,l}} \|(\mathbf{x}_W^j - \mathbf{x}_W) \times (\mathbf{x}_W^j - \mathbf{x}_W^l)\|_2 \right. \\ & \left. - \frac{\partial \|(\mathbf{x}_W^j - \mathbf{x}_W) \times (\mathbf{x}_W^j - \mathbf{x}_W^l)\|_2}{\partial \mathbf{x}_W^{i,j,,l}} (\mathbf{x}_W^i - \mathbf{x}_W^j)^\top \Big((\mathbf{x}_W^j - \mathbf{x}_W) \times (\mathbf{x}_W^j - \mathbf{x}_W^l) \Big) \right)\end{aligned} \tag{C.37}$$

with

$$\begin{aligned}&\frac{\partial (\mathbf{x}_W^i - \mathbf{x}_W^j)^\top \Big((\mathbf{x}_W^j - \mathbf{x}_W) \times (\mathbf{x}_W^j - \mathbf{x}_W^l) \Big)}{\partial \mathbf{x}_W^{i,j,,l}} = \\ &\begin{bmatrix} \Big((\mathbf{x}_W^j - \mathbf{x}_W) \times (\mathbf{x}_W^j - \mathbf{x}_W^l) \Big)^\top, \\ \Big((\mathbf{x}_W^j - \mathbf{x}_W^l) \times (\mathbf{x}_W^j - \mathbf{x}_W) + (\mathbf{x}_W^j - \mathbf{x}_W^i) \times (\mathbf{x}_W - \mathbf{x}_W^l) \Big)^\top, \\ \Big((\mathbf{x}_W^i - \mathbf{x}_W^j) \times (\mathbf{x}_W^j - \mathbf{x}_W^l) \Big)^\top, \\ \Big((\mathbf{x}_W^j - \mathbf{x}_W) \times (\mathbf{x}_W^i - \mathbf{x}_W^j) \Big)^\top \end{bmatrix}\end{aligned} \tag{C.38}$$

$$\begin{aligned}&\frac{\partial \|(\mathbf{x}_W^j - \mathbf{x}_W) \times (\mathbf{x}_W^j - \mathbf{x}_W^l)\|_2}{\partial \mathbf{x}_W^{i,j,,l}} = \\ &\qquad \frac{\Big((\mathbf{x}_W^j - \mathbf{x}_W) \times (\mathbf{x}_W^j - \mathbf{x}_W^l) \Big)^\top}{\|(\mathbf{x}_W^j - \mathbf{x}_W) \times (\mathbf{x}_W^j - \mathbf{x}_W^l)\|_2} \frac{\partial (\mathbf{x}_W^j - \mathbf{x}_W) \times (\mathbf{x}_W^j - \mathbf{x}_W^l)}{\partial \mathbf{x}_W^{i,j,}}\end{aligned} \tag{C.39}$$

$$\frac{\partial(\mathbf{x}_W^j - \mathbf{x}_W) \times (\mathbf{x}_W^j - \mathbf{x}_W^l)}{\partial \mathbf{x}_W^{i,j,,l}} = \left[\mathbf{0}_{3\times 3} \quad , \quad (\mathbf{x}_W^l - \mathbf{x}_W)^\wedge \quad , \quad (\mathbf{x}_W^j - \mathbf{x}_W^l)^\wedge \quad , \quad (\mathbf{x}_W - \mathbf{x}_W^j)^\wedge\right] \tag{C.40}$$

C.3.3 Point-to-line

$$\frac{\partial d_l}{\partial \mathbf{x}_W^{i,j,}} = \frac{\frac{\partial \|(\mathbf{x}_W^i - \mathbf{x}_W^j) \times (\mathbf{x}_W^i - \mathbf{x}_W)\|_2}{\partial \mathbf{x}_W^{i,j,}} \|(\mathbf{x}_W^j - \mathbf{x}_W)\|_2 - \frac{\partial \|(\mathbf{x}_W^j - \mathbf{x}_W)\|_2}{\partial \mathbf{x}_W^{i,j,}} \|(\mathbf{x}_W^i - \mathbf{x}_W^j) \times (\mathbf{x}_W^i - \mathbf{x}_W)\|_2}{\|(\mathbf{x}_W^j - \mathbf{x}_W)\|_2^2} \tag{C.41}$$

with

$$\frac{\partial \|(\mathbf{x}_W^i - \mathbf{x}_W^j) \times (\mathbf{x}_W^i - \mathbf{x}_W)\|_2}{\partial \mathbf{x}_W^{i,j,}} = \frac{\left((\mathbf{x}_W^i - \mathbf{x}_W^j) \times (\mathbf{x}_W^i - \mathbf{x}_W)\right)^\top}{\|(\mathbf{x}_W^i - \mathbf{x}_W^j) \times (\mathbf{x}_W^i - \mathbf{x}_W)\|_2} \frac{\partial (\mathbf{x}_W^i - \mathbf{x}_W^j) \times (\mathbf{x}_W^i - \mathbf{x}_W)}{\partial \mathbf{x}_W^{i,j,}}, \tag{C.42}$$

$$\frac{\partial \|(\mathbf{x}_W^j - \mathbf{x}_W)\|_2}{\partial \mathbf{x}_W^{i,j,}} = \frac{(\mathbf{x}_W^j - \mathbf{x}_W)^\top}{\|(\mathbf{x}_W^j - \mathbf{x}_W)\|_2} \frac{\partial (\mathbf{x}_W^j - \mathbf{x}_W)}{\partial \mathbf{x}_W^{i,j,}} \tag{C.43}$$

and

$$\frac{\partial (\mathbf{x}_W^i - \mathbf{x}_W^j) \times (\mathbf{x}_W^i - \mathbf{x}_W)}{\partial \mathbf{x}_W^{i,j,}} = \left[(\mathbf{x}_W - \mathbf{x}_W^j)^\wedge \quad , \quad (\mathbf{x}_W^i - \mathbf{x}_W)^\wedge \quad , \quad (\mathbf{x}_W^j - \mathbf{x}_W^i)^\wedge\right] \tag{C.44}$$

$$\frac{\partial\left(\mathbf{x}_W^j - \mathbf{x}_W\right)}{\partial \mathbf{x}_W^{i,j,}} = \begin{bmatrix} \mathbf{0}_{3\times3} & , & \mathbf{I}_{3\times3} & , & -\mathbf{I}_{3\times3} \end{bmatrix} \tag{C.45}$$

$$\frac{\partial \mathbf{x}_W^i}{\partial [\mathbf{R}_I^L]_{(:)^T}} = \begin{bmatrix} \left[\left(\mathbf{R}_W^{\tau_m} \Delta\mathbf{R}_{\tau_m}^{t_i}\right)^\top \mathbf{q}_1 \mathbf{x}_L^{i\top}\right]_{(:)^T} \\ \left[\left(\mathbf{R}_W^{\tau_m} \Delta\mathbf{R}_{\tau_m}^{t_i}\right)^\top \mathbf{q}_2 \mathbf{x}_L^{i\top}\right]_{(:)^T} \\ \left[\left(\mathbf{R}_W^{\tau_m} \Delta\mathbf{R}_{\tau_m}^{t_i}\right)^\top \mathbf{q}_3 \mathbf{x}_L^{i\top}\right]_{(:)^T} \end{bmatrix} \tag{C.46}$$

$$\frac{\partial \mathbf{x}_W^i}{\partial \mathbf{p}_I^L} = \mathbf{R}_W^{\tau_m} \Delta\mathbf{R}_{\tau_m}^{t_i}. \tag{C.47}$$

C.3.4 Noise propagation

$$\frac{\partial \mathbf{x}_W^i}{\partial (\Delta\phi_m^i, \Delta\mathbf{p}_{\tau_m}^{t_i})} = \begin{bmatrix} \begin{bmatrix} \left[\left(\mathbf{q}_1{}^\top \mathbf{R}_W^{t_i}\right)^\top \left(\mathbf{R}_I^L \mathbf{x}_L^i + \mathbf{p}_I^L\right)^\top\right]_{(:)^T} \\ \left[\left(\mathbf{q}_2{}^\top \mathbf{R}_W^{t_i}\right)^\top \left(\mathbf{R}_I^L \mathbf{x}_L^i + \mathbf{p}_I^L\right)^\top\right]_{(:)^T} \\ \left[\left(\mathbf{q}_3{}^\top \mathbf{R}_W^{t_i}\right)^\top \left(\mathbf{R}_I^L \mathbf{x}_L^i + \mathbf{p}_I^L\right)^\top\right]_{(:)^T} \end{bmatrix} \mathbf{J}_{Exp}; & \mathbf{R}_W^{t_i} \end{bmatrix} \tag{C.48}$$

$$\frac{\partial \mathbf{x}_W^i}{\partial \mathbf{x}_L^i} = \mathbf{R}_W^{t_i} \Delta\mathbf{R}_{\tau_m}^{t_i} \mathbf{R}_I^L \tag{C.49}$$

Appendix D

IDOL Jacobians

This appendix provides the reader with the different Jacobians involved in IDOL's optimisation. Our implementation uses Ceres and its *local parameterisation* feature. Consequently, the Jacobians in this section are derived with respect to the over-parameterised SO(3) variables. Note that many computations could be further optimised as many matrices are quite sparse.

The following notation is employed:

- $[\mathbf{M}]_{(:)^T}$ the operator that transforms a r-by-c matrix $\mathbf{M}$ into a rc-by-1 vector.
- $[\mathbf{M}]_{(j,:)}$ the operator that isolates the j^{th} row of $\mathbf{M}$.
- $[\mathbf{M}]_{(:,j)}$ the operator that isolates the j^{th} column of $\mathbf{M}$.

Examples:

$$\mathbf{M} = \begin{bmatrix} m_1 & m_4 & m_7 \\ m_2 & m_5 & m_8 \\ m_3 & m_6 & m_9 \end{bmatrix} \quad \text{then} \quad [\mathbf{M}]_{(:)^T} = \begin{bmatrix} m_1 & m_2 & m_3 & m_4 & m_5 & m_6 & m_7 & m_8 & m_9 \end{bmatrix}, \tag{D.1}$$

$$\mathbf{M} = \begin{bmatrix} m_1 & m_4 & m_7 \\ m_2 & m_5 & m_8 \\ m_3 & m_6 & m_9 \end{bmatrix} \quad \text{then} \quad [\mathbf{M}]_{(1,:)} = \begin{bmatrix} m_1 & m_4 & m_7 \end{bmatrix}, \tag{D.2}$$

and

$$\mathbf{M} = \begin{bmatrix} m_1 & m_4 & m_7 \\ m_2 & m_5 & m_8 \\ m_3 & m_6 & m_9 \end{bmatrix} \quad \text{then} \quad [\mathbf{M}]_{(:,1)} = \begin{bmatrix} m_1 \\ m_2 \\ m_3 \end{bmatrix} \tag{D.3}$$

Additionally, we define $\Delta t_m^i = t_i - \tau_m$, $\text{Log}(\bullet) = \log(\bullet^\vee)$, $\text{Exp}(\bullet) = \exp(\bullet^\wedge)$, $\mathbf{q}_1 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^\top$, $\mathbf{q}_2 = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^\top$, and $\mathbf{q}_3 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^\top$.

D.1 Event-to-line

The Jacobian of the event-to-line distance is

$$\frac{\partial r_e^i}{\partial \mathbf{d}_{a_l}^{t_i}, \mathbf{d}_{b_l}^{t_i}} = \frac{\frac{\partial \left(\mathbf{e}^i m - \mathbf{d}_{a_l}^{t_i}\right) \times \left(\mathbf{d}_{b_l}^{t_i} - \mathbf{d}_{a_l}^{t_i}\right)}{\partial \mathbf{d}_{a_l}^{t_i}, \mathbf{d}_{b_l}^{t_i}} \| \left(\mathbf{d}_{b_l}^{t_i} - \mathbf{d}_{a_l}^{t_i}\right) \|_2 - \frac{\partial \| \left(\mathbf{d}_{b_l}^{t_i} - \mathbf{d}_{a_l}^{t_i}\right) \|_2}{\partial \mathbf{d}_{a_l}^{t_i}, \mathbf{d}_{b_l}^{t_i}} \left(\left(\mathbf{e}^i m - \mathbf{d}_{a_l}^{t_i}\right) \times \left(\mathbf{d}_{b_l}^{t_i} - \mathbf{d}_{a_l}^{t_i}\right) \right)}{\| \left(\mathbf{d}_{b_l}^{t_i} - \mathbf{d}_{a_l}^{t_i}\right) \|_2^2} \tag{D.4}$$

with

$$\frac{\partial \| \left(\mathbf{d}_{b_l}^{t_i} - \mathbf{d}_{a_l}^{t_i}\right) \|_2}{\partial \mathbf{d}_{a_l}^{t_i}, \mathbf{d}_{b_l}^{t_i}} = \frac{\left(\mathbf{d}_{b_l}^{t_i} - \mathbf{d}_{a_l}^{t_i}\right)^\top}{\| \left(\mathbf{d}_{b_l}^{t_i} - \mathbf{d}_{a_l}^{t_i}\right) \|} \frac{\partial \left(\mathbf{d}_{b_l}^{t_i} - \mathbf{d}_{a_l}^{t_i}\right)}{\partial \mathbf{d}_{b_l}^{t_i} \mathbf{d}_{a_l}^{t_i}} \tag{D.5}$$

$$\frac{\partial \left(\mathbf{d}_{b_l}^{t_i} - \mathbf{d}_{a_l}^{t_i}\right)}{\partial \mathbf{d}_{a_l}^{t_i}, \mathbf{d}_{b_l}^{t_i}} = \begin{bmatrix} -\mathbf{I}_{2\times 2}, & \mathbf{I}_{2\times 2} \end{bmatrix} \tag{D.6}$$

and

$$\frac{\partial\left(\mathbf{e}^{i}m-\mathbf{d}_{a_l}^{t_i}\right)\times\left(\mathbf{d}_{b_l}^{t_i}-\mathbf{d}_{a_l}^{t_i}\right)}{\partial\mathbf{d}_{a_l}^{t_i},\mathbf{d}_{b_l}^{t_i}}= \\ \left[\mathbf{e}^{i}m_{[1]}-\mathbf{d}_{b_l\,[1]}^{t_i},\quad \mathbf{d}_{b_l\,[0]}^{t_i}-\mathbf{e}^{i}m_{[0]},\quad \mathbf{d}_{a_l\,[1]}^{t_i}-\mathbf{e}^{i}m_{[1]},\quad \mathbf{e}^{i}m_{[0]}-\mathbf{d}_{a_l\,[0]}^{t_i}\right]. \tag{D.7}$$

The "symbol" $\bullet_{[n]}$ represents the n^{th} component of the vector $\bullet$.

D.2 Projection from 3D to 2D

Given the camera matrix $\mathbf{K}$:

$$\frac{\partial\mathbf{d}_{\bullet_l}^{t_i}}{\partial\mathbf{l}_{\bullet_{t_i}}^{l}}=\frac{\partial\mathbf{d}_{\bullet_l}^{t_i}}{\partial\mathbf{Kl}_{\bullet_{t_i}}^{l}}\mathbf{K}^{\top}=\frac{\begin{bmatrix}\mathbf{Kl}_{\bullet_{t_i}[2]}^{l} & 0 & -\mathbf{Kl}_{\bullet_{t_i}[0]}^{l}\\ 0 & \mathbf{Kl}_{\bullet_{t_i}[2]}^{l} & -\mathbf{Kl}_{\bullet_{t_i}[1]}^{l}\end{bmatrix}}{{\mathbf{Kl}_{\bullet_{t_i}[2]}^{l}}^{2}}\mathbf{K}^{\top}. \tag{D.8}$$

D.2.1 3D transformation

The reprojection

$$\mathbf{l}_{\bullet_{t_i}}^{l}={\mathbf{R}_{W}^{C}}^{\top}\mathbf{l}_{\bullet_W}^{l}-{\mathbf{R}_{W}^{C}}^{\top}\mathbf{p}_{W}^{C} \tag{D.9}$$

$$\mathbf{T}_{W}^{C}(t)=\mathbf{T}_{W}^{t_i}\mathbf{T}_{I}^{C} \tag{D.10}$$

Consequently,

$$\mathbf{R}_{W}^{C}(t)=\mathbf{R}_{W}^{t_i}\mathbf{R}_{I}^{C} \tag{D.11}$$

and

$$\mathbf{p}_{W}^{C}(t)=\mathbf{p}_{W}^{t_i}+\mathbf{R}_{W}^{t_i}\mathbf{p}_{I}^{C} \tag{D.12}$$

$$\mathbf{R}_W^{t_i} = \mathbf{R}_W^{\tau_m} \Delta\mathbf{R}_{\tau_m}^{t_i} \tag{D.13}$$

$$\mathbf{p}_W^{t_i} = \mathbf{p}_W^{\tau_m} + \mathbf{v}_W^{\tau_m} \Delta t_m^i + \frac{1}{2}\mathbf{g}\Delta {t_m^i}^2 + \mathbf{R}_W^{\tau_m} \Delta\mathbf{p}_{\tau_m}^{t_i} \tag{D.14}$$

Combining the previous equations we get

$$\mathbf{l}_{\bullet_{t_i}}^{l} = {\mathbf{R}_I^C}^\top {\Delta\mathbf{R}_{\tau_m}^{t_i}}^\top {\mathbf{R}_W^{\tau_m}}^\top \big(\mathbf{l}_{\bullet W}^{l} - \mathbf{p}_W^{\tau_m} - \mathbf{v}_W^{\tau_m}\Delta t_m^i - \frac{1}{2}\mathbf{g}\Delta {t_m^i}^2\big) - {\mathbf{R}_I^C}^\top {\Delta\mathbf{R}_{\tau_m}^{t_i}}^\top \Delta\mathbf{p}_{\tau_m}^{t_i} - {\mathbf{R}_I^C}^\top \mathbf{p}_I^C \tag{D.15}$$

$$\frac{\partial \mathbf{l}_{\bullet_{t_i}}^{l}}{\partial [\mathbf{R}_W^{\tau_m}]_{(:)^T}} = \begin{bmatrix} \Big[\big(\mathbf{l}_{\bullet W}^{l} - \mathbf{p}_W^{\tau_m} - \mathbf{v}_W^{\tau_m}\Delta t_m^i - \frac{1}{2}\mathbf{g}\Delta {t_m^i}^2\big)\mathbf{q}_1^\top {\mathbf{R}_I^C}^\top {\Delta\mathbf{R}_{\tau_m}^{t_i}}^\top\Big]_{(:)^T} \\ \Big[\big(\mathbf{l}_{\bullet W}^{l} - \mathbf{p}_W^{\tau_m} - \mathbf{v}_W^{\tau_m}\Delta t_m^i - \frac{1}{2}\mathbf{g}\Delta {t_m^i}^2\big)\mathbf{q}_2^\top {\mathbf{R}_I^C}^\top {\Delta\mathbf{R}_{\tau_m}^{t_i}}^\top\Big]_{(:)^T} \\ \Big[\big(\mathbf{l}_{\bullet W}^{l} - \mathbf{p}_W^{\tau_m} - \mathbf{v}_W^{\tau_m}\Delta t_m^i - \frac{1}{2}\mathbf{g}\Delta {t_m^i}^2\big)\mathbf{q}_3^\top {\mathbf{R}_I^C}^\top {\Delta\mathbf{R}_{\tau_m}^{t_i}}^\top\Big]_{(:)^T} \end{bmatrix} \tag{D.16}$$

$$\frac{\partial \mathbf{l}_{\bullet_{t_i}}^{l}}{\partial \mathbf{l}_{\bullet W}^{l}} = {\mathbf{R}_I^C}^\top {\Delta\mathbf{R}_{\tau_m}^{t_i}}^\top {\mathbf{R}_W^{\tau_m}}^\top \tag{D.17}$$

$$\frac{\partial \mathbf{l}_{\bullet_{t_i}}^{l}}{\partial \mathbf{p}_W^{\tau_m}} = -{\mathbf{R}_I^C}^\top {\Delta\mathbf{R}_{\tau_m}^{t_i}}^\top {\mathbf{R}_W^{\tau_m}}^\top \tag{D.18}$$

$$\frac{\partial \mathbf{l}_{\bullet_{t_i}}^{l}}{\partial \mathbf{v}_W^{\tau_m}} = -{\mathbf{R}_I^C}^\top {\Delta\mathbf{R}_{\tau_m}^{t_i}}^\top {\mathbf{R}_W^{\tau_m}}^\top \Delta t_m^i \tag{D.19}$$

$$\frac{\partial \mathbf{l}_{\bullet_{t_i}}^{l}}{\partial \mathbf{b}_f^m} = -{\mathbf{R}_I^C}^\top {\Delta\mathbf{R}_{\tau_m}^{t_i}}^\top \frac{\partial \Delta\mathbf{p}_{\tau_m}^{t_i}}{\partial \mathbf{b}_f} \tag{D.20}$$

$$\frac{\partial \mathbf{l}_{\bullet_{t_i}}^{l}}{\partial \mathbf{b}_\omega^m} = \frac{\partial {\mathbf{R}_I^C}^\top {\Delta\mathbf{R}_{\tau_m}^{t_i}}^\top {\mathbf{R}_W^{\tau_m}}^\top \big(\mathbf{l}_{\bullet W}^{l} - \mathbf{p}_W^{\tau_m} - \mathbf{v}_W^{\tau_m}\Delta t_m^i - \frac{1}{2}\mathbf{g}\Delta {t_m^i}^2\big)}{\partial \mathbf{b}_\omega^m} - \frac{\partial {\mathbf{R}_I^C}^\top {\Delta\mathbf{R}_{\tau_m}^{t_i}}^\top \Delta\mathbf{p}_{\tau_m}^{t_i}}{\mathbf{b}_\omega^m} \tag{D.21}$$

with

$$\frac{\partial \mathbf{R}_I^{C\top} \Delta \mathbf{R}_{\tau_m}^{t_i}{}^{\top} \mathbf{R}_W^{\tau_m\top} \left(\mathbf{l}_\bullet{}_W^l - \mathbf{p}_W^{\tau_m} - \mathbf{v}_W^{\tau_m} \Delta t_m^i - \frac{1}{2}\mathbf{g}\Delta t_m^i{}^2\right)}{\partial \mathbf{b}_\omega^m} = \frac{\partial \mathbf{R}_I^{C\top} \Delta \mathbf{R}_{\tau_m}^{t_i}{}^{\top} \mathbf{R}_W^{\tau_m\top} \left(\mathbf{l}_\bullet{}_W^l - \mathbf{p}_W^{\tau_m} - \mathbf{v}_W^{\tau_m} \Delta t_m^i - \frac{1}{2}\mathbf{g}\Delta t_m^i{}^2\right)}{\partial [\Delta \mathbf{R}_{\tau_m}^{t_i}]_{(:)^T}} \frac{\partial [\Delta \mathbf{R}_{\tau_m}^{t_i}]_{(:)^T}}{\partial \mathbf{b}_\omega^m} \tag{D.22}$$

$$\frac{\partial \mathbf{R}_I^{C\top} \Delta \mathbf{R}_{\tau_m}^{t_i}{}^{\top} \mathbf{R}_W^{\tau_m\top} \left(\mathbf{l}_\bullet{}_W^l - \mathbf{p}_W^{\tau_m} - \mathbf{v}_W^{\tau_m} \Delta t_m^i - \frac{1}{2}\mathbf{g}\Delta t_m^i{}^2\right)}{\partial [\Delta \mathbf{R}_{\tau_m}^{t_i}]_{(:)^T}} = \begin{bmatrix} \left[\mathbf{R}_W^{\tau_m\top} \left(\mathbf{l}_\bullet{}_W^l - \mathbf{p}_W^{\tau_m} - \mathbf{v}_W^{\tau_m} \Delta t_m^i - \frac{1}{2}\mathbf{g}\Delta t_m^i{}^2\right) \mathbf{q}_1^\top \mathbf{R}_I^{C\top}\right]_{(:)^T} \\ \left[\mathbf{R}_W^{\tau_m\top} \left(\mathbf{l}_\bullet{}_W^l - \mathbf{p}_W^{\tau_m} - \mathbf{v}_W^{\tau_m} \Delta t_m^i - \frac{1}{2}\mathbf{g}\Delta t_m^i{}^2\right) \mathbf{q}_2^\top \mathbf{R}_I^{C\top}\right]_{(:)^T} \\ \left[\mathbf{R}_W^{\tau_m\top} \left(\mathbf{l}_\bullet{}_W^l - \mathbf{p}_W^{\tau_m} - \mathbf{v}_W^{\tau_m} \Delta t_m^i - \frac{1}{2}\mathbf{g}\Delta t_m^i{}^2\right) \mathbf{q}_3^\top \mathbf{R}_I^{C\top}\right]_{(:)^T} \end{bmatrix} \tag{D.23}$$

$$\begin{aligned} \frac{\partial [\Delta \mathbf{R}_{\tau_m}^{t_i}]_{(:)^T}}{\partial \mathbf{b}_\omega^m} &= \frac{\partial \left[\bar{\Delta \mathbf{R}}_{\tau_m}^{t_i} \mathrm{Exp}\left(\frac{\partial \Delta \mathbf{R}_{\tau_m}^{t_i}}{\partial \mathbf{b}_\omega} \mathbf{b}_\omega^m\right)\right]_{(:)^T}}{\partial \mathbf{b}_\omega^m} \\ &= \frac{\partial \left[\bar{\Delta \mathbf{R}}_{\tau_m}^{t_i} \mathrm{Exp}\left(\frac{\partial \Delta \mathbf{R}_{\tau_m}^{t_i}}{\partial \mathbf{b}_\omega} \mathbf{b}_\omega^m\right)\right]_{(:)^T}}{\partial [\mathrm{Exp}\left(\frac{\partial \Delta \mathbf{R}_{\tau_m}^{t_i}}{\partial \mathbf{b}_\omega} \mathbf{b}_\omega^m\right)]_{(:)^T}} \frac{[\partial \mathrm{Exp}\left(\frac{\partial \Delta \mathbf{R}_{\tau_m}^{t_i}}{\partial \mathbf{b}_\omega} \mathbf{b}_\omega^m\right)]_{(:)^T}}{\partial \left(\frac{\partial \Delta \mathbf{R}_{\tau_m}^{t_i}}{\partial \mathbf{b}_\omega} \mathbf{b}_\omega^m\right)} \frac{\partial \left(\frac{\partial \Delta \mathbf{R}_{\tau_m}^{t_i}}{\partial \mathbf{b}_\omega} \mathbf{b}_\omega^m\right)}{\partial \mathbf{b}_\omega^m} \\ &= \begin{bmatrix} \bar{\Delta \mathbf{R}}_{\tau_m}^{t_i} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \bar{\Delta \mathbf{R}}_{\tau_m}^{t_i} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \bar{\Delta \mathbf{R}}_{\tau_m}^{t_i} \end{bmatrix} \mathbf{J}_{\mathrm{Exp}}\left(\frac{\partial \Delta \mathbf{R}_{\tau_m}^{t_i}}{\partial \mathbf{b}_\omega} \mathbf{b}_\omega^m\right) \frac{\partial \Delta \mathbf{R}_{\tau_m}^{t_i}}{\partial \mathbf{b}_\omega} \end{aligned} \tag{D.24}$$

and

$$\frac{\partial \mathbf{R}_I^{C\top} \Delta \mathbf{R}_{\tau_m}^{t_i}{}^{\top} \Delta \mathbf{p}_{\tau_m}^{t_i}}{\partial \mathbf{b}_\omega^m} = \begin{bmatrix} \frac{\partial \mathbf{q}_1^\top \mathbf{R}_I^{C\top} \Delta \mathbf{R}_{\tau_m}^{t_i}{}^{\top} \Delta \mathbf{p}_{\tau_m}^{t_i}}{\partial \mathbf{b}_\omega^m} \\ \frac{\partial \mathbf{q}_2^\top \mathbf{R}_I^{C\top} \Delta \mathbf{R}_{\tau_m}^{t_i}{}^{\top} \Delta \mathbf{p}_{\tau_m}^{t_i}}{\partial \mathbf{b}_\omega^m} \\ \frac{\partial \mathbf{q}_3^\top \mathbf{R}_I^{C\top} \Delta \mathbf{R}_{\tau_m}^{t_i}{}^{\top} \Delta \mathbf{p}_{\tau_m}^{t_i}}{\partial \mathbf{b}_\omega^m} \end{bmatrix} \tag{D.25}$$

$$\frac{\partial \mathbf{q}_\bullet^\top \mathbf{R}_I^{C\top} \Delta \mathbf{R}_{\tau_m}^{t_i}{}^{\top} \Delta \mathbf{p}_{\tau_m}^{t_i}}{\partial \mathbf{b}_\omega^m} = \mathbf{q}_\bullet^\top \mathbf{R}_I^{C\top} \Delta \mathbf{R}_{\tau_m}^{t_i}{}^{\top} \frac{\partial \Delta \mathbf{p}_{\tau_m}^{t_i}}{\partial \mathbf{b}_\omega^m} + \Delta \mathbf{p}_{\tau_m}^{t_i}{}^{\top} \frac{\partial \mathbf{q}_\bullet^\top \mathbf{R}_I^{C\top} \Delta \mathbf{R}_{\tau_m}^{t_i}{}^{\top}}{\partial \mathbf{b}_\omega^m} \tag{D.26}$$

with

$$\frac{\partial \Delta \mathbf{p}_{\tau_m}^{t_i}}{\partial \mathbf{b}_{\omega}^{m}} = \frac{\partial \Delta \mathbf{p}_{m}^{i}}{\partial \mathbf{b}_{\omega}} \tag{D.27}$$

and

$$\frac{\partial \mathbf{q}_{\bullet}^{\top} \mathbf{R}_I^{C\top} \Delta \mathbf{R}_{\tau_m}^{t_i}{}^{\top}}{\partial \mathbf{b}_{\omega}^{m}} = \frac{\partial \mathbf{q}_{\bullet}^{\top} \mathbf{R}_I^{C\top} \Delta \mathbf{R}_{\tau_m}^{t_i}{}^{\top}}{\partial [\Delta \mathbf{R}_{\tau_m}^{t_i}]_{(:)^T}} \frac{\partial [\Delta \mathbf{R}_{\tau_m}^{t_i}]_{(:)^T}}{\partial \mathbf{b}_{\omega}^{m}} \tag{D.28}$$

$$\begin{aligned} \frac{\partial \mathbf{q}_{\bullet}^{\top} \mathbf{R}_I^{C\top} \Delta \mathbf{R}_{\tau_m}^{t_i}{}^{\top}}{\partial \Delta \mathbf{R}_{\tau_m}^{t_i}} &= \begin{bmatrix} \frac{\partial \mathbf{q}_{\bullet}^{\top} \mathbf{R}_I^{C\top} \Delta \mathbf{R}_{\tau_m}^{t_i}{}^{\top} \mathbf{q}_1}{\partial \Delta \mathbf{R}_{\tau_m}^{t_i}} \\ \frac{\partial \mathbf{q}_{\bullet}^{\top} \mathbf{R}_I^{C\top} \Delta \mathbf{R}_{\tau_m}^{t_i}{}^{\top} \mathbf{q}_2}{\partial \Delta \mathbf{R}_{\tau_m}^{t_i}} \\ \frac{\partial \mathbf{q}_{\bullet}^{\top} \mathbf{R}_I^{C\top} \Delta \mathbf{R}_{\tau_m}^{t_i}{}^{\top} \mathbf{q}_3}{\partial \Delta \mathbf{R}_{\tau_m}^{t_i}} \end{bmatrix} \\ &= \begin{bmatrix} [\mathbf{q}_1 \mathbf{q}_{\bullet}^{\top} \mathbf{R}_I^{C\top}]_{(:)^T} \\ [\mathbf{q}_2 \mathbf{q}_{\bullet}^{\top} \mathbf{R}_I^{C\top}]_{(:)^T} \\ [\mathbf{q}_3 \mathbf{q}_{\bullet}^{\top} \mathbf{R}_I^{C\top}]_{(:)^T} \end{bmatrix} \end{aligned} \tag{D.29}$$

D.3 Splitting force

If $l_i < 0$

$$\frac{\partial r_s^{\alpha^i}}{\partial \mathbf{d}_{a_l}^{t_i}, \mathbf{d}_{b_l}^{t_i}} = \frac{\frac{\partial \left(\mathbf{e}^i m - \mathbf{d}_{a_l}^{t_i}\right)^{\top} \left(\mathbf{d}_{b_l}^{t_i} - \mathbf{d}_{a_l}^{t_i}\right)}{\partial \mathbf{d}_{a_l}^{t_i}, \mathbf{d}_{b_l}^{t_i}} \|\left(\mathbf{d}_{b_l}^{t_i} - \mathbf{d}_{a_l}^{t_i}\right)\|_2 - \frac{\partial \|\left(\mathbf{d}_{b_l}^{t_i} - \mathbf{d}_{a_l}^{t_i}\right)\|_2}{\partial \mathbf{d}_{a_l}^{t_i}, \mathbf{d}_{b_l}^{t_i}} \left(\left(\mathbf{e}^i m - \mathbf{d}_{a_l}^{t_i}\right)^{\top} \left(\mathbf{d}_{b_l}^{t_i} - \mathbf{d}_{a_l}^{t_i}\right)\right)}{\|\left(\mathbf{d}_{b_l}^{t_i} - \mathbf{d}_{a_l}^{t_i}\right)\|_2^2} \tag{D.30}$$

$$\frac{\partial \left(\mathbf{e}^i m - \mathbf{d}_{a_l}^{t_i}\right)^{\top} \left(\mathbf{d}_{b_l}^{t_i} - \mathbf{d}_{a_l}^{t_i}\right)}{\partial \mathbf{d}_{a_l}^{t_i}, \mathbf{d}_{b_l}^{t_i}} = \left[(2\mathbf{d}_{a_l}^{t_i} - \mathbf{d}_{b_l}^{t_i} - \mathbf{e}^i m)^{\top}, \quad (\mathbf{e}^i m - \mathbf{d}_{b_l}^{t_i})^{\top}\right] \tag{D.31}$$

If $l_i > \|\mathbf{d}_{b_l}^{t_i} - \mathbf{d}_{a_l}^{t_i}\|$

$$\frac{\partial r_s^{\alpha^i}}{\partial \mathbf{d}_{a_l}^{t_i}, \mathbf{d}_{b_l}^{t_i}} = \frac{\partial \|\left(\mathbf{d}_{b_l}^{t_i} - \mathbf{d}_{a_l}^{t_i}\right)\|_2}{\partial \mathbf{d}_{a_l}^{t_i}, \mathbf{d}_{b_l}^{t_i}} - \left.\frac{\partial r_s^{\alpha^i}}{\partial \mathbf{d}_{a_l}^{t_i}, \mathbf{d}_{b_l}^{t_i}}\right|_{l_i<0} \tag{D.32}$$

If $0 < l_i < \|\mathbf{d}_{b_l}^{t_i} - \mathbf{d}_{a_l}^{t_i}\|$

$$\frac{\partial r_s^{\alpha^i}}{\partial \mathbf{d}_{a_l}^{t_i}, \mathbf{d}_{b_l}^{t_i}} = \mathbf{0}_{1\times 4} \tag{D.33}$$

D.4 Attraction force

$$\frac{\partial r_a^m}{\partial \mathbf{d}_{a_l}^{t_i}, \mathbf{d}_{b_l}^{t_i}} = \frac{\frac{\partial \|\mathbf{d}_{b_l}^{t_i} - \mathbf{d}_{a_l}^{t_i}\|_2}{\partial \mathbf{d}_{a_l}^{t_i}, \mathbf{d}_{b_l}^{t_i}}}{2\sqrt{\|\mathbf{d}_{b_l}^{t_i} - \mathbf{d}_{a_l}^{t_i}\|_2}} \tag{D.34}$$

with

$$\frac{\partial \|\mathbf{d}_{b_l}^{t_i} - \mathbf{d}_{a_l}^{t_i}\|_2}{\partial \mathbf{d}_{a_l}^{t_i}, \mathbf{d}_{b_l}^{t_i}} = \frac{\left(\mathbf{d}_{b_l}^{t_i} - \mathbf{d}_{a_l}^{t_i}\right)^\top}{\|\left(\mathbf{d}_{b_l}^{t_i} - \mathbf{d}_{a_l}^{t_i}\right)\|} \frac{\partial\left(\mathbf{d}_{b_l}^{t_i} - \mathbf{d}_{a_l}^{t_i}\right)}{\partial \mathbf{d}_{b_l}^{t_i}, \mathbf{d}_{a_l}^{t_i}} \tag{D.35}$$

$$\frac{\partial\left(\mathbf{d}_{b_l}^{t_i} - \mathbf{d}_{a_l}^{t_i}\right)}{\partial \mathbf{d}_{a_l}^{t_i}, \mathbf{d}_{b_l}^{t_i}} = \begin{bmatrix} -\mathbf{I}_{2\times 2}, & \mathbf{I}_{2\times 2} \end{bmatrix} \tag{D.36}$$

Bibliography

[1] Todd Lupton and Salah Sukkarieh. Visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions. *IEEE Transactions on Robotics*, 28(1):61–76, 2012.

[2] Christian Forster, Luca Carlone, Frank Dellaert, and Davide Scaramuzza. IMU preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation. *Robotics: Science and Systems*, pages 6–15, 2015.

[3] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: part i. *IEEE Robotics Automation Magazine*, 13(2):99–110, 2006.

[4] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, Jose Neira, Ian Reid, and John J. Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332, 2016.

[5] Hugh F. Durrant-Whyte. An autonomous guided vehicle for cargo handling applications. *International Journal of Robotics Research*, 15(5):407–440, 1996.

[6] Anastasios I. Mourikis and Stergios I. Roumeliotis. A multi-state constraint Kalman filter for vision-aided inertial navigation. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 3565–3572, 2007.

[7] Raul Mur-Artal and Juan D. Tardos. Visual-Inertial Monocular SLAM with Map Reuse. *IEEE Robotics and Automation Letters*, 2(2):796–803, 2017.

[8] Tong Qin, Peiliang Li, and Shaojie Shen. VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator. *IEEE Transactions on Robotics*, 34(4): 1–17, 2018.

[9] Michael Bosse, Robert Zlot, and Paul Flick. Zebedee : Design of a spring-mounted 3-D range sensor with application to mobile mapping. *IEEE Transactions on Robotics*, 28(October):1–15, 2012.

[10] Ji Zhang and Sanjiv Singh. LOAM : Lidar odometry and mapping in real-time. *Robotics: Science and Systems*, pages 7–15, 2014.

[11] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3354–3361, 2012.

[12] Mitch Bryson, Matthew Johnson-Roberson, and Salah Sukkarieh. Airborne smoothing and mapping using vision and inertial sensors. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 2037–2042, 2009.

[13] Ji Zhang and Sanjiv Singh. Enabling aggressive motion estimation at low-drift and accurate mapping in real-time. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 5051–5058, 2017.

[14] Patrick Geneva and Kevin Eckenhoff. LIPS: LiDAR-Inertial 3D Plane SLAM. *IEEE International Conference on Intelligent Robots and Systems*, 2018.

[15] Cedric Le Gentil, Teresa Vidal-Calleja, and Shoudong Huang. IN2LAMA : INertial Lidar Localisation And MApping. *IEEE International Conference on Robotics and Automation*, 2019.

[16] Jeffrey Delmerico and Davide Scaramuzza. A Benchmark Comparison of Monocular Visual-Inertial Odometry Algorithms for Flying Robots. *IEEE International Conference on Robotics and Automation*, pages 2502–2509, 2018.

[17] Christian Forster, Luca Carlone, Frank Dellaert, and Davide Scaramuzza. On-Manifold Preintegration for Real-Time Visual-Inertial Odometry. *IEEE Transactions on Robotics*, 33(1):1–21, 2017.

[18] Kevin Eckenhoff, Patrick Geneva, and Guoquan Huang. Closed-form preintegration methods for graph-based visualinertial navigation. *International Journal of Robotics Research*, 38(5):563–586, 2019.

[19] Cedric Le Gentil, Teresa Vidal-Calleja, and Shoudong Huang. 3D Lidar-IMU Calibration based on Upsampled Preintegrated Measurements for Motion Distortion Correction. *IEEE International Conference on Robotics and Automation*, 2018.

[20] Simo Särkkä. Linear operators and stochastic partial differential equations in Gaussian process regression. *Artificial Neural Networks and Machine Learning–ICANN 2011*, pages 151–158, 2011.

[21] Paul Furgale, Timothy D Barfoot, and Gabe Sibley. Continuous-Time Batch Estimation using Temporal Basis Functions. *IEEE International Conference on Robotics and Automation*, pages 2088–2095, 2012.

[22] Seungpyo Hong, Heedong Ko, and Jinwook Kim. VICP: Velocity updating iterative closest point algorithm. *Proceedings - IEEE International Conference on Robotics and Automation*, (Section 3):1893–1898, 2010.

[23] Michael Bosse and Robert Zlot. Continuous 3D Scan-Matching with a Spinning 2D Laser. *IEEE International Conference on Robotics and Automation*, 2009.

[24] Chanoh Park, Peyman Moghadam, Soohwan Kim, Alberto Elfes, Clinton Fookes, and Sridha Sridharan. Elastic LiDAR Fusion: Dense Map-Centric Continuous-Time SLAM. *IEEE International Conference on Robotics and Automation*, 2018.

[25] Sean Anderson and Timothy D Barfoot. Towards Relative Continuous-Time SLAM. *IEEE International Conference on Robotics and Automation*, (Ccd):1033–1040, 2013.

[26] Steven Lovegrove, Alonso Patron-Perez, and Gabe Sibley. Spline fusion: A continuous-time representation for visual-inertial fusion with application to rolling

shutter cameras. *BMVC 2013 - Electronic Proceedings of the British Machine Vision Conference 2013*, pages 1–12, 2013.

[27] Olivier A. Bauchau and Jou Young Choi. The vectorial parameterization of motion. *Proceedings of the ASME Design Engineering Technical Conference*, 5 A:11–20, 2003.

[28] Alonso Patron-Perez, Steven Lovegrove, and Gabe Sibley. A Spline-Based Trajectory Representation for Sensor Fusion and Rolling Shutter Cameras. *International Journal of Computer Vision*, 113(3):208–219, 2015.

[29] C E Rasmussen and C K I Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[30] Timothy D. Barfoot, Chi Hay Tong, and Simo Särkkä. Batch nonlinear continuous-time trajectory estimation as exactly sparse Gaussian process regression. *Robotics: Science and Systems*, 2014.

[31] Sean Anderson and Timothy D. Barfoot. Full STEAM ahead: Exactly sparse Gaussian process regression for batch continuous-time trajectory estimation on SE(3). *IEEE International Conference on Intelligent Robots and Systems*, 2015-Decem(3): 157–164, 2015.

[32] Timothy D. Barfoot. *State Estimation for Robotics*. 2017.

[33] Paul J. Besl and Neil D. McKay. A Method for Registration of 3-D Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1992.

[34] A Segal, D Haehnel, and S Thrun. Generalized-ICP. *Robotics: Science and Systems*, 5:168–176, 2009.

[35] Ellon Mendes, Pierrick Koch, and Simon Lacroix. ICP-based pose-graph SLAM. *IEEE International Symposium on Safety, Security and Rescue Robotics*, pages 195–200, 2016.

[36] David Droeschel and Sven Behnke. Efficient Continuous-time SLAM for 3D Lidar-based Online Mapping. *IEEE International Conference on Robotics and Automation (ICRA)*, (May):5000–5007, 2018.

[37] Haoyang Ye, Yuying Chen, and Ming Liu. Tightly coupled 3D Lidar inertial odometry and mapping. *Proceedings - IEEE International Conference on Robotics and Automation*, 2019-May:3144–3150, 2019.

[38] Milad Ramezani, Georgi Tinchev, Egor Iuganov, and Maurice Fallon. Online LiDAR-SLAM for Legged Robots with Robust Registration and Deep-Learned Loop Closure. In *IEEE International Conference on Robotics and Automation*, pages 4158–4164, 2020.

[39] Simona Nobili, Raluca Scona, Marco Caravagna, and Maurice Fallon. Overlap-based ICP tuning for robust localization of a humanoid robot. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 4721–4728, 2017.

[40] Hanspeter Pfister, Jeroen Van Baar, Matthias Zwicker, and Markus Gross. Surfels : Surface Elements as Rendering. *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, 2000.

[41] Chanoh Park, Soohwan Kim, Peyman Moghadam, Clinton Fookes, and Sridha Sridharan. Probabilistic Surfel Fusion for Dense LiDAR Mapping. *Proceedings - 2017 IEEE International Conference on Computer Vision Workshops, ICCVW 2017*, 2018-Janua:2418–2426, 2017.

[42] Jens Behley and Cyrill Stachniss. Efficient Surfel-Based SLAM using 3D Laser Range Data in Urban Environments. *Robotics: Science and Systems XIV*, 2018.

[43] Thomas Whelan, Renato F Salas-Moreno, Ben Glocker, Andrew J Davison, and Stefan Leutenegger. ElasticFusion: Real-time dense SLAM and light source estimation. *The International Journal of Robotics Research*, 35(14):1697–1716, 2015.

[44] John Novatnack and Ko Nishino. Scale-dependent/invariant local 3D shape descriptors for fully automatic registration of multiple sets of range images. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5304 LNCS(PART 3):440–453, 2008.

[45] Yu Zhong. Intrinsic shape signatures: A shape descriptor for 3D object recognition. *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops 2009*, pages 689–696, 2009.

[46] Federico Tombari, Samuele Salti, and Luigi DiStefano. Performance evaluation of 3D keypoint detectors. *International Journal of Computer Vision*, 102(1-3):198–220, 2013.

[47] Samuele Salti, Federico Tombari, and Luigi Di Stefano. SHOT: Unique signatures of histograms for surface and texture description. *Computer Vision and Image Understanding*, 125:251–264, 2014.

[48] Tixiao Shan and Brendan Englot. LeGO-LOAM: Lightweight and Ground-Optimized Lidar Odometry and Mapping on Variable Terrain. *IEEE International Conference on Intelligent Robots and Systems*, pages 4758–4765, 2018.

[49] Jean-Emmanuel Deschaud. IMLS-SLAM: scan-to-model matching based on 3D data. *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2480–2485, 2018.

[50] Frank Neuhaus, Tilman Koß, Robert Kohnen, and Dietrich Paulus. MC2SLAM: Real-Time Inertial Lidar Odometry Using Two-Scan Motion Compensation. *German Conference on Pattern Recognition*, 2018.

[51] Jacopo Serafin, Edwin Olson, and Giorgio Grisetti. Fast and robust 3D feature extraction from sparse point clouds. *IEEE International Conference on Intelligent Robots and Systems*, 2016-Novem:4105–4112, 2016.

[52] Ji Zhang and Sanjiv Singh. Visual-lidar Odometry and Mapping: Low-drift, Robust, and Fast. *IEEE International Conference on Robotics and Automation*, pages 2174–2181, 2015.

[53] Ji Zhang, Michael Kaess, and Sanjiv Singh. Real-time depth enhanced monocular odometry. *IEEE International Conference on Intelligent Robots and Systems*, (Iros): 4973–4980, 2014.

[54] Ji Zhang and Sanjiv Singh. Laservisualinertial odometry and mapping with high robustness and low drift. *Journal of Field Robotics*, 35(8):1242–1264, 2018.

[55] Yashar Balazadegan Sarvrood, Siavash Hosseinyalamdary, and Yang Gao. Visual-LiDAR Odometry Aided by Reduced IMU. *ISPRS International Journal of Geo-Information*, 5(1):3, 2016.

[56] Johannes Graeter, Alexander Wilczynski, and Martin Lauer. LIMO: Lidar-Monocular Visual Odometry. *IEEE International Conference on Intelligent Robots and Systems*, pages 7872–7879, 2018.

[57] Yuxing Xie, Jiaojiao Tian, and Xiao Xiang Zhu. Linking Points With Labels in 3D. *IEEE Geoscience and Remote Sensing Magazine*, (March), 2020.

[58] Renaud Dubé, Daniel Dugas, Elena Stumm, Juan Nieto, Roland Siegwart, and Cesar Cadena. SegMatch: Segment based loop-closure for 3D point clouds. In *IEEE International Conference on Robotics and Automation*, 2017.

[59] Leo Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

[60] Renaud Dubé, Andrei Cramariuc, Daniel Dugas, Juan Nieto, Roland Siegwart, and Cesar Cadena. SegMap: 3D Segment Mapping using Data-Driven Descriptors. In *Robotics: Science and Systems*, 2018.

[61] Sebastian Ratz, Marcin Dymczyk, Roland Siegwart, and Renaud Dubé. OneShot Global Localization: Instant LiDAR-Visual Pose Estimation. *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[62] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. NetVLAD: CNN Architecture for Weakly Supervised Place Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(6):1437–1451, 2018.

[63] Xieyuanli Chen, Andres Milioto, Emanuele Palazzolo, Philippe Giguere, Jens Behley, and Cyrill Stachniss. SuMa++: Efficient LiDAR-based Semantic SLAM. *IEEE International Conference on Intelligent Robots and Systems*, pages 4530–4537, 2019.

[64] Andres Milioto, Ignacio Vizzo, Jens Behley, and Cyrill Stachniss. RangeNet ++: Fast and Accurate LiDAR Semantic Segmentation. *IEEE International Conference on Intelligent Robots and Systems*, (i):4213–4220, 2019.

[65] Patrick Lichtsteiner, Christoph Posch, and Tobi Delbruck. A 128 x 128 120 dB 15 μs Latency Asynchronous Temporal Temporal Contrast Vision Sensor. *Solid State Circuit*, 43(2):566–576, 2008.

[66] Christian Brandli, Raphael Berner, Minhao Yang, Shih Chii Liu, and Tobi Delbruck. A 240 180 130 dB 3 μs latency global shutter spatiotemporal vision sensor. *IEEE Journal of Solid-State Circuits*, 49(10):2333–2341, 2014.

[67] David Weikersdorfer, Raoul Hoffmann, and Jörg Conradt. Simultaneous localization and mapping for event-based vision systems. *Computer Vision Systems*, pages 133–142, 2013.

[68] Hanme Kim, Stefan Leutenegger, and Andrew J. Davison. Real-Time 3D Reconstruction and 6-DoF Tracking with an Event Camera. *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 349–364, 2016.

[69] Jakob Engel, Thomas Sch, and Daniel Cremers. LSD-SLAM: Large-Scale Direct Monocular SLAM. *European Conference on Computer Vision*, pages 834–849, 2014.

[70] Henri Rebecq, Timo Horstschaefer, Guillermo Gallego, and Davide Scaramuzza. EVO: A Geometric Approach to Event-Based 6-DOF Parallel Tracking and Mapping in Real Time. *IEEE Robotics and Automation Letters*, 2(2):593–600, 2017.

[71] Alex Zihao Zhu, Nikolay Atanasov, and Kostas Daniilidis. Event-based feature tracking with probabilistic data association. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 4465–4470, 2017.

[72] Henri Rebecq, Timo Horstschaefer, and Davide Scaramuzza. Real-time Visual-Inertial Odometry for Event Cameras using Keyframe-based Nonlinear Optimization. *British Machince Vision Conference*, 2017.

[73] Antoni Rosinol Vidal, Henri Rebecq, Timo Horstschaefer, and Davide Scaramuzza. Ultimate SLAM? Combining Events, Images, and IMU for Robust Visual SLAM in HDR and High-Speed Scenarios. *IEEE Robotics and Automation Letters*, 3(2): 994–1001, 2018.

[74] Elias Mueggler, Guillermo Gallego, and Davide Scaramuzza. Continuous-time trajectory estimation for event-based vision sensors. *Robotics: Science and Systems*, 11, 2015.

[75] Elias Mueggler, Guillermo Gallego, Henri Rebecq, and Davide Scaramuzza. Continuous-Time Visual-Inertial Odometry for Event Cameras. *IEEE Transactions on Robotics*, 34(6):1425–1440, 2018.

[76] Valentina Vasco, Arren Glover, and Chiara Bartolozzi. Fast event-based Harris corner detection exploiting the advantages of event-driven cameras. *IEEE International Conference on Intelligent Robots and Systems*, 2016-Novem:4144–4149, 2016.

[77] C. Harris and M. Stephens. A Combined Corner and Edge Detector. *In Proc. of Fourth Alvey Vision Conference*, pages 147–152, 1988.

[78] Ignacio Alzugaray and Margarita Chli. Asynchronous Corner Detection and Tracking for Event Cameras in Real Time. *IEEE Robotics and Automation Letters*, 3(4):3177–3184, 2018.

[79] Elias Mueggler, Chiara Bartolozzi, and Davide Scaramuzza. Fast Event-based Corner Detection. *British Machince Vision Conference*, 1:1–11, 2017.

[80] Ryad Benosman, Charles Clercq, Xavier Lagorce, Sio Hoi Ieng, and Chiara Bartolozzi. Event-based visual flow. *IEEE Transactions on Neural Networks and Learning Systems*, 25(2):407–417, 2014.

[81] Xavier Lagorce, Garrick Orchard, Francesco Galluppi, Bertram E. Shi, and Ryad B. Benosman. HOTS: A Hierarchy of Event-Based Time-Surfaces for Pattern Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(7): 1346–1359, 2017.

[82] Ignacio Alzugaray and Margarita Chli. ACE: An efficient asynchronous corner tracker for event cameras. *Proceedings - 2018 International Conference on 3D Vision, 3DV 2018*, pages 653–661, 2018.

[83] Jacques Manderscheid, Amos Sironi, Nicolas Bourdis, Davide Migliore, and Vincent Lepetit. Speed invariant time surface for learning to detect corner points with event-based cameras. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June:10237–10246, 2019.

[84] Guillermo Gallego, Tobi Delbrruck, Garrick Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew Davison, Jörg Conradt, Kostas Daniilidis, and Davide Scaramuzza. Event-based Vision : A Survey. pages 1–30, 2019.

[85] Christian Brandli, Jonas Strubel, Susanne Keller, Davide Scaramuzza, and Tobi Delbruck. ELiSeD-An event-based line segment detector. *International Conference on Event-Based Control, Communication, and Signal Processing, EBCCSP 2016*, 2016.

[86] Lukas Everding and Jörg Conradt. Low-latency line tracking using event-based dynamic vision sensors. *Frontiers in Neurorobotics*, 12(FEB):1–13, 2018.

[87] Craig Glennie. Calibration and Kinematic Analysis of the Velodyne HDL-64E S2 Lidar Sensor. *Photogrammetric Engineering & Remote Sensing*, 78(4):339–347, 2012.

[88] Jesse Levinson and Sebastian Thrun. Unsupervised calibration for multi-beam lasers. *Springer Tracts in Advanced Robotics*, 79:179–193, 2014.

[89] Zachary Taylor and Juan Nieto. Motion-based calibration of multimodal sensor arrays. *IEEE International Conference on Robotics and Automation*, pages 4843–4850, 2015.

[90] João Alves, Jorge Lobo, and Jorge Dias. Camera-inertial sensor modelling and alignment for visual navigation. *International Conference on Advanced Robotics*, 5 (3):103–111, 2003.

[91] J. Lobo and J. Dias. Relative pose calibration between visual and inertial sensors. *The International Journal of Robotics Research*, 26:561–575, 2007.

[92] Jonathan Kelly and Gaurav S. Sukhatme. Fast Relative Pose Calibration for Visual and Inertial Sensors. *Springer Tracts in Advanced Robotics*, 54:515–524, 2009.

[93] Paul Furgale, Joern Rehder, and Roland Siegwart. Unified temporal and spatial calibration for multi-sensor systems. *IEEE International Conference on Intelligent Robots and Systems*, pages 1280–1286, 2013.

[94] Joern Rehder and Roland Siegwart. Camera IMU calibration. *IEEE Sensors Journal*, 17(11):1–2, 2017.

[95] Qilong Zhang and Robert Pless. Extrinsic Calibration of a Camera and Laser Range Finder (improves camera calibration). *IEEE International Conference on Intelligent Robots and Systems*, 3:2301–2306, 2004.

[96] Abdallah Kassir and Thierry Peynot. Reliable Automatic Camera-Laser Calibration. *Australasian Conference on Robotics and Automation*, 2010.

[97] Shinpei Kato, Eijiro Takeuchi, Yoshio Ishiguro, Yoshiki Ninomiya, Kazuya Takeda, and Tsuyoshi Hamada. An Open Approach to Autonomous Vehicles. *IEEE Micro*, 35(6):60–69, 2015.

[98] Sungdae Sim, Juil Sock, and Kiho Kwak. Indirect correspondence-based robust extrinsic calibration of LiDAR and camera. *Sensors (Switzerland)*, 16(6), 2016.

[99] Yoonsu Park, Seokmin Yun, Chee Sun Won, Kyungeun Cho, Kyhyun Um, and Sungdae Sim. Calibration between color camera and 3D LIDAR instruments with a polygonal planar board. *Sensors (Switzerland)*, 14(3):5333–5353, 2014.

[100] Martin Velas, Michal Spanel, Zdeněk Materna, and Adam Herout. Calibration of RGB camera with velodyne LiDAR. *WSCG 2014 Communication Papers Proceedings*, pages 135–144, 2014.

[101] Zachary Taylor and Juan Nieto. Parameterless automatic extrinsic calibration of vehicle mounted lidar-camera systems. *International Conference on Robotics and Automation: Long Term Autonomy Workshop*, (October):3–6, 2014.

[102] Juan Castorena, Ulugbek S Kamilov, and Petros T Boufounos. Autocalibration of LIDAR and optical cameras via edge alignement. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2862–2866, 2016.

[103] Gaurav Pandey, James R Mcbride, Silvio Savarese, and Ryan M Eustice. Automatic targetless extrinsic calibration of a 3D lidar and camera by maximizing mutual information. *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, pages 2053–2059, 2012.

[104] Joern Rehder, Paul Beardsley, Roland Siegwart, and Paul Furgale. Spatio-temporal laser to visual/inertial calibration with applications to hand-held, large scale scanning. *IEEE International Conference on Intelligent Robots and Systems*, (Iros): 459–465, 2014.

[105] Jonathan Brookshire and Seth Teller. Automatic calibration of multiple coplanar sensors. *Proceedings of Robotics: Science and Systems*, pages 33–40, 2011.

[106] Zachary Taylor and Juan Nieto. *Automatic Markerless Calibration of Multi-Modal Sensor Arrays*. PhD thesis, University of Sydney, 2015.

[107] Cedric Le Gentil, Teresa Vidal-Calleja, and Shoudong Huang. Gaussian Process Preintegration for Inertial-Aided State Estimation. *IEEE Robotics and Automation Letters*, 5(2):2108–2114, 2020.

[108] Michael Boyle. The Integration of Angular Velocity. *Advances in Applied Clifford Algebras*, 27(3):2345–2374, 2017.

[109] Christian Forster, Luca Carlone, Frank Dellaert, and Davide Scaramuzza. Supplementary Material to : IMU Preintegration on Manifold for E cient Visual-Inertial Maximum-a-Posteriori Estimation IMU Preintegration : Noise Propagation and Bias Updates. *Technical Report GT-IRIM-CP&R-2015-001*, 2015.

[110] Vasiliy M. Tereshkov. A Simple Observer for Gyro and Accelerometer Biases in Land Navigation Systems. *Journal of Navigation*, 68(04):635–645, 2015.

[111] Cedric Le Gentil, Teresa Vidal-Calleja, and Shoudong Huang. IN2LAAMA: INertial Lidar Localisation Autocalibration And MApping. *IEEE Transactions on Robotics*, 2021.

[112] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, sep 1975.

[113] Karen Kafadar. Efficiency of the Biweight As a Robust Estimator of Location. *Journal of Research of the National Bureau of Standards (United States)*, 88(2): 105–116, 1983.

[114] Vasiliy M. Tereshkov. An Intuitive Approach to Inertial Sensor Bias Estimation. *International Journal of Navigation and Observation*, 2013:1–6, 2013.

[115] Zhengshi Yu and John L. Crassidis. Accelerometer Bias Calibration Using Attitude and Angular Velocity Information. *Journal of Guidance, Control, and Dynamics*, 39 (4):741–753, 2016.

[116] Shuang Du, Wei Sun, and Yang Gao. Improving observability of an inertial system by rotary motions of an IMU. *Sensors (Switzerland)*, 17(4):1–20, 2017.

[117] Yulin Yang, Patrick Geneva, Kevin Eckenhoff, and Guoquan Huang. Degenerate Motion Analysis for Aided INS with Online Spatial and Temporal Sensor Calibration. *IEEE Robotics and Automation Letters*, 4(2):2070–2077, 2019.

[118] Michael Bosse, Paul Newman, John Leonard, Martin Soika, Wendelin Feiten, and Seth Teller. An Atlas framework for scalable mapping. *Proceedings - IEEE International Conference on Robotics and Automation*, 2:1899–1906, 2003.

[119] Zhenfei Yang and Shaojie Shen. Monocular visual-inertial state estimation with online initialization and camera-IMU extrinsic calibration. *IEEE Transactions on Automation Science and Engineering*, 14(1):39–51, 2017.

[120] Cedric Le Gentil, Florian Tschopp, Ignacio Alzugaray, Teresa Vidal-calleja, Roland Siegwart, and Juan Nieto. IDOL : A Framework for IMU-DVS Odometry using Lines. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5863–5870, 2020.

[121] Elias Mueggler, Henri Rebecq, Guillermo Gallego, Tobi Delbruck, and Davide Scaramuzza. The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and SLAM. *International Journal of Robotics Research*, 36 (2):142–149, 2017.

[122] Cedric Scheerlinck, Henri Rebecq, Daniel Gehrig, Nick Barnes, Robert E. Mahony, and Davide Scaramuzza. Fast image reconstruction with an event camera. *Proceedings - 2020 IEEE Winter Conference on Applications of Computer Vision, WACV 2020*, pages 156–163, 2020.

[123] Henri Rebecq, Rene Ranftl, Vladlen Koltun, and Davide Scaramuzza. High Speed and High Dynamic Range Video with an Event Camera. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, XX(XX):1–1, 2019.

[124] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the Properties of Neural Machine Translation: EncoderDecoder Approaches. In *Proceedings ofSSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, 2014.

[125] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention MICCAI*, volume 9351, pages 234–241. 2015.

[126] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wang-chun Woo. Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. In *Advances in Neural Information Processing Systems 28*, pages 802–810, 2015.

[127] Konstantinos Zampogiannis, Cornelia Fermüller, and Yiannis Aloimonos. Cilantro: A lean, versatile, and efficient library for point cloud data processing. *MM 2018 - Proceedings of the 2018 ACM Multimedia Conference*, pages 1364–1367, 2018.

[128] Michael Bloesch, Sammy Omari, Marco Hutter, and Roland Siegwart. Robust visual inertial odometry using a direct EKF-based approach. *IEEE International Conference on Intelligent Robots and Systems*, 2015-Decem:298–304, 2015.

[129] Thomas Schneider, Marcin Dymczyk, Marius Fehr, Kevin Egger, Simon Lynen, Igor Gilitschenski, and Roland Siegwart. Maplab: An Open Framework for Research in Visual-Inertial Mapping and Localization. *IEEE Robotics and Automation Letters*, 3(3):1418–1425, 2018.

[130] Stefan Leutenegger, Margarita Chli, and Roland Y. Siegwart. BRISK: Binary Robust invariant scalable keypoints. *Proceedings of the IEEE International Conference on Computer Vision*, pages 2548–2555, 2011.

[131] Zichao Zhang and Davide Scaramuzza. A Tutorial on Quantitative Trajectory Evaluation for Visual(-Inertial) Odometry. *IEEE International Conference on Intelligent Robots and Systems*, pages 7244–7251, 2018.

[132] Benny Dai, Cedric Le Gentil, and Teresa Vidal-Calleja. Connecting the dots for real-time LiDAR-based object detection with YOLO. *Australasian Conference on Robotics and Automation, ACRA*, 2018-Decem, 2018.

[133] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You Only Look Once : Unified , Real-Time Object Detection. *Conference on Computer Vision and Pattern Recognition*, 2016.

[134] Yin Zhou and Oncel Tuzel. VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 4490–4499, 2018.

[135] Dorian Galvez-Lopez and Juan D. Tardos. Bags of Binary Words for Fast Place Recognition in Image Sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197, 2012.

[136] Raul Mur-Artal, J. M.M. Montiel, and Juan D. Tardos. ORB-SLAM: A Versatile and Accurate Monocular SLAM System. *IEEE Transactions on Robotics*, 31(5): 1147–1163, 2015.

[137] Martin Magnusson, Henrik Andreasson, a. Nuchter, and a.J. Lilienthal. Appearance-based loop detection from 3D laser data using the normal distributions transform. *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, (2): 23–28, 2009.

[138] Naveed Muhammad and Simon Lacroix. Loop closure detection using small-sized signatures from 3D LIDAR data. *9th IEEE International Symposium on Safety, Security, and Rescue Robotics, SSRR 2011*, pages 333–338, 2011.

[139] Karl Granström, Thomas Schön, Karl Granstr, and Thomas B Sch. Linköping University Post Print Learning to Close the Loop from 3D Point Clouds Learning to Close the Loop from 3D Point Clouds. *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 2089–2095, 2010.

[140] Yoav Freund and Robert E. Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55:119–139, 1997.

[141] Bastian Steder, Michael Ruhnke, Slawomir Grzonda, and Wolfram Burgard. Place recognition in 3D scans using a combination of bag of words and point feature based relative pose estimation. *IEEE International Conference on Intelligent Robots and Systems*, 2011.

[142] Giseop Kim and Ayoung Kim. Scan Context: Egocentric Spatial Descriptor for Place Recognition Within 3D Point Cloud Map. *IEEE International Conference on Intelligent Robots and Systems*, pages 4802–4809, 2018.

[143] Cedric Le Gentil, Mallikarjuna Vayugundla, Riccardo Giubilato, Wolfgang St, Teresa Vidal-calleja, and Rudolph Triebel. Gaussian Process Gradient Maps for Loop-Closure Detection in Unstructured Planetary Environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1895–1902, 2020.

[144] Rik Bahnemann, Michael Burri, Enric Galceran, Roland Siegwart, and Juan Nieto. Sampling-based motion planning for active multirotor system identification. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 3931–3938, 2017.

[145] Thomas Schneider, Mingyang Li, Cesar Cadena, Juan Nieto, and Roland Siegwart. Observability-Aware Self-Calibration of Visual and Inertial Sensors for Ego-Motion Estimation. *IEEE Sensors Journal*, 19(10):3846–3860, 2019.

[146] Mitchell Usayiwevu, Cedric Le Gentil, Jasprabhjit Mehami, Chanyeol Yoo, Robert Fitch, and Teresa Vidal-calleja. Information Driven Self-Calibration for Lidar-Inertial Systems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9961–9967, 2020.

[147] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An Open Urban Driving Simulator. (CoRL):1–16, 2017.

[148] Alex Zihao Zhu, Dinesh Thakur, Tolga Özaslan, Bernd Pfrommer, Vijay Kumar, and Kostas Daniilidis. The Multivehicle Stereo Event Camera Dataset: An Event Camera Dataset for 3D Perception. *IEEE Robotics and Automation Letters*, 3(3): 2032–2039, 2018.