Robust Self-Calibration and Mapping for Long Term Autonomy

by

Fernando Nobre

M.Sc., University of Colorado Boulder, 2016

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

2018

This thesis entitled:
Robust Self-Calibration and Mapping for Long Term Autonomy
written by Fernando Nobre
has been approved for the Department of Computer Science

Professor Christoffer Heckman

Professor Nisar Ahmed

Date

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.

Nobre, Fernando (Ph.D., Computer Science)

Robust Self-Calibration and Mapping for Long Term Autonomy

Thesis directed by Professor Christoffer Heckman

Robust long term autonomy represents one of the most important targets for advancing robotic applications in the next 10 to 15 years, particularly because of the the advances in the so-called Classical Age of Simultaneous Localization and Mapping (SLAM) which place us in the age of *Robust-Perception*. Creating algorithms which allow robots to operate for years, unsupervised, in any environment is key step in the direction of true autonomy. This thesis presents a suite of algorithms to help enable robust long term autonomy, specifically robustness to a robot's calibration parameters (internal knowledge the robot must possess in order operate) and to the environment the robot is situated in. Starting from the fundamentals of SLAM, the now *de facto* formulation is presented as a segue into self-calibration - the task of estimating calibration parameters such as the camera position on the robot. The following extensions are then developed: (i) an approach to treat slowly varying quantities, such as the position of a sensor drifting over years of operation, (ii) an algorithm which allows a robot to learn what movements it needs to perform in order to know its calibration parameters - using a reinforcement learning framework for self-calibration and (iii) all the insight from previous research is used to create a real-time self-calibration system which is capable of dealing with drift, unobservable parameters - for example when the robot is constrained to planar movement - and an information theoretic based segment selection mechanism which only choses "informative" segments of the trajectory in order to reduce computation time. However robustness is not only in regards to internal parameters such as a robot's sensor position - the environment the robot operates in is dynamic - dealing with that environment is the final contribution, where an online probabilistic approximate joint feature persistence model is presented to determine which parts of the world are changing.

Dedication

To my family.[1]

[1] To my parents, for years of unquestioning love and support; my sisters, Patricia and Carolina for being pretty much the best, coolest sisters ever; my niece Julia, for being part of the reason this work motivates me; and my love, Paige, for our life together and the great adventures ahead.

Acknowledgements

Field robotics is collaborative work, and this thesis would not have been possible without the help and support of many people and organizations. The agile platform and hardware deployment would not have been possible without all of the following: electrical and general mechanical knowledge from Sina Aghli; hardware support and technical advice regarding low-level serial actuation from Steve Mcguire; great video recording work from John Stechschulte, Mike Kasper and excellent piloting from Steve Mcguire who successfully avoided running over a squirrel with the robot.

I have been lucky to have excellent mathematical tutors at the Department of Computer Science including Gabe Sibley who introduced me to the field of robotics and state estimation; Christoffer Heckman who contributed to the work throughout the entire thesis and provided excellent mathematical grounding. The work in Chapter 6 would not have been possible without the fruitful discussions with the folks at Toyota Research Institute, specifically Jeff Walls, Paul Ozog and Ryan Wolcott;

I would also like to acknowledge the help of many of the members of the Autonomous Robotics and Perception Group, specifically Mike Kasper for his tireless patience in discussing ideas, his donation to many whiteboard hours and for being an understanding desk neighbor; Nima Keivan for his excellent ideas and extensive codebase; Cecilia Mauceri, Corin Sandford, Mahsa Ghafarianzadeh and Juan Falquez for engaging discussion and making these years a time I shall look back fondly upon.

Contents

Tables

Figures

Chapter 1

Introduction

Simultaneous localization and mapping (SLAM) is the problem of simultaneously estimating the state of the robot and a map of the environment. Much progress has been made under the banner of SLAM. However, for long-term autonomous operations in real-world environments, localization and mapping must be viewed as tasks that continue over the lifetime of robot. Real environments change in their shape and appearance over time, both gradually and acutely. Figure 1.1 shows a classic example of a long-term application. Recently, a number of exciting new approaches to handling environmental change to support lifelong localization and mapping have been proposed.

In its most simple form the state is the robot's position, consisting of its position and rotation with regards to some reference frame. The state can optionally be augmented with other quantities such as biases, velocity and calibration parameters. The map can take several representations depending on the aspects of interest, such as the position of landmarks, objects, a dense 3D mesh, etc. describing the environment in which the robot operates. The motivation behind building a map is twofold: A map is often required for other, higher-level robot tasks, such as path-planning or for providing an intuitive visualization for a human operator. Second, the map allows for limiting the error in the robot's state estimation: by re-visiting an area of the map the robot can "reset" its localization error (so called **loop closures**). The motivation behind estimating the robot's state is immediate as any higher-level action on the robot requires the knowledge of its current state. Therefore SLAM has application in any scenario where a prior map is not available and needs to be built.

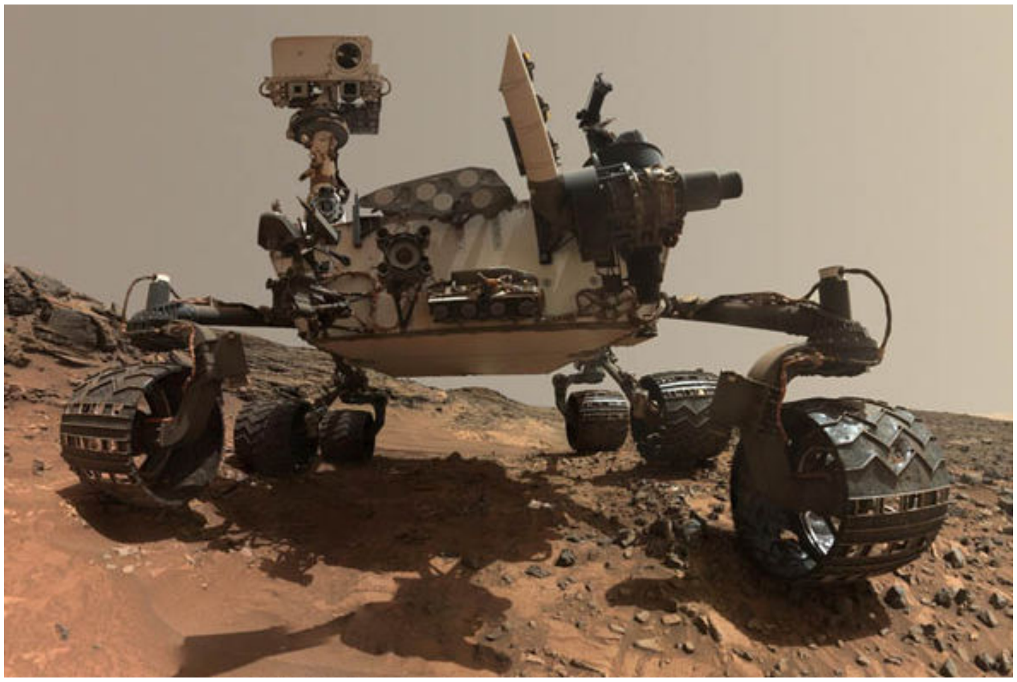


Figure 1.1: Example of a life-long robotic application: NASA Curiosity rover on Mars.

In other robotics applications, an **a priori** environment is known, such as a robot operating on a factory floor where a manually built map is provided, on a self-driving car operating on streets that have previously been mapped with laser scanners or even in outdoor environments where GPS measurements are available (the GPS measurements represent known locations in the map). In these scenarios the problem can be simplified to simply estimating the robot state. SLAM owes most of its popularity due to the flexibility of dealing with both scenarios: indoor environments where no pre-existing map exists, operating in an environment with a pre-built map and seamlessly switching between both scenarios.

Given the current state of SLAM, it begs the question **is SLAM solved?**. This is a question often asked in robotics communities [42]. The question is difficult to answer since SLAM is such a brad topic, a given robot/environment/performance combination. In general terms, the following aspects must be defined for any well posed answer to be formed:

- **platform**: available sensors, dynamics, computational resources;
- **environment**: 3D or planar, presence of static landmarks, amount of dynamic elements;
- **performance**: robot state estimation accuracy, accuracy for environment representation, success rate, latency, memory requirements.

For example, mapping in a planar environment with a platform equipped with wheel encoders and a laser scanner with sufficient accuracy and low failure rate can be considered largely solved. Vision-based SLAM with slowly-moving robots (domestic robots, mars rovers, etc) and visual-inertial odometry can be considered mature research fields. Pretty much ever other platform, environment, performance combination still deserves a large amount of research. Current algorithms fail when the motion of the robot/environment is overly challenging (e.g., fast robot dynamics, acutely dynamic environments) or when the performance requirements are very strict (e.g., high rate estimation for fast closed-loop control).

1.1 Life-Long SLAM

SLAM systems have matured to the point where we are asking, and most importantly, answering questions such as "What is long-term autonomy?", "Do we need semi-autonomy to get us there?", "What novel map representations are required?", "Do we need redundant sensing for robustness?", and "What kind of long-duration field experiments are required?". In this section we will build on the basic optimization-based SLAM formulation presented in Section **??** and analyze some aspects related to life-long SLAM; where current research has gotten us, what are the major shortcomings and the unexplored areas that need future work. These aspects can be broadly divided into two main categories: (i) **Robustness**, which addresses calibration and the many failure modes of SLAM when applied to long-term scenarios and (ii) **Scalability** which looks at dealing with the growing state and environment footprints in a resource constrained platform.

A SLAM system can be fragile in two major aspects: algorithmic or hardware-related. The former class includes failure modes inherent to the existing SLAM algorithms (e.g., difficulty to handle extremely dynamic environments). The latter includes failures due to sensor degradation over time. In order to achieve long-term operation, explicitly addressing these failure modes is crucial, where simplified assumptions such as static environment or full reliance on on-board sensors no longer hold. We will omit the (large) class of software-related failures, while briefly mentioning that system integration and testing are non-trivial and key aspects of any SLAM system.

1.1.1 Algorithmic Failures

Data association is one of the main sources of algorithmic failures. As described in Section 2.3 data association matches each measurement to a corresponding portion of the state vector. In feature-based visual SLAM, it associates each visual feature (or **key-point**) to a specific 3D landmark. Perceptual aliasing, the phenomenon in which different sensory inputs lead to the same sensor signature makes this problem particularly hard. This may cause incorrect data-associations (false positives) which, if not handled correctly, results in incorrect estimates from the back-end.

The presence of unmodeled dynamics in the environment makes the situation worse, with things such as short and long-term seasonal changes, which deceive the data-association module. A common assumption in most SLAM approaches is that the world remains unchanged as the robot moves through it (static landmark assumption). This **static world assumption** holds true for short mapping runs in small scale environments, as long as there is a limited number of short term dynamics (e.g., people/objects moving about the scene). We are primarily concerned with mapping over longer time scales, in arbitrarily large environments, where change is inevitable. Variations from day to night, seasonal changes, such as foliage, and even change in the structure of the environment, as new buildings rise and old buildings are demolished, all directly affect the performance of SLAM systems. Another aspect of robustness is that of doing SLAM in adverse environments, such as underwater [10, 35, 36, 71] where the limited visibility, constantly changing conditions and impossibility of using sensors such as laser range finders make the problem challenging. The most immediate robustness issue to tackle is data association. It can have devastating effects on the optimization and can be addressed in the front-end and/or in the back-end of a SLAM system. Traditionally the front-end has been entrusted with establishing correct data association. The easiest scenario is short-term data association: if the sampling rate of the sensor is relatively fast to the dynamics of the robot, tracking features that correspond to the same 3D landmark is considered a solved problem. For example, if we wish to track a 3D point across consecutive images, assuming the framerate is sufficiently high, standard approaches based on descriptor matching and optical flow [133] ensure consistent and reliable tracking. This short-term visual tracking is the basis of **visual odometry**. On the other hand, long-term data association is more challenging and involves both loop closure detection and validation. For loop closure detection on the front-end, the naive brute-force approach which detects features in the current measurement (e.g., image) and tries to match them against all previously detected features becomes impractical. Bag-of-words [137] is a popular solution to this problem, by quantizing the feature-space and allowing more efficient searches. Furthermore Bag-of-words can be arranged in to a hierarchical vocabulary tree [112] that enable quick lookup in large-scale datasets. Bag-of-words-based techniques such as [25, 44]

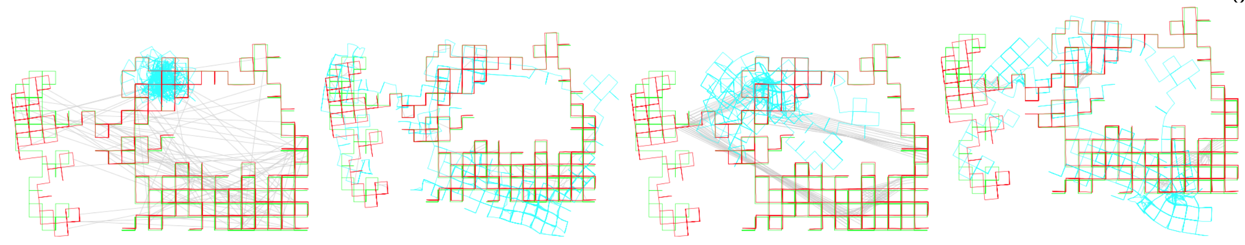


Figure 1.2: Example of incorrect data association provided by the front end; the back end has to be robust to theses incorrect associations. Standard "robust" back ends such as g2o using the Huber norm fail to converge (blue), the proposed robust switchable constraint back end [141] (red) discards incorrect loop closures. Figure courtesy of Niko Sünderhauf and Peter Protzel [141].

have shown reliable performance on the task of loop closure detection. However these approaches have the significant drawback of not handling severe illumination changes, since visual words can no longer be matched. This shortcoming has led to the development of methods that explicitly account for such variations by matching sequences [102], gathering different possible visual appearances into a unified representation [21], or using spacial and appearance information [57]. Lowry **et al.** provides a detailed survey on visual place recognition [91]. There are also approaches for detecting loop closures in other sensor modalities, such as laser scanners: Tipaldi **et al.** [142] proposes FLIRT features in 2D laser scans. Loop closure validation consists of additional geometric validation steps to ascertain the quality of the loop closure detection. In the case of vision-based loop closures, RANSAC is commonly used for geometric verification and outlier rejection, [41] and references therein provide a thorough coverage.

Despite all the effort towards detecting and validating loop closures on the front-end, the presence of perceptual aliasing makes it unavoidable that wrong loop closures are fed to the back-end. These incorrect associations can severely corrupt the quality of the estimate [141]. Due to this, a recent line of research proposes techniques [3, 18, 81, 117, 141] to make the SLAM back-end resilient against spurious measurements. Figure 1.2 shows an example of the proposed solution in [141], where the back-end tries to "optimize out" incorrect data associations provided by the front end. These methods all reason on the validity of the loop closure constraints by looking at the residual error induced by the constraints during the optimization. Constraints that increase the residual by an unreasonable amount are considered outlines and are 'disabled'. Other methods [118, 128] try to detect incorrect loop closures before any optimization is run, by identifying loop closures that are not supported by the odometry. In dynamic environments the challenge is twofold. First, the system has to detect, discard or track changes. Mainstream approaches attempt to discard dynamic portions of the scene [109], other works incorporate the dynamic elements as part of the model [125, 147, 146]. The second challenge involves modeling permanent and semi-permanent changes to the environment, and update the map accordingly. Current approaches that deal with dynamic environments either maintain multiple, time-dependent maps of the same location [27, 74] or use a single representation, parameterized by a time-varying parameter [76, 131].

1.1.2 Calibration

An issue every roboticist deals with routinely is sensor calibration. In order to be able to use sensor data in the back-end (Section 2.2) each sensor needs to be precisely calibrated. For vision-based sensors for example, knowing the camera lens distortion parameters, focal length and central point (necessary for a pinhole projection model) are essential for correct feature extraction/tracking. An inertial measurement unit (IMU) is even more complicated, with time-varying biases on each of the accelerometer and gyroscope axis. Furthermore, if multiple sensors are used, as is typical in most SLAM applications (e.g. the visual-inertial navigation system is a common example of sensor fusion [94]) then sensor extrinsic calibration parameters are also needed (e.g. the rigid-body SE(3) relative pose between sensors). There has been a considerable amount of effort dedicated to sensor intrinsic and extrinsic self-calibration [61, 99, 93, 97, 24, 32, 150, 115, 66, 68]. Both [56] and [121] considered self-calibration in a batch setting with various tailoring to different intrinsic parameters. [39] presented a method to calibrate the varying intrinsics of a pinhole camera in a batch setting, given the rotation of the camera was known. A solution was also offered to align the rotation sensor and camera data in time. Many current techniques for vision-aided inertial navigation use filtering approaches [61, 69, 106] or a smoothing formulation. In either case the estimation is made constant-time by rolling past information into a prior distribution. Filtering methods present the significant drawback of introducing inconsistencies due to linearization errors of past measurements which cannot be corrected post hoc, particularly troublesome for non-linear camera models. Some recent work has tackled these inconsistencies; see, e.g. [87, 55, 23, 88]. The state-of-the-art includes methods to estimate poses and landmarks along with calibration parameters, but these approaches do not output the marginals for the calibration parameters, which are desirable for long-term autonomy applications.

Building on these works, simultaneous solutions to the SLAM and self-calibration problem have been proposed but generally all online solutions assume constant calibration parameters. [22] proposed a method to recursively estimate camera and landmark 3D parameters as well as the intrinsic parameters of a nonlinear camera model in an online framework. [89] also developed a filtering solution to estimate both the camera pose and also intrinsics and extrinsics for a non-linear camera model with rolling shutter and a commercial grade IMU in an online framework, but that approach does not output covariances in an MLE sense. Nobre **et al.** [115] extended the work presented in [68] to propose an extensible self-calibrating and change-detecting intrinsic and extrinsic pipeline that operates online and in real-time. The key insight is collecting informative segments of the trajectory for estimating the calibration parameters, and using a statistical test to ascertain if the calibration parameters have changed.

1.2 Outline

This thesis will first define the standard "architecture" for SLAM (Chapter 2) based on an optimization-based maximum-a-posterior formulation which is broken down into a **back-end** (Section 2.2) and **front-end** (Section 2.3). A gentle introduction to Lie Groups, and how it pertains to representing rotations is also provided (Appendix A). Chapter 3 introduces self-calibration and presents a constant-time algorithm for both estimating a robot's calibration parameters as well as detecting changes in those parameters and adjusting accordingly. Chapter 4 extends Chapter 3 to deal with the case when a robot is operating for extended periods of time and the changes in parameters are gradual and harder to detect. In Chapter 5 an active approach is taken, where the robot explores its action space and *learns* what motions it needs to perform in order to calibrate itself. Chapter 6 shifts the perspective to the world in which the robot operates, developing a novel formulation on environment feature persistence which leverages joint information from correlated features to detect small changes in the environment. Chapter 7 takes lessons learned from all the previous chapters and builds upon them to develop a low time complexity robust self-calibrating algorithm which is experimentally validated in long term autonomy applications. Finally, conclusions and future work directions are discussed in Chapter 8.

1.3 Publications

Some of the contributions described in this dissertation have first appeared as the following publications:

- **Chapter 3**: "Multi-Sensor SLAM with Online Self-Calibration and Change Detection", presented at the International Symposium on Experimental Robotics (ISER), 2016 [115]

- **Chapter 4**: "Drift-Correcting Self-Calibration for Visual-Inertial SLAM", presented at the International Conference on Robotics and Automation (ICRA), 2017 [116]

- **Chapter 5**: "Reinforcement Learning for Assisted Visual-Inertial Robotic Calibration", presented at the International Symposium on Robotics Research (ISRR), 2017 [113] and invited submission to the International Journal on Robotics Research (IJRR) special edition.

- **Chapter 6**: "Online Probabilistic Change Detection in Feature-Based Maps", presented at the International Conference on Robotics and Automation (ICRA), 2018 [114] and US Patent Application Serial Number IP-A-2602.

- **Chapter 7**: "FastCal: Robust Online Self-Calibration for Robotic Systems", under submission to the International Symposium on Experimental Robotics (ISER), 2018

Chapter 2

State Parameterization for SLAM

The genesis of the probabilistic SLAM problem dates back to the 1986 IEEE Robotics and Automation Conference held in San Francisco, California. This sparked the development of the basic probabilistic SLAM formulations, such as the Extended Kalman Filter, Rao-Blackwellized Particle Filters and the maximum likelihood estimation; also it outlined the challenges associated with data association and robustness. This period can be roughly set as starting in 1986 and ending in 2004. A thorough overview of these first 20 years of SLAM can be found in the surveys by Durrant-Whyte and Bailey in [33, 9]. More recently, a partial survey conducted by Dissanayake **et al.** in [31] covers a period refereed to as the **algorithmic analysis age** where the fundamental properties of SLAM were studied, including observability, convergence, consistency and non-linearity. Also of note in this period is the discovery and use of the sparsity structure in SLAM for the creation of efficient solvers that use the Schur-Complement to exploit the sparse nature of the linearized SLAM system, which will be covered in section 2.2. Several open-source SLAM libraries were created in this period.

A large collection of surveys on SLAM have been published, with the later ones focusing on specific sub-fields such as Multi-Robot SLAM and visual place recognition. Table 2.1 contains the main SLAM surveys to date. The large amount of research on SLAM in the past 30 years is not surprising if one thinks about the breath of manifolds SLAM encompasses; at a lower level (called the **front end**, Section 2.3) SLAM intersects other research fields such as computer vision and signal processing. On a higher level (which will later be referred to as the **back end**, Section 2.2) SLAM is an interesting combination of statistics, graph theory, geometry and probabilistic estimation. On top of these layers there are non-negligible practical aspects ranging from sensor modeling to system integration.

Table 2.1: SLAM Surveys

Year	**Topic**	**Reference**
2006	Probabilistic approaches	Durrant-Whyte and Bailey [20, 6]
2008	Filtering methods	Aulinas **et. al.** [5]
2008	Visual SLAM	Neira **et. al.** [63]
2008	Visual Navigation	Bonin-Font **et. al.** [8]
2008	Loop Closures	Mahon **et. al.** [57]
2011	Observability, Consistency, Convergence	Dissanayake **et. al.** [18]
2011	SLAM Back-End	Grisetti **et. al.** [28]
2012	Visual Odometry	Scaramuzza and Fraundofer [81, 80]
2012	Driverless Cars	Ros **et. al.** [74]
2012	Visual SLAM	Fuentes-Pacheco **et. al.** [26]
2015	Rotation Estimation	Carlone **et. al.** [10]
2016	Multi-Robot SLAM	Saeedi **et. al.** [77]
2016	Visual Place recognition	Lowry **et. al.** [53]
2016	Convergence, Observability, Robustness	Huang **et. al.** [35]

2.1 Anatomy of a SLAM system

There are three prevalent formulations of the Simultaneous Localization and Mapping Problem (SLAM) problem: Probabilistic, Extended Kalman Filter (EKF) or the Particle Filter formulation such as the Rao-Blackwellized Filter. We will take the Probabilistic approach since it is useful for viewing SLAM from the traditional Statistical Point Estimation pespective as that reveals the underlying problem structure. This approach makes evident the least squares minimization prin-

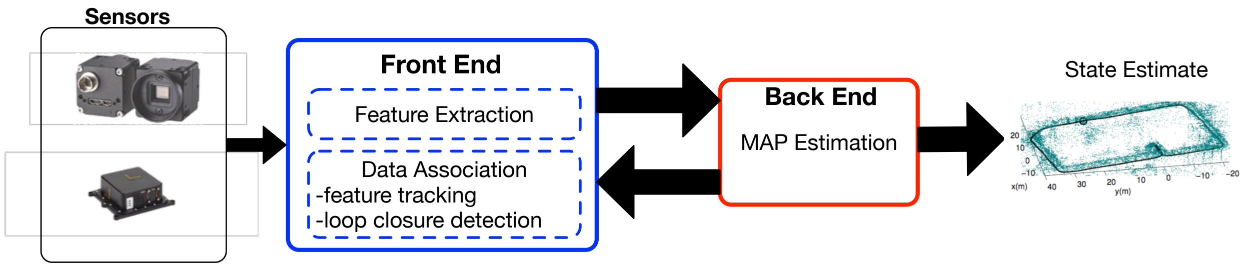


Figure 2.1: Typical front-end and back-end division in a SLAM algorithm.

underlying problem structure. This approach makes evident the least squares minimization principle, which is less evident in the EKF (or Recursive Bayesian Estimation) formulation. Also, this approach clearly shows the underlying probability density functions, which highlight the Gaussian probabilistic nature of SLAM - which means that SLAM is simply tracking a normal distrubution though the state space; a state space that is dynamic in its number of elements as we remove parameters though the probabilistic operation of marginalization or add parameters via conditioning. Finally, the statistical point estimation approach directly leverages a large body of knowledge on the convergence of least squares estimation, which is not so readily available in the recursive non-linear (EKF) perspective.

Modern SLAM systems can be divided into two main parts: **front-end** and **back-end.** Figure 2.1 shows a high-level division.

2.2 Maximum-a-Posteriori SLAM back-end

While there are different formulations for the back-end, such as the Extended Kalman Filter or Particle Filters, the **de-facto** standard is the Maximum-A-Posteriori (MAP) estimation, which can be traced back to 1997 in Lu and Milios's work [92] and Gutmann and Konolige [49]. In the following 16 years multiple approaches have improved the efficiency and robustness of the underlying optimization problem. In all these approaches, SLAM is formulated as a maximum-a-posteriori estimation problem and most use the formalism of **factor graphs** [78] to reason about the interdependence among variables.

Assuming we have a set of random variables we wish to estimate, denoted by $\mathcal{X}$; in SLAM the variable $\mathcal{X}$ usually includes the robot trajectory (as a discrete set of 6-DOF poses) and the position of landmarks in the environment. The state can be augmented with biases, gravity, calibration parameters, etc. Given a set of measurements $\mathcal{Z} = \{z_k : k = 1, ..., m\}$ such that each measurement can be expressed as a function of $\mathcal{X}$; $z_k = h_k(\mathcal{X}_k) + \epsilon_k$ where $\mathcal{X}_k \subseteq \mathcal{X}$ is a subset of the full state vector, $h_k(\cdot)$ is a known function (**measurement** or **observation** model) and ϵ_k is random measurement noise, usually assumed to be zero-mean Gaussian. We wish to estimate $\mathcal{X}$ by computing the optimal assignment of variables $\mathcal{X}^*$ that attains the maximum of the posterior $\mathbb{P}(\mathcal{X}|\mathcal{Z})$, which can be interpreted as the **belief** over $\mathcal{X}$, given all the measurements:

$$\mathcal{X}^* \doteq \arg\max_{\mathcal{X}} \mathbb{P}(\mathcal{X}|\mathcal{Z}) = \arg\max_{\mathcal{X}} \mathbb{P}(\mathcal{Z}|\mathcal{X})\mathbb{P}(\mathcal{X}) \tag{2.1}$$

Which follows from Bayes theorem. $\mathbb{P}(\mathcal{Z}|\mathcal{X})$ is the **likelihood** of the measurements $\mathcal{Z}$ given the current state assignment $\mathcal{X}$, and $\mathbb{P}(\mathcal{X})$ is the prior probability distribution over $\mathcal{X}$. The prior encodes any information known about the state vector before measurements are taken; in case no prior is available, $\mathbb{P}(\mathcal{X})$ reduces to a uniform distribution (constant) which does not affect the maximization problem and can be dropped from the optimization. In these cases the MAP estimation is reduced to a **maximum likelihood estimation**. The common assumption is that the measurements $\mathcal{Z} = \{z_k : k = 1, ..., m\}$ are independent (i.e., the noise affecting the measurements are not correlated), problem (2.1) can be factorized into:

$$\mathcal{X}^* = \arg\max_{\mathcal{X}} \mathbb{P}(\mathcal{X}) \prod_{k=1}^{m} \mathbb{P}(z_k|\mathcal{X}) = \arg\max_{\mathcal{X}} \mathbb{P}(\mathcal{X}) \prod_{k=1}^{m} \mathbb{P}(z_k|\mathcal{X}_k) \tag{2.2}$$

Where z_k now only depends on the subset $\mathcal{X}_k$ of the state vector. This problem can be interpreted as inference over a factor graph, where the nodes in the graph correspond to the variables in the state vector $\mathcal{X}$ and the terms $\mathbb{P}(z_k|\mathcal{X}_k)$ and $\mathbb{P}(\mathcal{X})$ are called **factors** that encode probabilistic constraints over a subset of nodes. Thus a factor-graph is a graphical model which encodes the dependence between the **k**-th factor (with its measurement z_k) and the corresponding variables $\mathcal{X}_k$. An immediate advantage of this model is that it enables an intuitive visualization of the problem. Figure 2.2 shows an example factor graph underlying a simple SLAM problem. The figure shows the robot poses, landmarks and camera calibration parameters (the variables that compose the robot state) and the factors that impose constraints between these variables. A less immediate advantage of factor graphs is generality: complex inference problems can be modeled, with heterogeneous variables and factors, with arbitrary interconnections. For example, completely different sensor inputs, such as camera data and inertial measurement unit readings can be seamlessly integrated into a factor graph containing two nodes. A node can contain variables embedded in different manifolds, such as an euclidean translation, a SO(3) rotation and calibration parameters. Also of note is that the connectivity of the factor graph is directly correlated with the sparsity of the resulting SLAM problem, as discussed below.

We now will write Eq. (2.2) in a more explicit and tractable form, which in turn will allow us to derive the basic MAP SLAM algorithm. Assume that the measurement noise ϵ_k is a zero-mean Gaussian noise with information matrix Π_k. Then the measurement likelihood in Eq. (2.2) becomes:

$$\mathbb{P}\left(z_k|\mathcal{X}_k\right) \propto exp\left(-\frac{1}{2}||h_k\left(\mathcal{X}_k\right) - z_k||^2_{\Pi_k}\right) \tag{2.3}$$

Where for some vector e, $||e||^2_{\Pi} = e^T \Pi e$ as the squared Mahalanobis distance. We make the similar assumption that the prior $\mathbb{P}\left(\mathcal{X}\right)$ is Normally distributed:

$$\mathbb{P}\left(\mathcal{X}\right) \propto exp\left(-\frac{1}{2}||h_0\left(\mathcal{X}\right) - z_0||^2_{\Pi_0}\right) \tag{2.4}$$

for a given $h_0\left(\cdot\right)$, prior mean z_0 and information matrix Π_0.

Maximizing the posterior is the same as minimizing the **negative log-posterior**, therefore the MAP estimate in Eq. (2.2) becomes:

$$\mathcal{X}^* = \arg\min_{\mathcal{X}} -log\left(\mathbb{P}\left(\mathcal{X}\right)\prod_{k=1}^{m}\mathbb{P}\left(z_k|\mathcal{X}_k\right)\right) = \arg\min_{\mathcal{X}} \sum_{k=1}^{m}\frac{1}{2}||h_k\left(\mathcal{X}_k\right) - z_k||^2_{\Pi_k} \tag{2.5}$$

Which is a nonlinear least squares problem, given that, as in most robotics problems, $h(\cdot)$ is a nonlinear function. It is worth noting that the formulation in Eq. (2.5) follows from the assumption of Normally distributed noise. If other assumptions are made on the noise distribution a different cost function would be formulated; for instance, if the noise were modeled as a Laplace distribution, the squared ℓ_2-norm in Eq. (2.5) is replaced by the ℓ_1-norm. It is also common to substitute the squared ℓ_2-norm in Eq. (2.5) with a robust loss functions such as Huber or Tukey loss [59] to increase resilience to outliers.

The reader with a background in computer vision may notice the resemblance between Eq. (2.5) and **bundle adjustment** (BA) in the Structure from Motion [145] problem; both Eq. (2.5) and BA stem from a maximum-a-posteriori formulation, however there are two features that distinguish SLAM from BA. First, the factors in Eq. (2.5) are general and not constrained to model projective geometry as in BA. These factors usually include a wide variety of sensor models (**sensor-fusion**) such as inertial sensors, wheel encoders, GPS, laser scanners, to mention a few. Second, Eq. (2.5) is designed to be solved **incrementally**: new measurements are made available at each time step as the robot moves through an environment, and an updated state-estimate is computed, usually with real-time constraints. Bundle Adjustment focuses on the batch solution after all measurements have been collected.

The solution to Eq. (2.5), as in most nonlinear least squares problem, is done via successive linearizations, e.g., the Gauss-Newton (GN) or Levenberg-Marquadt methods are the classical algorithms . The Gauss-Newton method starts at an initial guess $\hat{\mathcal{X}}$, and proceeds iteratively. At each iteration GN approximates the minimization problem Eq. (2.5) as

$$\delta^*_{\mathcal{X}} = \arg\min_{\delta_{\mathcal{X}}} \frac{1}{2}\sum_{k=0}^{m} ||A_k\delta_{\mathcal{X}} - b_k||^2_{\Pi_k} = \arg\min_{\delta_{\mathcal{X}}} \frac{1}{2}||A\delta_{\mathcal{X}} - b||^2_{\Pi} \tag{2.6}$$

where $\delta_{\mathcal{X}}$ is a small "update" w.r.t. the linearization point $\hat{\mathcal{X}}$, $A_k \doteq \frac{\partial h_k(\mathcal{X})}{\partial \mathcal{X}}$ is the Jacobian of the measurement function $h_k(\cdot)$ with respect to the state vector $\mathcal{X}$, $b_k \doteq z_k - h\left(\hat{\mathcal{X}}\right)$ is the **residual error** at $\hat{\mathcal{X}}$. On the right hand side of Eq. (2.6) **A** and **b** are obtained by stacking A_k

(resp. b_k); Π is the block-diagonal matrix including the measurement information matrices Π_k as diagonal blocks. The optimal update $\delta^*_{\mathcal{X}}$ which minimizes Eq. (2.6) can be computed in closed form from the normal equations:

$$\delta^*_{\mathcal{X}} = \left(A^T \Pi A\right)^{-1} A^T \Pi b \tag{2.7}$$

This allows, at each iteration, the linearization point to be updated via $\hat{\mathcal{X}} \leftarrow \hat{\mathcal{X}} + \delta^*_{\mathcal{X}}$. The matrix $\left(A^T \Pi A\right)$ is an approximation of the **Hessian.**

Vector spaces. Up until this point we have assumed that $\mathcal{X}$ is embedded in a vector space (thus the sum $\hat{X} + \delta^*_{\mathcal{X}}$ is well defined). When $\mathcal{X}$ includes variables belonging to smooth manifolds (e.g., rotations), the general structure of the GN method remains unchanged, but the euclidean sum ($\hat{X} + \delta^*_{\mathcal{X}}$) is replaced with a suitable mapping, called a **retraction** [2]. It is common to denote the retraction operator with $\oplus$ which maps the "small correction" $\delta^*_{\mathcal{X}}$ defined in the tangent space of the manifold at $\hat{\mathcal{X}}$, to an element of the manifold, i.e., the linearization point is updated as $\hat{\mathcal{X}} \leftarrow \hat{\mathcal{X}} \oplus \delta^*_{\mathcal{X}}$. A common retraction is the lie algebra of SO(3), denoted by $\mathfrak{so}(3)$. In this case the retraction is the exponential map, and its inverse operator is the logarithm (which goes from $SO(3)$ to $\mathfrak{so}(3)$). The $\mathfrak{so}(3)$ lie algebra is motivated and detailed in Appendix A.

Problem Sparsity. A key element behind all modern SLAM solvers is the the Jacobian matrix A appearing in Eq. (7.4) is sparse, and that sparseness is dictated by the topology of the underlying factor graph. This enables the use of fast linear solvers to compute $\delta^*_{\mathcal{X}}$ [65, 63, 80]. This also allows the design of **incremental** or **online** solvers, which update the estimate of $\mathcal{X}$ as new observations are acquired, in real-time [136, 63, 65, 122]. Many current SLAM libraries (e.g., GTSAM [29], g2o [80], Ceres [4], iSAM [63] and SLAM++ [130]) are able to solve problems with tens of thousands of variables within seconds.

MAP vs. Filtering. The formulation described up to this point is commonly referred to as **maximum-a-posteriori** estimation, **factor graph optimization**, **graph-SLAM**, **full smoothing**, or **smoothing and mapping** (SAM). The term **smoothing** is used since all poses are op-

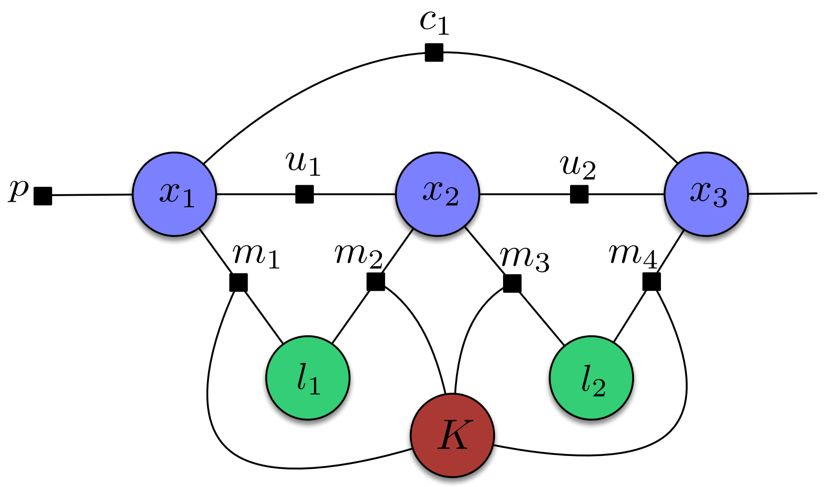


Figure 2.2: SLAM as a factor graph. Blue circles denote robot poses at consecutive time steps (x1,x2,...), green circles denote landmark positions (l1 , l2 , . . .), the red circle denotes the variable associated with the intrinsic camera calibration parameters (K). Factors are shows are black dots: the label u marks factors corresponding to odometry constraints, m marks factors corresponding to camera observations, c denotes loop closures, and p denotes prior factors.

timized over, as opposed to filtering techniques which maintain only the most recent pose. MAP estimation has been proven to be more accurate and efficient than the original nonlinear filtering approaches for SLAM. The surveys [9, 33] provide an overview on filtering approaches and [138] provides an extensive comparison between filtering and smoothing approaches. It is worth noting however, that some EKF-based SLAM systems have also demonstrated state-of-the-art performance. Examples of modern EKF-based SLAM systems include the Multi-State Constraint Kalman Filter of Mourikis and Roumeliotis [106] and the visual-inertial navigation systems of Kottas **et al.** [75] and Hesch **et al.** [53]. The performance gap between filtering and smoothing based SLAM approaches narrows as the linearization point of the EKF improves (as is the case with visual-inertial navigation), with the use of sliding-window filters and when the sources of inconsistency in the EKF are addressed [75, 54, 1].

As will be discussed in the next section, the MAP estimation described here is performed on pre-processed sensor data. As such, it is often referred to as the SLAM **back-end** since it does not concern itself with receiving, pre-processing or the data-association aspects of sensor handling.

2.3 SLAM front-end

When implementing a SLAM algorithm, it is not practical to write the sensor measurements as an analytic function of the state, as required in the MAP estimation described in Section 2.2. If the raw sensor data is an image, we would have to represent the intensity of each pixel as a function of the SLAM state; this difficulty arises even with simpler sensors such as single-beam lasers. The underlying problem is that we are not able to design a sufficiently general and tractable representation of the environment, which would connect the the direct measurements to the state parameters we wish to estimate. This is the main motivation behind the so-called SLAM **front-end**, which is broadly tasked with three main functions: First, translating raw sensor data into its relevant features. For example, in sparse vision-based SLAM the front-end extracts the pixel-location of a few distinguishable points in the environment; these pixel locations are much more amenable to modeling in the back-end. Second is **data association**; associating each measurement (pixel location) to a specific landmark (3D point). More abstractly, the data association module is tasked with associating each sensor measurement z_k with a subset of the state vector $\mathcal{X}_k$ such that $z_k = h_k\left(\mathcal{X}_k\right) + \epsilon_k$. Finally, the front end may provide an initial guess for the state vector $\mathcal{X}$ in the nonlinear back-end optimization. Figure 2.1 provides a pictorial representation of this division of tasks in a SLAM system. The front-end block's data association module contains two main components: short-term data association and long-term. Short-term is tasked with associating corresponding features in consecutive sensor measurements; for example, the short-term association would associate two pixel measurements in consecutive frames to the same 3D point, this is what is commonly referred to as pure visual odometry [41]. Long-term data association, also known as **loop closure** is in charge of associating new measurements to older landmarks. It is common for the back-end to provide feedback information to the front-end to support loop closure detection and validation. The SLAM front-end is naturally sensor-dependent, since the notion of a **feature** changes depending on the sensor we consider; it may be a pixel location for an image provided by a camera, or a laser scan provided by a 3D laser scanner.

Chapter 3

Multi-Sensor SLAM with Online Self-Calibration and Change Detection

3.1 Problem Statement

Autonomous platforms equipped with visual and inertial sensors have become increasingly ubiquitous. Generally these platforms must undergo sophisticated calibration routines to estimate extrinsic and intrinsic parameters to high degrees of certainty before sensor data may be interpreted and fused. Even once fielded, these platforms may experience changes in these parameters. Self-calibration addresses this by inferring intrinsic and/or extrinsic parameters pertaining to proprioceptive and exteroceptive sensors without using a known calibration mechanism or a specific calibration routine. The motivation behind self-calibration is to remove the explicit, tedious, and sometimes nearly impossible calibration procedure from robotic applications such as localization and mapping. By continuously estimating calibration parameters, no prior knowledge of calibration procedures is required. Furthermore, with the addition of statistical change detection on calibration parameters, long-term autonomy applications are greatly robustified.

Most current techniques for vision-aided inertial navigation use filtering approaches [61, 69, 106] or a smoothing formulation. In either case the estimation is made constant-time by rolling past information into a prior distribution. Filtering methods present the significant drawback of introducing inconsistencies due to linearization errors of past measurements which cannot be corrected post hoc, particularly troublesome for non-linear camera models. Some recent work has tackled these inconsistencies; see, e.g. [87, 55, 23, 88]. The state-of-the-art includes methods to estimate poses and landmarks along with calibration parameters, but these approaches do not

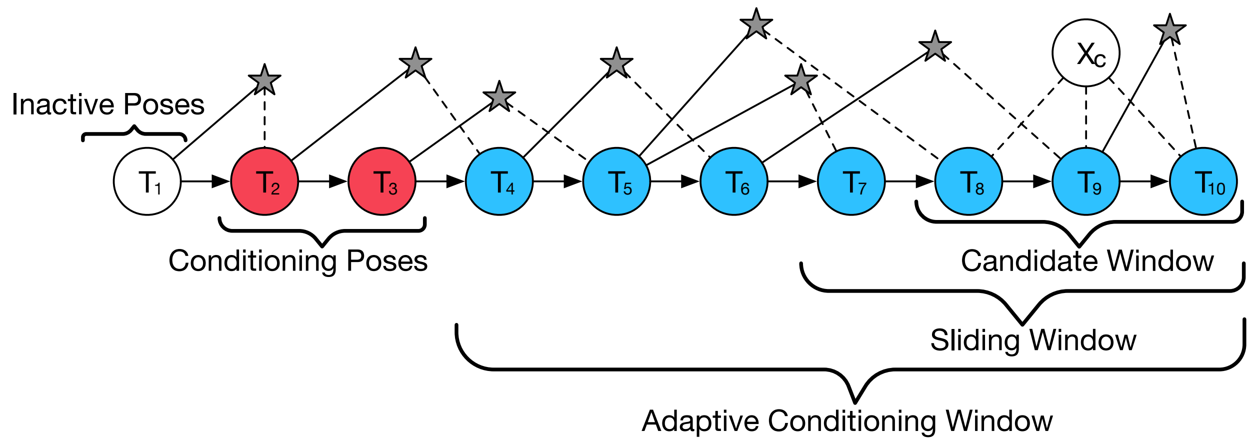


Figure 3.1: Example pose graph. Poses being estimated (*blue*) are conditioned on past poses (*red*) and landmark positions (stars). Both the fixed sliding window and the adaptive window are conditioned on previous poses. The candidate window is not conditioned since it does not make the assumption that previous poses are correctly estimated.

output the marginals for the calibration parameters, which are desirable for long-term autonomy applications.

To address these considerations, we propose a method that avoids using any prior distribution; instead, a conditioning approach is used [67], coupled with selecting only highly informative segments of the trajectory [68]. The method discards segments capturing degenerate motions which provide little to no information for both camera intrinsic and camera-IMU extrinsic [61, 69] parameters. However, unlike the intrinsic parameters of a linear camera model [66], the convergence basin for the six degree of freedom camera-IMU transform is found to be very narrow. An initialization procedure similar to [32, 19] is employed to initialize the camera-IMU transform, which is then used in a maximum-likelihood estimator. The use of a maximum-likelihood formulation is especially useful as it provides the covariance matrix for the estimated parameters, which makes it possible to establish a fitness score for each segment of the trajectory.

We also propose an extension to the framework presented in [68], allowing for multiple sensors to be self-calibrated in an online setting, leveraging [61, 69] to disambiguate unobservable degrees of freedom. Note that while the global position of the IMU and the rotation axis about gravity are *not* observable, the following quantities *are* generally observable: 1) IMU roll and pitch with respect to the horizontal plane; 2) IMU position, orientation and velocity with respect to the initial IMU position; 3) feature position with respect to the initial IMU position; and 4) IMU-to-camera transformation. Finally, we introduce per-sensor candidate trajectory segments, which we find to be necessary to properly estimate each sensors' relevant parameters online.

3.2 Formulation and Methodology

3.2.1 Initialization

As shown in [32, 19], having a good initial estimate can mean the difference between fast convergence and complete divergence. As such, we leverage the work from [32, 61, 69] which shows that with a minimum of three frames and five tracked features, it is possible to obtain the camera-to-IMU rotation. This initial rotation estimate can then be used to solve a linear system for an initial guess at the translation estimate.

We consider the scenario where enough (five or more) features are observed across at least three frames. The tracked features can be used to obtain the relative rotation between two camera frames i, j: ${}^{C}R_{ij}$ and integrating the IMU measurements to obtain the relative rotation: ${}^{B}R_{ij}$, where C represents the camera frame and B the body frame, which is defined without loss of generality as the IMU frame. The following equation relates the camera rotation to the body rotation:

$$ {}^{C}R_{ij} = {}^{C}_{B}R\,{}^{B}R_{ij}\,{}^{B}_{C}R \Rightarrow {}^{C}R_{ij}\,{}^{C}_{B}R = {}^{C}_{B}R\,{}^{B}R_{ij}, \tag{3.1} $$

where ${}^{C}_{B}R$ is the rotation of the body frame in the camera frame. In order to obtain ${}^{C}_{B}R$ we employ an error-state formulation to minimize a robustified over-constrained least squares problem.

In our experience we find that collecting more than 3 frames yielded more reliable estimates; therefore, we use 20 frames for the initial rotation estimate. Once the estimate on ${}^{C}_{B}R$ has converged, translation can be obtained by employing the method described in [32] by solving a linear system derived from transferring the 3D position of a landmark from the camera to the body frame.

3.2.2 Constant Time Self-Calibration

The constant time self-calibrating framework is briefly summarized here; for more details, refer to [66]. Due to the limited observability and high connectivity of calibration parameters in the SLAM graph, it is impractical to estimate these parameters in real-time applications using conventional filtering or smoothing approaches [106, 86, 84, 88]. Instead every segment of m frames in the trajectory is analyzed, and the n most informative segments are added to a *priority queue*, where m and n are tuning parameters dependent on the the calibration parameters being estimated. In order to assess the informativeness of a segment, a score is computed based on the marginals of the calibration parameters estimated by a particular *candidate segment*.

If the candidate segment outperforms the worst-scoring window in the priority queue by a predefined margin, it is swapped in. Every time the priority queue is updated, a batch optimization

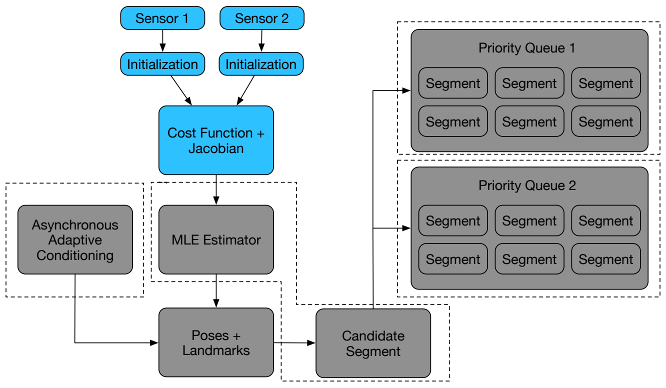


Figure 3.2: System architecture with two sensors. For new sensors to be added only the blue boxes need to be provided. Asynchronous Adaptive Conditioning and the Priority Queue boxes each run in their own thread (*dotted regions*). The main thread is only tasked with the maximum-likelihood estimator and analyzing candidate segments.

over poses, landmarks and calibration parameters is run on all the segments in the queue to obtain a new set of calibration parameters. As such, the priority queue represents a rolling estimate of the n most informative segments in the trajectory. For estimating camera intrinsic parameters, such as focal length and principal point, only visual measurements are used in the candidate segment. When the camera-to-IMU transform is estimated, inertial residuals are added to the candidate window estimation. The priority queue optimization's null space therefore requires careful treatment as it is carried out over several non-continuous segments of the trajectory with varying sensor data. Figure 4.3 shows the optimization windows over a sample set of poses. Figure 3.2 shows the proposed architecture for multiple sensors.

3.2.3 Change Detection

The priority queue posterior (with covariance Σ'_{PQ}) represents the uncertainty over the calibration parameters considering the top k segments in the trajectory. As these segments are usually not temporally consecutive, this distribution encodes the long term belief over the calibration parameters. Conversely, the candidate segment posterior (with covariance Σ_s) is calculated based on the most recent measurements and represents an instantaneous belief over the calibration parameters. If there is a sudden change in calibration parameters, for example if the camera is rotated or moved to a different location on the platform, then this will manifest as a difference in the means of the two posterior distributions. This task of comparing the means of two multivariate normal distributions with different covariances is known as the Multivariate Behrens-Fisher problem.

When the posterior of the priority queue and the candidate segment is over a set of calibration parameters that represent an SE(3) pose, special attention has to be given to comparing the means of these distributions, particularly with regards to the rotation. A minimal *local parameterization* is used for the rotation component of the 6 DOF SE(3) pose, so when comparing two posteriors over rotations in the $\mathfrak{so}(3)$ *tangent space*, one posterior must be transported to the tangent space of the other by means of the Adjoint map, which for SO(3) is:

$$Ad_R : \mathbb{R}^3 \to \mathbb{R}^3, \quad Ad_R = R, \tag{3.2}$$

which allows moving the matrix exponential from the right-hand side to the left-hand side:

$$A \cdot exp(\widehat{x}\,) = exp(\widehat{Ad_A \cdot x}) \cdot A, \tag{3.3}$$

where if $q \in \mathfrak{so}(3)$ is in minimal 3-vector tangent representation, and $M^{-}_{3\times3}$ is the space of (3×3) skew-symmetric matrices, then the map $\widehat{(\cdot)} : q \to M^{-}_{3\times3}$.

By transporting the tangent space rotation posterior from the candidate segment to the tangent space of the priority queue posterior, the null hypothesis that the means are equal can be tested:

$$H_0 = \mu_{PQ} = \mu_S \tag{3.4}$$

The F distribution for the null hypothesis is as in [68].

3.2.4 Adaptive Asynchronous Conditioning

An adaptive asynchronous conditioning [67] solution is employed to avoid the use of a prior distribution on the sliding window SLAM. When conditioning is used instead of marginalization, current active parameters are conditioned on previous parameters, which are assumed to be correct. However since new information may alter the estimate for previous poses, a sliding window pose and landmark estimation is run on a separate thread. This sliding window can adaptively increase its size to alter previous poses based on new measurements. The criteria to increase the window is based on the "tension" of the conditioning residuals, explained as follows. Conditioning residuals are the residual terms connecting an active and inactive pose. For example, a landmark that has a reference frame in an inactive pose, but is seen in an active pose will have a conditioning visual residual. The window is expanded when the the current estimate for a parameter falls outside of the expected estimate based on the conditioning residual. Since multiple sensor modalities are used, the Mahalanobis distance of each conditioning residual is thresholded in a χ^2 test to probabilistically determine when a residual is outside of its expected interval (inducing "tension" in that residual).

3.3 Experimental Results

In order to evaluate the proposed method, experiments were run on two sensor platforms known as "rigs." Both rigs were equipped with a monocular camera and a commercial grade MEMS-based IMU. Rig **A** is a smartphone-like mobile device with an integrated global shutter camera with a wide field-of-view lens at 640×480 resolution and a commercial MEMS IMU sampled at 120Hz. Rig **B** is a Ximea MQ022CG-CM camera with a wide field-of-view lens at 2040×1080 resolution downsampled to 640×480 coupled with a LORD MicroStrain 3DM-GX3 MEMS IMU, sampled at 200Hz. Cameras on both rigs capture images at 30 frames per second. In all experiments, the AAC system is comprised of a fixed-window estimator with a 10 keyframe window width and an asynchronous adaptive estimator (as per Section 3.2.4) with a minimum window size of 20 keyframes. As broached in Section 3.2.1, when both the camera intrinsic parameters

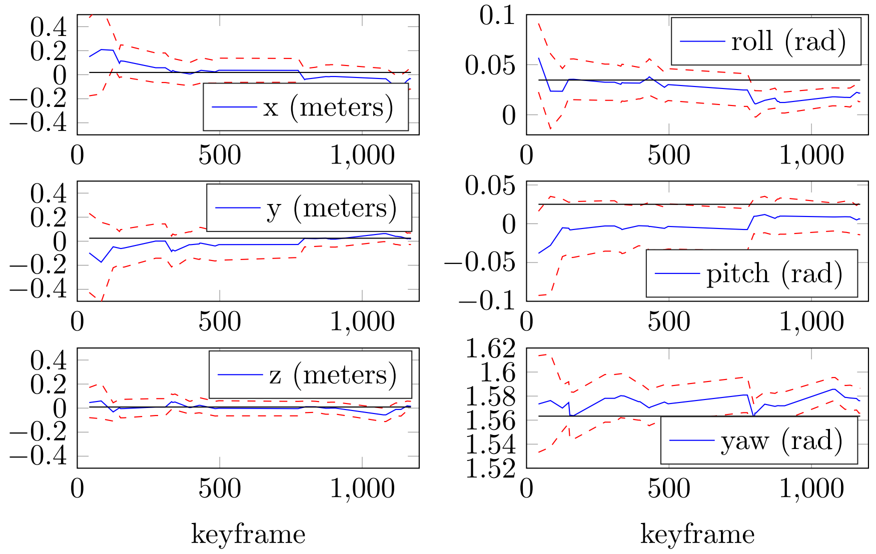


Figure 3.3: Results of a reconstructed indoor dataset spanning 1200 keyframes and 2972 frames. The priority queue consisted of 5 segments with 30 poses in each segment. Camera-to-IMU translation and rotation estimates (*solid blue line*), with their 3 sigma bounds (*dotted red line*). The pseudo ground truth (*solid black line*), obtained by offline calibration procedures is shown to be close to the online estimates, with average sub-degree rotation error and centimeter-level translation error.

and the camera-to-IMU transform are unknown, an initial batch optimization comprising all poses, landmarks and calibration parameters (but no IMU measurements) runs until its entropy falls below a predetermined threshold, at which point the camera intrinsic calibration is handed over to the self-calibration framework discussed in Section 3.2.2. At this point the IMU initialization procedure is engaged—first separately estimating rotation and translation by solving a linear system, then handing over initial estimates on the camera-to-IMU transform to a batch estimation for refinement. Once the batch camera-to-IMU estimation has fallen below a predetermined entropy, the estimation is passed on to the rolling self-calibrating framework for constant-time estimation.

A second experiment was performed on Rig **B**, where only the camera-to-IMU parameters were being estimated, but the position of the IMU was physically changed mid-dataset. This experiment's results are show in Figure 3.5.

3.4 Discussion

In Figure 3.4, a sharp drop is witnessed in uncertainty on all intrinsic parameters around keyframe 820, where a particularly informative segment was swapped into the queue. The same behavior is not witnessed around keyframe 820 for the camera-to-IMU transform estimate in Figure 3.3, which strongly suggests the need for different queues for different sensors. Supporting the initialization sequence used for SE(3) transform approximation, Figure 3.5 demonstrates rapid convergence to new translation parameters when the sensors are moved with respect to one another on Rig **B**. The entropy of the priority queue increases temporarily until enough post-change segments are added.

Some discrepancies between the offline values and the estimates from the priority queue can be observed (such as on the rotation values in Figure 3.3). This can be caused by a number of factors: 1) the offline calibration is only a pseudo-ground truth, and 2) lack of observability of these parameters, especially yaw, since we only use naturally occurring features. Note that the self-calibration sequence we suggest relies on non-degenerate motions that excite the appropriate degrees of freedom so as to render them observable, which we have found to occur naturally in

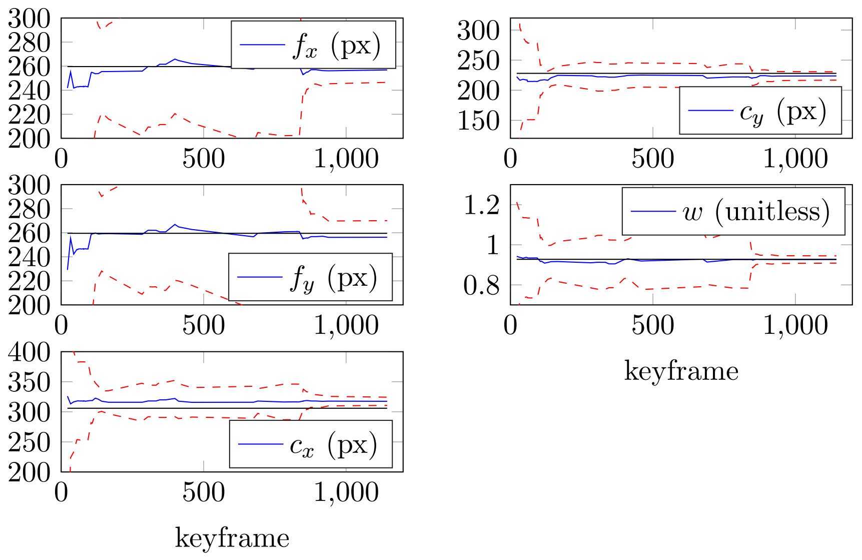


Figure 3.4: Self-calibration camera intrinsic parameters. Neither camera intrinsic or camera-to-IMU extrinsic parameters were known. Even with total uncertainty on *all* calibration parameters at the start, convergence to offline values is observed for both camera intrinsic and extrinsic parameters.

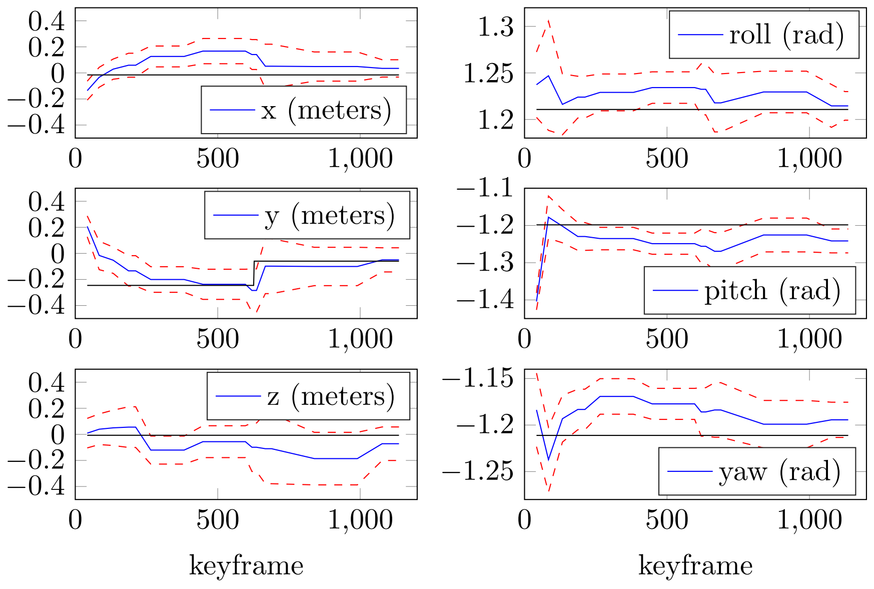


Figure 3.5: Indoor dataset on Rig **B**, the IMU position was manually changed mid-dataset. Only the y component of translation was changed, all other parameters remained the same. as shown by the pseudo ground truth line (*black line*). The system automatically detected a change in mean and re-estimated all parameters.

experimental hand-held datasets.

A particular failure case is through slow changes of calibration parameters through a data collection. Changes in parameter values are currently induced as a step function; however, if a calibration parameter changes incrementally over time, it will not trigger a change event, as per Section 3.2.3. Instead, new segments with low entropy will be swapped into the priority queue, mixed with past segments that presented a different mean. Another failure case is related to the determinant-based scoring system, which could result in a very low uncertainty for an unobservable parameter. These drawbacks warrant further development of a more robust scoring system.

3.5 Summary

This chapter presents online, constant-time self-calibration and change detection with re-calibration for joint estimation of camera-to-IMU transform and camera intrinsic parameters, using only naturally occurring features. The system is evaluated with experimental data and shown to converge to offline calibration estimates with centimeter level accuracy for camera-to-IMU translation, and sub-degree accuracy for rotation. The statistical change detection framework presented in [68] and summarized in Section 3.2.3 has been extended to the camera-to-IMU transform, including a statistical comparison of distributions over candidate segments for a SE(3) pose.

The use of an adaptive conditioning window for re-estimation of past poses allows this framework to operate in long-term applications where the accumulation of linearization errors in a prior distribution would lead to significant drift. We presented a framework that supports adding additional sensors while maintaining online operation. To the authors' best knowledge this is the first application of multi-sensor self-calibration with automatic change detection and re-estimation of parameters.

Chapter 4

Drift-Correcting Self-Calibration for Visual-Inertial SLAM

4.1 Problem Statement

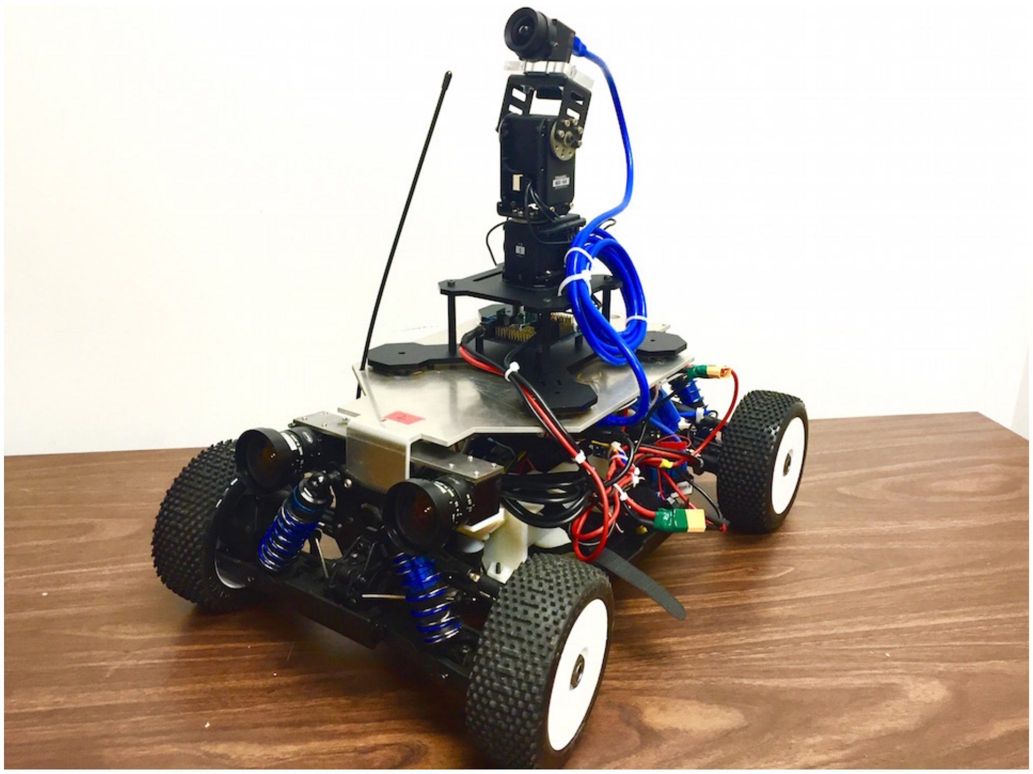
Autonomous platforms destined for long-term applications equipped with visual and inertial sensors have become increasingly ubiquitous. Generally these platforms must undergo sophisticated calibration routines to estimate extrinsic and intrinsic parameters to high degrees of certainty before sensor data may be interpreted and fused. Once fielded, calibration parameters are generally fixed for the lifetime of the platform, or are modeled as a piecewise constant function [68]. For many applications however, these platforms may experience gradual changes in calibration parameters due to e.g. temperature dilation, non-rigid mounting or accidental bumps that can change both sensor intrinsic and extrinsic parameters. Self-calibration addresses this by inferring intrinsic and extrinsic parameters pertaining to proprioceptive and exteroceptive sensors without using a known calibration target or a specific calibration routine. The motivation behind self-calibration is to remove the explicit, tedious, and sometimes nearly impossible calibration procedure from robotic applications and to enable robust long-term autonomous operation. Most approaches to online self-calibration that do not rely on marginalization (such as filtering, which is subject to linearization errors of past measurements) either assume calibration parameters remain constant or change in a piecewise constant fashion. These approximations work well for single digit percent drift over calibration parameters on relatively short (<200m) trajectories but induce considerable drift for longer periods of operation. Figure 4.1 (bottom) shows a simulation of time-varying calibration parameters and the traditional piecewise-constant approximations, demonstrating long periods of incorrectly-estimated parameters.

We present a novel approach that approximately models calibration parameters as a continuous time-varying quantity, which we refer to as drift-correcting self-calibration (DCSC). By continuously estimating calibration parameters, no prior knowledge of calibration values or procedures is required. Furthermore, with the addition of statistical change detection and regression on drifting calibration parameters, long-term autonomy applications are greatly robustified against accidental changes where the calibration varies over time. This approach is based on probabilistically determining segments where the motion provides enough excitation on the calibration parameters to permit observability [61, 69]. This permits seamless handling of degenerate motions and unknown calibration parameters, enabling the much sought after "power-on-and-go" operation. Our approach includes probabilistic change detection as well as change regression; the former system detects change events that require completely re-estimating calibration parameters, and the latter identifies the start of the change region so that past poses can be re-estimated with the correct calibration parameters. This approach is validated on camera intrinsic and camera-to-IMU extrinsic parameters, however is easily extensible to an arbitrary number of sensors [115]. To the authors' knowledge, DCSC is the first proposed solution to long-term drift due to time varying calibration parameters that does not rely on a prior distribution.

4.2 Related Work

The problem of self-calibration with varying camera intrinsics has received much attention in the literature in part due to the benefits outlined above. Both [56] and [121] considered a batch-solution self-calibration tailored to different intrinsic parameters. [39] presented a method to calibrate the varying intrinsics of a pinhole camera in a batch setting, given the rotation of the camera was known. A solution was also offered to align the rotation sensor and camera data in time.

Many current techniques for vision-aided inertial navigation use filtering approaches (e.g. [61, 69, 106]) or a smoothing formulation. In either case the estimation is made constant-time



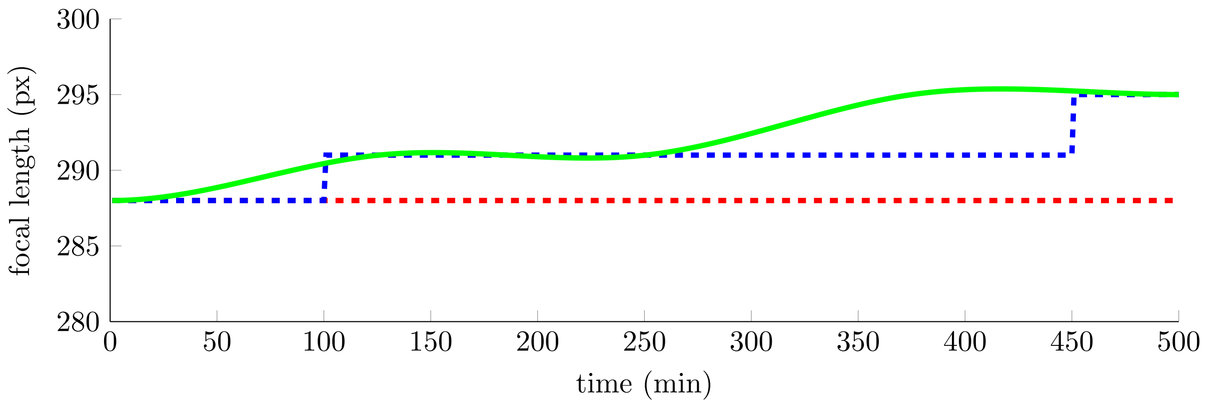


Figure 4.1: Top: experimental robotic vehicle with a camera mounted on a pan-tilt unit for changing camera-to-IMU extrinsics (IMU rigidly mounted inside body of platform). Bottom: camera focal length ground truth (*solid green line*) over an 8 hour trajectory using our visual-inertial simulation pipeline. The piecewise-constant approximation (*dashed blue line*) lags behind the ground truth due to uncertainty in the measurements which make a small change in calibration parameters indistinguishable from noise. The constant assumption is the *dashed red line*. The ground truth shows the continuous time-varying nature of the parameter, which our method approximates.

by rolling past information into a prior distribution. Filtering methods present the significant drawback of introducing inconsistencies due to linearization errors of past measurements which cannot be corrected post hoc, particularly troublesome for non-linear camera models. Some recent work has tackled these inconsistencies; see, e.g. [87, 55, 23, 88]. The state-of-the-art includes methods to estimate poses and landmarks along with calibration parameters, but these approaches do not output the marginals for the calibration parameters, which are desirable for long-term autonomy applications.

Building on these works, simultaneous solutions to the SLAM and self-calibration problem have been proposed but generally all online solutions assume constant calibration parameters. [22] proposed a method to recursively estimate camera and landmark 3D parameters as well as the intrinsic parameters of a nonlinear camera model in an online framework. [89] also developed a filtering solution to estimate both the camera pose and also intrinsics and extrinsics for a non-linear camera model with rolling shutter and a commercial grade IMU in an online framework, but that approach does not output covariances in an MLE sense.

4.3 Methodology

The proposed method aims to continuously estimate both intrinsic and extrinsic calibration parameters [115], while also detecting change events due to sensor perturbation and regressing calibration parameters in an arbitrarily large change region. As such, DCSC has three distinct components: *Constant Time Self-Calibration* [66] is needed in order to continuously estimate the instantaneous belief over intrinsic and extrinsic calibration parameters at any point in the trajectory; *Change Detection* [68] signals a statistically significant difference in the means of the instantaneous belief over calibration parameters and the long term belief, indicating a high probability that the calibration parameters have been perturbed; and finally, *Change Regression* checks for the start of the change region and regresses the calibration parameters in that region, allowing for re-estimation of past poses with the correct calibration parameters, reducing the long-term drift. Each of these components are described in the following sections, with emphasis given to Change

Regression which is a novel contribution on which this chapter is focused. Alongside DCSC, a keyframe-based [72] pose-and-landmark non-linear maximum likelihood estimation is performed for real-time map updates. Thus, our implementation details will also describe a system for *adaptive SLAM estimation* [67], which is used to ensure the maximum likelihood poses and landmarks are estimated.

4.3.1 Calibration Parameter Modeling

Central to the proposed self-calibration methodology explained in the following sections is the modeling of how calibration parameters change over time. Most approaches to self-calibration estimate calibration parameters such as camera intrinsics, sensor-to-sensor extrinsics and time offset at the start of the trajectory and make the assumption that those parameters will remain fixed, i.e.: $\boldsymbol{x}_c(t) = \boldsymbol{x}_c(0)$ where $\boldsymbol{x}_c$ represents the calibration parameter vector, a function of time. This assumption is valid for short trajectories, however it breaks down when long term operation is desired, due to the inherent drift in sensor calibrations. Another option is modeling calibration parameters as a multivariable piecewise constant function, and probabilistically detecting change events, as in [68] and [115]:

$$\boldsymbol{x}_c(t) = \boldsymbol{x}_c(t_i) \quad \text{if } t_i \leq t \leq t_{i+1}, \tag{4.1}$$

where t_i, $i = \{1, \ldots, n\}$ defines n points of time in which the estimated calibration parameters are detected to have changed. Note in this case that the time-dependent parameter vector is approximated by a piecewise constant constant function, $\boldsymbol{x}_c(t_i)$. While this approach is well-suited to large sudden changes, it does not handle cases where the calibration parameters are slowly drifting over a long period of time due to the difficulty in distinguishing small changes from sensor noise. Figure 4.1 depicts this scenario.

Given the fact that sensor calibration parameters are often slowly-varying functions of time in long term autonomy applications, we propose the following model:

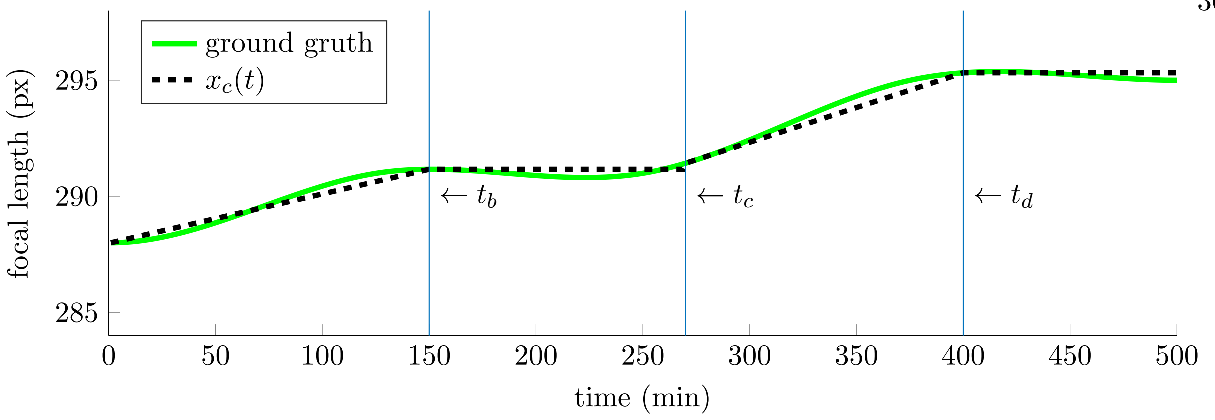


Figure 4.2: Piecewise time-varying calibration parameters over simulated dataset.

$$\boldsymbol{x}_c(t) = \begin{cases} \boldsymbol{x}_c(t_i) & t_i \le t \le t_{i+1} \\ \boldsymbol{f}_i(t) & t_{i+1} \le t \le t_{i+2} \\ \boldsymbol{x}_c(t_{i+2}) & t_{i+2} \le t \le t_{i+3} \\ & \dots \end{cases} \tag{4.2}$$

where e.g. $t_{i+1} \le t \le t_{i+2}$ represents a change region that can be arbitrarily long. The key considerations for this model are 1) determining the start and end points for the change event, and 2) establishing basis functions for $\boldsymbol{f}_i(t)$; Figure 4.2 shows an example of this approach with a linear basis function. Section 4.3.5 goes into detail on both these aspects.

4.3.2 Constant Time Self-Calibration

4.3.3 Initialization

As shown in [32, 19], having a good initial estimate can mean the difference between fast convergence and complete divergence. A good initial guess is needed on both intrinsic and extrinsic calibration parameters. We treat these cases separately as follows:

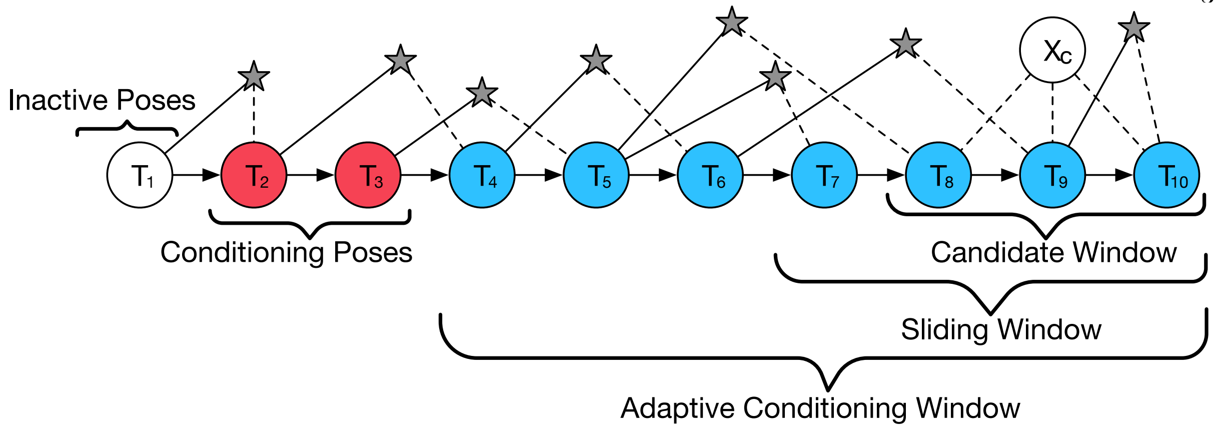


Figure 4.3: Example pose graph. Poses being estimated (*blue*) are conditioned on past poses (*red*) and landmark positions (stars). Both the fixed sliding window and the adaptive window are conditioned on previous poses. The candidate window is not conditioned since it does not make the assumption that previous poses are correctly estimated.

4.3.3.1 Camera Intrinsics Initialization

The camera intrinsic parameters are bootstrapped by running a batch optimization over the entire state vector (rig location, landmark inverse depths and all the camera intrinsic calibration parameters). Once the score of the batch estimation falls below a predetermined threshold, indicating that the uncertainty over calibration parameters is sufficiently small, estimation is handed over to the candidate segments and priority queue, as described in Section 4.3.2.

4.3.3.2 Camera-to-IMU Initialization

We leverage the work from [32, 61, 69] which shows that with a minimum of three frames and five tracked features, it is possible to obtain the camera-to-IMU rotation. This initial rotation estimate can then be used to solve a linear system for an initial guess at the translation estimate. We consider the scenario where enough (five or more) features are observed across at least three frames. The tracked features can be used to obtain the relative rotation between two camera frames i, j: ${}^{C}\boldsymbol{R}_{ij}$ and integrating the IMU measurements to obtain the relative rotation: ${}^{B}\boldsymbol{R}_{ij}$, where C represents the camera frame and B the body frame, which is defined without loss of generality as the IMU frame. The following equation relates the camera rotation to the body rotation:

$$^{C}\boldsymbol{R}_{ij} = {}^{C}_{B}\boldsymbol{R}^{B}\boldsymbol{R}_{ij}{}^{B}_{C}\boldsymbol{R} \Rightarrow^{C} \boldsymbol{R}_{ij}{}^{C}_{B}\boldsymbol{R} = {}^{C}_{B}\boldsymbol{R}^{B}\boldsymbol{R}_{ij}, \tag{4.3}$$

where ${}^{C}_{B}\boldsymbol{R}$ is the rotation of the body frame in the camera frame. In order to obtain ${}^{C}_{B}\boldsymbol{R}$ we employ an error-state formulation to minimize a robustified over-constrained least squares problem.

In our experience we find that collecting more than 3 frames yielded more reliable estimates; therefore, we use 20 frames for the initial rotation estimate. Once the estimate on ${}^{C}_{B}\boldsymbol{R}$ has converged, translation can be obtained by employing the method described in [32] by solving a linear system obtained from transferring the 3D position of a landmark from the camera to the body frame.

4.3.4 Change Detection

The priority queue posterior (with covariance Σ'_{PQ}) represents the uncertainty over the calibration parameters considering the top k segments in the trajectory. As these segments are usually not temporally consecutive, this distribution encodes the long term belief over the calibration parameters. Conversely, the candidate segment posterior (with covariance Σ_s) is calculated based on the most recent measurements and represents an instantaneous belief over the calibration parameters. If there is a sudden change in calibration parameters, for example if the camera is rotated or moved to a different location on the platform, then this will manifest as a difference in the means of the two posterior distributions. The simple difference in means cannot be used as a change detecting mechanism however, since the uncertainty associated to the estimate needs to be taken into account. This procedure, comparing the means of two multivariate normal distributions with different covariances, is known as the Multivariate Behrens-Fisher problem. Using an F distribution as in [68], the null hypothesis that the means of the candidate segment and that of the priority queue are equal can be tested:

$$H_0 : \mu_{PQ} = \mu_s \tag{4.4}$$

By comparing the p-value corresponding to the F distribution to a significance parameter $\alpha = 0.1$ the null hypothesis can be rejected for $p \leq \alpha$. There are several events, such as feature-less environments, motion blur, loss of tracking and non-static features which may lead to an incorrectly estimated posterior for the candidate segment. In order to avoid these scenarios a simple test is used where ν_{cs} consecutive candidate segments must have $p \leq \alpha$ for a change event to be triggered, where $\nu_{\text{cs}} = 3$.

A failure case for this approach when dealing with slow-changing parameters is when the candidate window mean consistently differs from the priority queue mean, but not enough to clearly be distinguishable from noise, so the condition $p \leq \alpha$ for ν_{cs} consecutive segments will not be met and a change event will not be triggered. This will cause the priority queue to slowly drift towards the new calibration parameter as candidate segments are swapped in, however since a change was not detected, candidate segments which where estimated prior to the change event will remain in the queue, resulting in a sub-optimal global estimate.

This prompts a second criterion for detecting a change: if the mean of the priority queue, equal to $\boldsymbol{x}_c$, over the past 3 seconds fits a linear curve with slope $\lambda > \lambda_{\text{th}}$ a tuned threshold, a change event is triggered and the priority queue is re-estimated. This allows for past segments to be removed from the queue and re-estimation on the new parameters. Note that λ_{th} sets the DCSC algorithm's sensitivity to slow changes.

4.3.5 Change Regression

The change detection mechanism presented in Section 4.3.4 will not detect the exact onset of the change event. As shown in Figure 4.4, there can be a considerable time-delay between the start of a change event and the change detection. The critical failure case is a drifting calibration parameter, which is indistinguishable from noise until it differs significantly from the previous estimate. The option of simply making the change detection mechanism more sensitive to changes by adjusting the threshold parameter α is not viable since that will trigger change events on inaccurately estimated candidate segments, causing the priority queue to be routinely cleared and re-estimated, which can result in worse priority queue estimates and has a direct impact on both accuracy and real-time performance.

Instead of attempting to detect the exact onset of a change event as it occurs, the start point for the change event is regressed from the change detection point, which will be after the start of the change: when a change event is triggered as in Section 4.3.4 at keyframe n^* every previous keyframe $n < n^*$ is tested as the starting position for the change event. This is done by leveraging the novel probabilistic change detection introduced in [68].

4.3.5.1 Start Point Estimation

The intuition behind detecting the start point is that, according to the model described in Section 4.3.1, the change region is preceded by a period in which the calibration parameters are relatively constant. Considering that the candidate window encodes the instantaneous belief over calibration parameters, then for every keyframe, the p-value for the null hypothesis test between the priority queue distribution and the latest candidate segment of which that frame was a part is stored. This allows for detecting the constant calibration regions, as depicted in Figure 4.2, by a segment in which the p-value is smaller than a threshold Φ_{th}. We employ the following heuristic: if ν_{sp} consecutive segments have a p-value divergence smaller than Φ_{th}, where $\nu_{\text{sp}} = 50$ is used. If the start point is unable to be estimated the change regression is aborted and the system falls back onto the default setting of re-estimating calibration parameters from the change detection point onward.

4.3.5.2 Parameter Regression

Once the start point for the change event has been determined, the calibration values for all poses during time (t_{i+1}, t_{i+2}) in Eq. (4.2) need to be re-estimated (see Figure 4.2 for an example; poses between times $(0, t_b)$ would require re-estimation). In our implementation this is accomplished by fitting a linear curve to the calibration value at the regressed start point of the change event and the converged priority queue values after the change event. The choice of basis function to fit is dependent on the sensor and how changes are expected. In our experiments camera focal length and camera-to-IMU extrinsics are estimated, which are considered to drift in an approximately linear form for handheld and vehicle-mounted applications. Special care is taken to guarantee that all poses in the change region are re-estimated with the updated calibration parameters.

4.3.6 Adaptive SLAM

Finally, an adaptive asynchronous conditioning [67] solution is employed to avoid the use of a prior distribution on the sliding window estimation. When conditioning is used instead of marginalization, current active parameters are conditioned on previous parameters, which are assumed to be correct. However since new information may alter the estimate for previous poses, a sliding window pose and landmark estimation is run on a separate thread. This sliding window can adaptively increase its size to alter previous poses based on new measurements. The criteria to increase the window is based on the "tension" of the conditioning residuals, explained as follows. Conditioning residuals are the residual terms connecting an active and inactive pose. For example, a landmark that has a reference frame in an inactive pose, but is seen in an active pose will have a conditioning visual residual. The window is expanded when the the current estimate for a parameter falls outside of the expected estimate based on the conditioning residual. Since multiple sensor modalities are used, the Mahalanobis distance of each conditioning residual is thresholded in a χ^2 test to probabilistically determine when a residual is outside of its expected interval (inducing "tension" in that residual).

4.3.7 Visual Tracking

Visual tracking is inspired on the tracking component of [38], where the photometric error of a patch is directly miminized to find the new location of the feature. Harris corners are used for feature initialization, in image regions where there are a small number of active tracks. NCC scores for corresponding feature patches are thresholded at 0.875 to reject large changes in appearance. A keyframing approach [100] is used for improved performance and to deal with situations such as

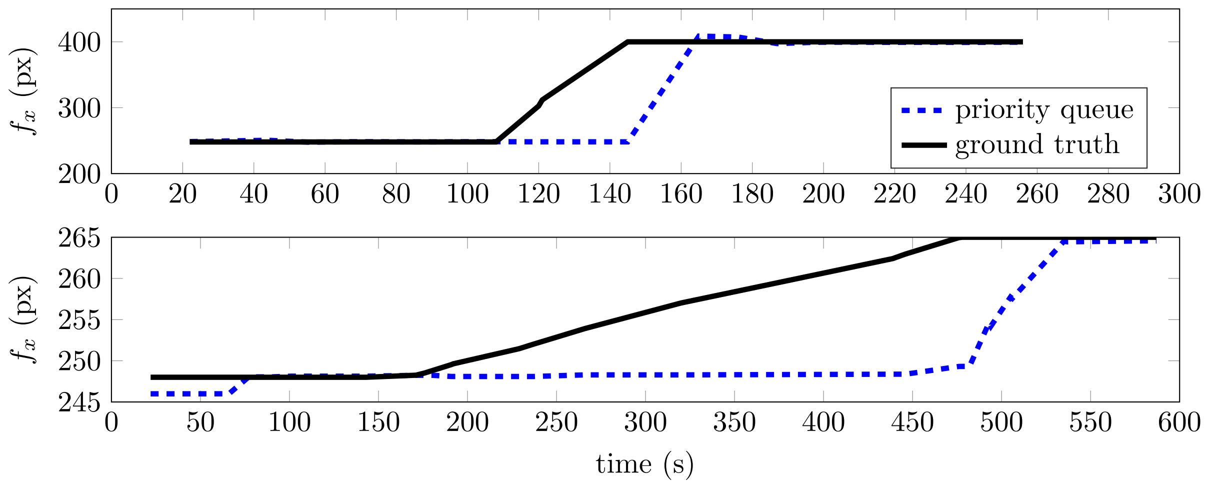


Figure 4.4: Comparison of a fast change of the camera field of view from 130° to 45° (top figure) in camera focal length and a slow drift from 130° to 120° degrees (bottom figure). The drift over a long period of time greatly increases the area between the ground truth and the priority queue estimate, resulting in incorrect estimates for a large segment of the trajectory. Each are averaged over 10 simulations with noise added.

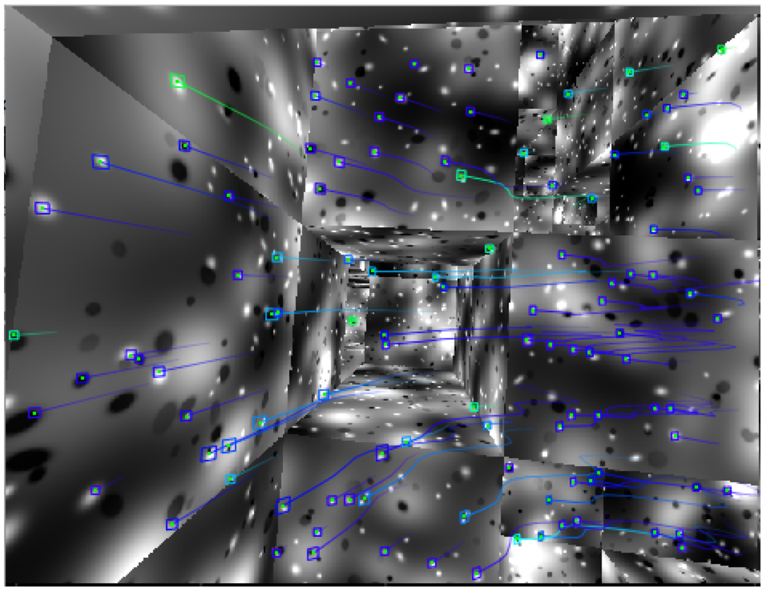


Figure 4.5: Simulated visual trajectory with feature tracks.

stationary camera.

4.4 Simulations

Special attention was given to creating a pipeline for generating simulated visual and inertial data for evaluation of both the proposed algorithm and the effects of not properly accounting for changing calibration parameters over long trajectories. In order to be able to evaluate arbitrary motions, a random 6-DOF pose graph is generated using a system inspired by video game dynamics. This trajectory is then used to carve out a path in maze of cubes, ensuring that any simulated motion will be visually trackable. Figure 4.5 shows one such path, and the corresponding features and feature tracks. Simulated inertial accelerometer and gyroscope data corresponding to the trajectory is also generated for consistent scale and evaluating camera-to-IMU estimation. The use of simulated data allows for exact ground truth comparison, which is especially challenging when evaluating the response to slowly drifting to calibration parameters in real world settings.

4.5 Experiments and Results

4.5.1 Simulation

In order to evaluate the proposed method, the simulation pipeline described in Section 4.4 was used to generate arbitrary trajectories with corresponding synchronized visual-inertial data. A series of Monte Carlo simulations of camera intrinsics and camera-to-IMU calibrations were performed. We chose a set of 5 trajectories, with varying degrees of excitation on each degree of freedom (so as to avoid known degenerate motions). For each trajectory we ran 30 simulations with varying calibration parameters: camera focal length and camera-to-IMU rotation. The camera's field of view was initialized at 130° and changed to 120° over a time period t_{change} drawn from a Gaussian distribution with a mean of 60s and standard deviation of 10s. The camera-to-IMU rotation was initialized at ${}^{C}_{B}\boldsymbol{\rho} = [0.53\ 0.53\ 0.53]^T$ and changed to ${}^{C}_{B}\boldsymbol{\rho} = [0.53\ 0.53\ 0.58]^T$. Camera focal length and camera-to-IMU rotation initial and final values were perturbed by a zero-mean Gaussian with standard deviation of 5°. Simulated camera data was captured at 15 frames per second, with IMU updates at 100Hz. For each projected feature point from the simulated images (640 × 480 resolution), independent zero-mean Gaussian noise with $\sigma = 0.5$ was added to the (u, v) pixel coordinates. Zero-mean Gaussian noise with $\sigma = 10^{-3}$ was also added to the IMU accelerometer and gyroscope measurements and biases. Each simulation was run on three different calibration schemes: the proposed DCSC algorithm, the piece-wise constant method described in [68] and [115] and the constant method where the calibration is obtained in the start of the trajectory and held constant throughout. The start time for the change event for each simulation is drawn from a uniform distribution in the first half of the trajectory. The results of these simulations are shown in Table 4.1 for the DCSC algorithm, where ${}^{B}_{C}\rho$ is the total camera-to-IMU rotation error, % Drift is average translation drift and % Change Start is at which fraction of the change region the start point was detected.

One such trajectory is shown in Figure 4.6 where the DCSC method obtains final translation error of 28m which corresponds to 0.04% of the distance traveled and an average rotation error of

Table 4.1: DCSC Monte Carlo Simulation Results

	f_x Err (px)		f_y Err (px)		c_x Err		c_y Err		${}^B_C\rho$ Err		% Drift	% Change
Set	μ	3σ	μ	3σ	μ	3σ	μ	3σ	μ	3σ	-	-
1	1.01	16.62	4.34	25.21	0.95	10.42	1.14	11.02	0.09	0.05	0.012	0.195
2	2.32	20.24	3.58	27.85	1.32	14.98	2.04	17.90	0.12	0.04	0.069	0.153
3	1.76	14.22	2.91	18.82	0.52	11.52	0.87	9.88	0.16	0.09	0.093	0.107
4	3.42	23.13	4.02	30.14	1.80	12.92	1.91	11.71	0.07	0.03	0.095	0.206
5	1.89	13.87	1.94	15.02	0.89	9.78	0.99	10.53	0.15	0.11	0.094	0.124

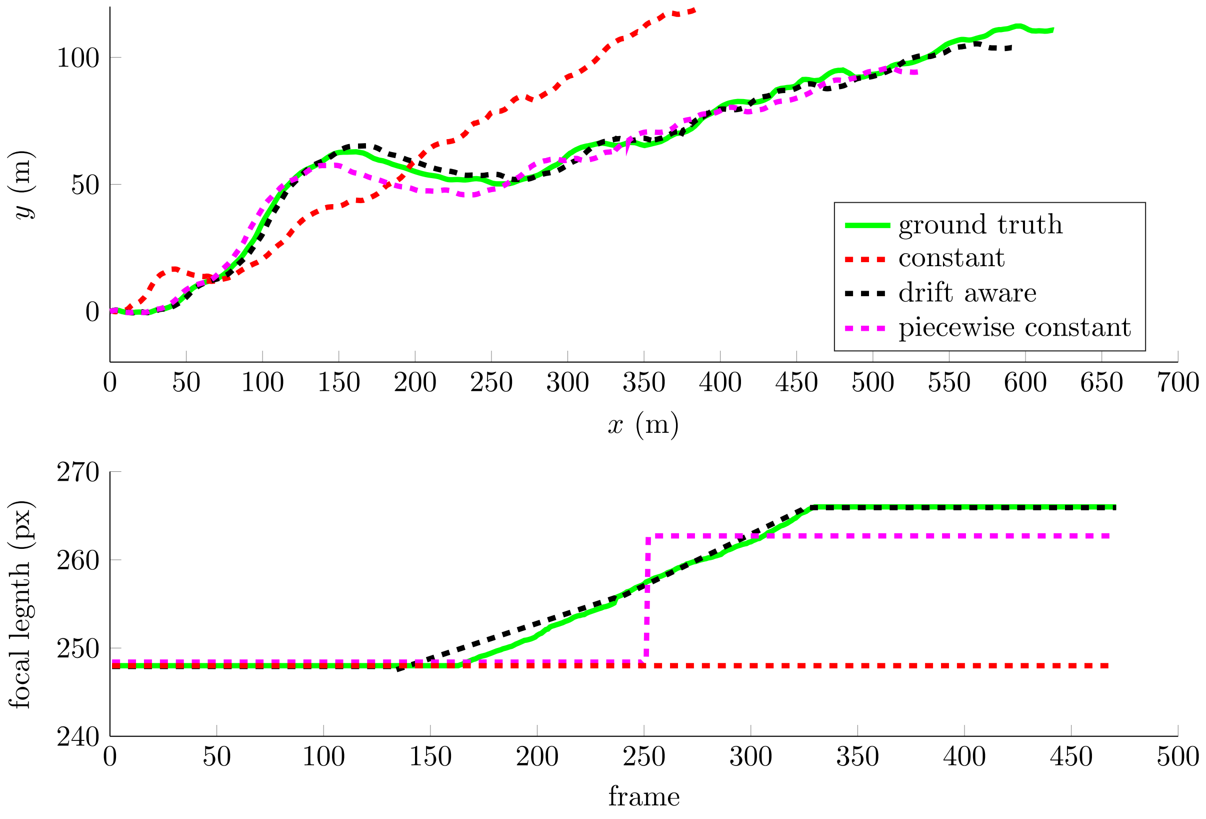


Figure 4.6: 700m simulated trajectory. The camera focal length was changed linearly from 120° to 130° over a 60 second period. The proposed DCSC algorithm obtains the smallest translation error at 0.04% of the total distance traveled. The bottom figure shows the focal length (f_x) as the change occurs, and how each self-calibration scheme responds. DCSC also correctly finds the start and end point of the change event.

0.406 rad

4.5.2 Experiments with the Mobile Platform

In order to determine the performance of the proposed algorithm on data from real hardware, we used our experimental vehicle, depicted in Figure 4.1. The vehicle is equipped with a global shutter Ximea MQ022CG-CM camera with a wide field-of-view lens at 2040×1080 resolution downsampled to 640×480 mounted on a pan-tilt unit and an onboard Gladiator MEMS IMU capturing at 200Hz. Images were captured at 30Hz. The platform was driven over a 300m trajectory where the camera-to-IMU rotation was changed with the pan-tilt sensor over the course of 2 minutes. Figure 4.7 shows the comparison of the piecewise priority queue with it's 3σ bounds and the DCSC algorithm. The covariance on the priority queue spikes at change events when the queue is wiped but quickly tightens around the mean as segments are added to the queue. A ground truth was not available for the experiment so a comparison of start and end poses was used: The DCSC algorithm had a translation error of 1.13m, equivalent to 0.3% of the trajectory. Using the piecewise constant approximation the final translation error was 2.22m, or 0.7% of the trajectory.

4.6 Discussion

This chapter presents online, constant-time self-calibration and change detection with re-calibration for joint estimation of camera-to-IMU transform and camera intrinsic parameters, dealing explicitly with the case of drifting calibration parameters over long trajectories. The system is evaluated with experimental and simulated data and shown to converge to offline calibration estimates even in the presence of slowly drifting calibration parameters. The statistical change detection framework presented initially in [68] is used to detect change regions for drifting parameters and estimate the calibration parameters in the drift region.

The use of a drift correcting self-calibrating framework coupled with adaptive conditioning window for re-estimation of past poses allows this framework to operate in long-term applications where the accumulation of linearization errors in a prior distribution and the accumulation of

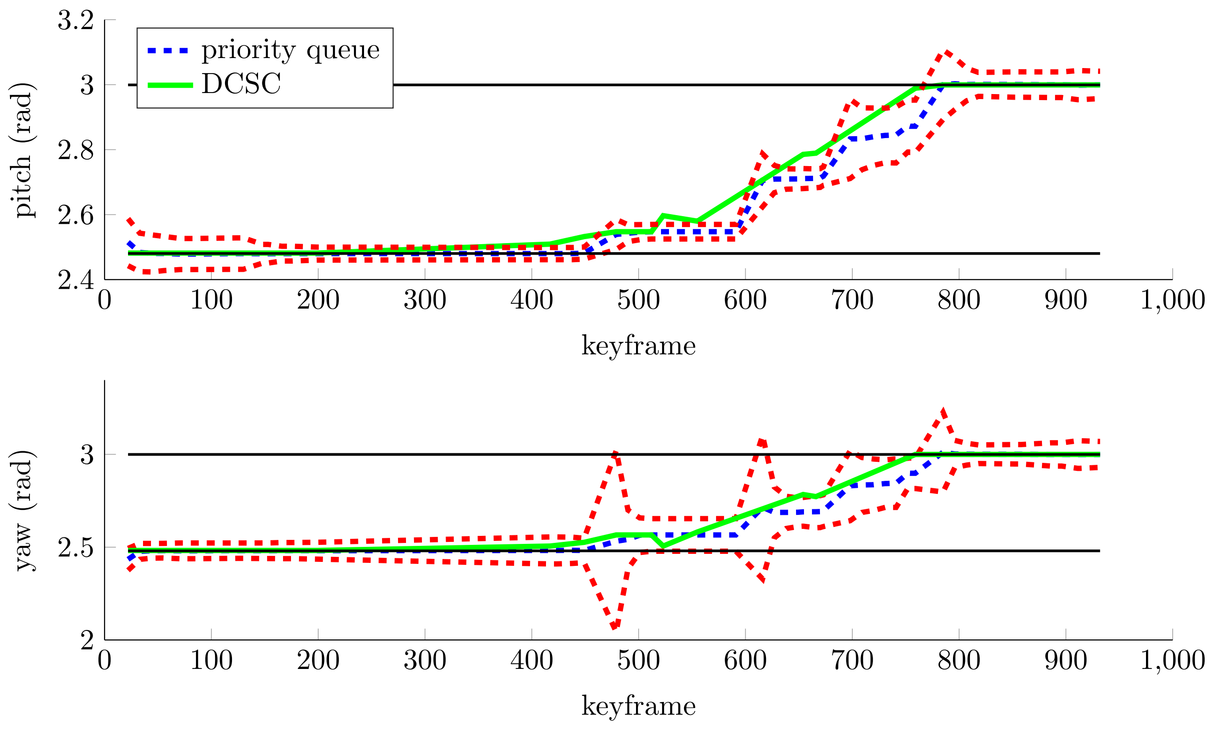


Figure 4.7: Self-calibration and drift correction on experimental platform. Priority queue estimates (*blue dashed line*) and its 3σ bounds (*red dashed line*) compared to the DSCS drift correcting estimates (*green solid line*)

incorrectly estimated calibration parameters over change periods would lead to significant drift. We present an analysis on the effects of inappropriate modeling of calibration parameters over long trajectories, and show how the use of a multivariate probabilistic change detection framework can greatly reduce the drift even in the presence of hard-to-detect incremental changes over time in calibration parameters. This method presents some failure cases that warrant further study, such as when the rate of drift is slow compared to the inherent noise in estimation errors, the boundary detection scheme presented may be inaccurate. An adaptive way of choosing all the tuning parameters is also necessary for this system to be easily usable in practice. In future work we would like to investigate the use of different basis functions, such as higher order polynomials or Gaussian Processes.

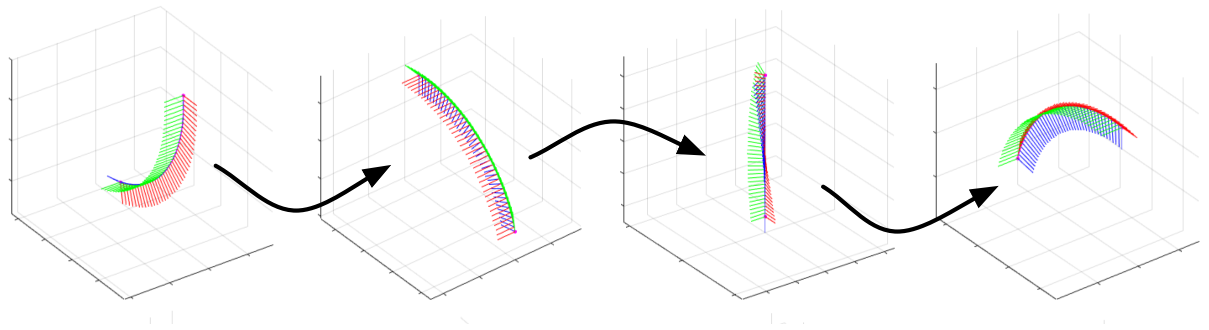
Chapter 5

Reinforcement Learning for Assisted Visual-Inertial Robotic Calibration

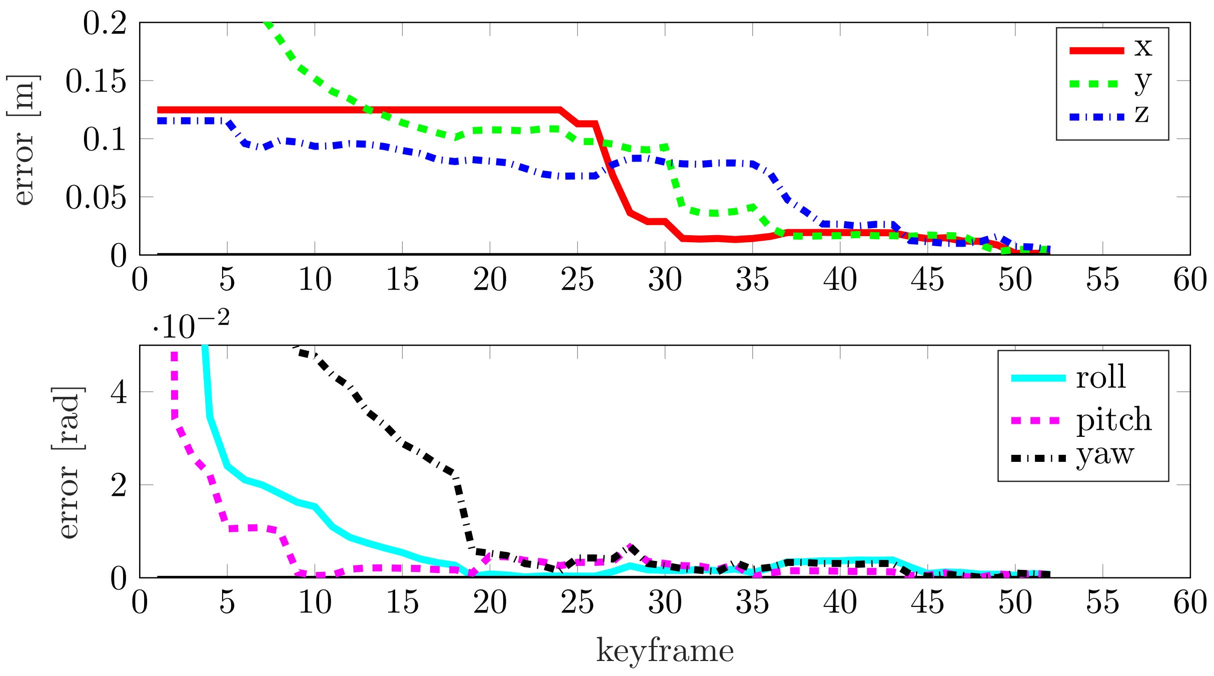
5.1 Problem Statement

Common to all robotic applications are some set of parameters—camera intrinsics, sensor extrinsics, biases, scale factors, model parameters, etc.—that are essential for higher level robotic tasks such as state estimation, planning and control. These parameters are called **calibration parameters** and are usually obtained offline with specific and sometimes sophisticated calibration routines, or online in a self-calibrating framework that relies on sufficiently exciting motions and naturally occurring data. "Sufficiently exciting motions" is a recurring phrase in almost all expositions of self-calibration in the literature [70]. While there has been work on determining the observability of different motions [70], the full observability of the calibration parameters may not be guaranteed for an arbitrary measurement sequence. This renders the calibration procedure non-trivial for an inexperienced operator, especially in non-holonomic platforms such as ground vehicles; it is simply not obvious how to physically move the platform so as to collect measurements that allow the desired parameters to be inferred.

In the context of life-long autonomous operation, self-calibrating systems are a necessity since they allow for compensation of errors induced over time, such as after a collision or due to sensor changes. Robust self-calibration also allows for using the same algorithm on multiple platforms and foregoing the offline calibration routine entirely. The downside of self-calibration is that it considerably increases the dimensionality of the state space while providing no additional measurements. Thus the task of collecting measurements that render the full state-space observable



(a) Suggested motions



(b) Extrinsics calibration with suggested motions

Figure 5.1: (a) Motions suggested to an inexperienced operator for camera-to-IMU extrinsic calibration, drawn from the learned policy. (b) Convergence of translation and rotation over the motions suggested in (a). Note how the first suggested movements provide little to no information on translation, but allow rotation to converge. This follows our practical knowledge that estimating rotation first improves convergence on translation.

is non-trivial. In practice this means that self-calibrating systems require specific motions that are executed by an expert operator who excites the platform until the desired states have converged to acceptable values, or when no operator is available, hoping the platform will undergo sufficient excitation so as to render the states observable. This problem can be addressed in two ways:

1 **Optimization**: optimize over trajectories in order to generate motions which render observable parameters.

2 **Learning**: determine which motions render the system observable through experiences.

The first option tackles the problem by optimizing over some measure of observability [51], such as the condition number of the linearized system, in order to produce trajectories that maximize the observability of the state space. While appealing in its elegance, this requires knowledge of the motion model of the platform, and can be computationally demanding. The second option, which is the approach taken in this chapter, forgoes the process model and uses a model-free reinforcement learning technique to learn which sequence of motions render the desired states observable. The appeal of this approach is that it can be applied to any platform, holonomic or non-holonomic, without the simplifying assumption of a process and environment model. A use case is a non-expert operator needing to calibrate a rig (physical assembly with sensors mounted) with a camera and an IMU (Inertial Measurement Unit). In currently-available calibration libraries [8] the user is tasked with waving the rig around until the calibration parameters are obtained with satisfactory uncertainty. Our approach allows for the system to learn which sequences of motions contain useful segments that render the desired state-space observable. Once the informative sequence of motions is learned, these can be suggested to any user, removing the random "hope-that-this-motion-is-sufficiently-exciting" approach in favor of a series of suggested motions that will obtain the desired calibration parameters deterministically.

Unsurprisingly, self-calibration has received considerable attention in the robotics community, and has proven to be a hard problem due to the slow time-varying nature (drift over time) and inference over naturally occurring data. The first of these problems has been addressed in part by existing self-calibrating algorithms [115], the second gives rise to a series of problems that are less commonly considered:

1 **Observability during normal operation**: normal operation may not render calibration parameters observable if e.g. two cameras do not share an overlapping field of view. In this case, planar motion renders the problem of finding the camera-to-camera extrinsic parameter degenerate.

2 **Parameters that appear observable, but in fact are not were noise not present**: related to the general observability of calibration parameters, it is possible that a solution is numerically obtained even in degenerate cases due to noisy measurements, leading to a physically uninterpretable solution. An example of this is a rank-deficient matrix which is made numerically full-rank due to noise. While a solution is possible, it has no physical meaning. This is rarely addressed in calibration systems.

3 **Which motions will render unobservable dimensions in parameter space observable**: calibration experiments are usually hand-engineered to guarantee that all parameters become observable, especially for platforms for which a process model is not available or desirable, this makes calibration a very tedious and error-prone activity for the average operator, since it is not obvious how the sensor has to be excited.

Existing algorithms handle (1) by hoping sufficiently exciting motions are provided. Some work has been done to address (2) but it is largely unexplored in most calibration systems. To the best of our knowledge no published self-calibration algorithm is able to cope with issue (3) without requiring a process model, some work on this direction has been done by [123] but it is limited to camera intrinsics and our approach does not require the user to follow the given instructions.

In this chapter we propose an algorithm to deal with all of the presented difficulties in calibration. Observability is handled by exploiting the link between the Fischer Information matrix and nonlinear observability [60]. Treating numerically unobservable parameters is performed by detecting directions in the parameter space that are unobservable through singular value thresholding of the scaled information matrix and avoiding updating the current state estimate in those directions. Deciding which motions will generate measurements that allow for inference over the desired parameters is done by incorporating reinforcement learning for empirically learning which sequence of motions maximizes the inference of the desired parameters.

The calibration procedure is modeled as a Markov Decision Process (MDP) and Q-learning is employed to learn the optimal action-selection policy. The action-space is discretized into motions that can be easily performed by a human operator, and the state-space is defined as the combination of possible parameter states. For example, the desired final state is when every direction of the parameter space can be inferred, intermediate states are composed of the subsets of the parameter space that may be independently observed—when calibrating camera-to-IMU extrinsics, a set of measurements may render the rotation observable, but not the translation, then another set of measurements may provide information on the translation (see Figure 5.1). Thus the only requirement of our system is that the sensors be equipped on a platform that is **capable** of performing motions that render the parameters observable, and that the human-operator is able to loosely follow instructions on moving the platform. The main question we aim to answer is: can a MDP with delayed reward regress a convergent policy for the calibration problem? The remainder of this chapter will cover related work in Section 5.2, the mathematical background in Section 5.3, the theoretical foundation of our method is explained in Section 5.4, Section 5.5 validates our approach with simulated and real-world experiments. Finally Section 5.6 provides a discussion on our findings.

5.2 Related Work

The use of a known calibration pattern such as a checkerboard coupled with nonlinear regression has become the most popular method for camera calibration in computer vision during the last decade; it has been deployed both for intrinsic camera calibration [140] and extrinsic calibration between heterogeneous sensors [151]. While being relatively efficient, this procedure still requires expert knowledge to reach a discerning level of accuracy. It can also be quite inconvenient on a mobile platform requiring frequent recalibration (e.g experimental platforms which undergo constant sensor changes). In an effort to automate the process in the context of mobile robotics, several authors have included the calibration problem in a state-space estimation framework, either with filtering [98] or smoothing [79] techniques. Filtering techniques based on the Kalman filter are appealing due to their inherently online nature. However, in case of nonlinear systems, smoothing techniques based on iterative optimization can be superior in terms of accuracy [139].

Our approach does not rely on formal observability analyses to identify degenerate paths of the calibration run as in [16], since these approaches still expect non-degenerate excitations.

A last class of methods relies on an energy function to be minimized. For instance, Levinson and Thrun [85] have defined an energy function based on surfaces and Sheehan et al. [134] on an information theoretic quantity measuring point cloud quality.

Despite considerable work in the field of calibration, very little is known regarding how to efficiently and actively deal with degenerate cases frequently occurring during a calibration routine. The current state of the literature frequently assumes that optimization routines are executed on well-behaved data. As demonstrated in the next sections, this can be a critically flawed assumption in real-world scenarios.

5.3 Problem Formulation

In the following exposition, we borrow the formalism of the probabilistic discrete-time Simultaneous Localization and Mapping (SLAM) model [34]. For the sake of clarity, we consider here a robot with a single camera observing a known number of landmarks at each timestep and a single inertial measurement unit sampling linear acceleration and angular velocity. A front-end inspired by [83] is tasked with establishing correspondences between sensor's measurements and landmarks.

5.4 Methodology

Let $\mathcal{X} = \{\mathbf{x}_{0:K}\}$ be a set of latent random variables (LRV) representing robot states up to timestep K, $\mathcal{L} = \{\mathbf{l}_{1:N}\}$ a set of LRV representing N landmark positions, $\mathcal{Z} = \{\mathbf{z}_{1_{1:N}:K_{1:N}}\}$ a set of LRV representing $K \times N$ landmark measurements, and Θ an LRV representing the calibration parameters of the robot's sensor. The goal of the calibration procedure is to compute the posterior marginal distribution of Θ given all the measurements up to timestep K,

$$p(\Theta|\mathcal{Z}) = \int_{\mathcal{X},\mathcal{L}} p(\Theta, \mathcal{X}, \mathcal{L}|\mathcal{Z}). \tag{5.1}$$

The full joint posterior on the right-hand side may be factorized into:

$$p(\Theta, \mathcal{X}, \mathcal{L}|\mathcal{Z}) \propto p(\Theta, \mathbf{x}_0, \mathcal{L}) \prod_{k=1}^{K} p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_k) \prod_{k=1}^{K}\prod_{i=1}^{N} p(\mathbf{z}_{k_i}|\mathbf{x}_k, \mathbf{l}_i, \Theta). \tag{5.2}$$

By making the common assumption that Eq. (5.2) is normally distributed with mean $\mu_{\Theta\mathcal{X}\mathcal{L}}$ and covariance $\Sigma_{\Theta\mathcal{X}\mathcal{L}}$ we can derive a Maximum a Posteriori (MAP) solution for the mean and covariance,

$$\hat{\mu}_{\Theta\mathcal{X}\mathcal{L}} = \arg\max_{\Theta\mathcal{X}\mathcal{L}} p(\Theta, \mathcal{X}, \mathcal{L}|\mathcal{Z}) = \arg\min_{\Theta\mathcal{X}\mathcal{L}} -\log p(\Theta, \mathcal{X}, \mathcal{L}|\mathcal{Z}); \tag{5.3}$$

we further define our model by defining an observation model $\mathbf{z}_{k_i} = \mathbf{g}(\mathbf{x}_k, \mathbf{l}_i, \Theta, \mathbf{n}_k)$ where $\mathbf{n}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{N}_k)$ is a normally distributed observation noise variable, with known covariance $\mathbf{N}_k$. Given that the observation model is usually nonlinear, such as a camera projection with lens distortion, we resort to a nonlinear least squares method that iteratively solve a linearized version of the problem. We employ the *Gauss-Newton* algorithm for this purpose.

Directly from Eq. (7.1) and the assumption of normally distributed measurements, we can turn the MAP problem into the minimization of a sum of squared error terms. This is covered in detail in [34], so we will briefly cover only the aspects that are relevant to our approach. The Gauss-Newton method only requires the Jacobian matrix of error terms, $\mathbf{J}$. In block matrix form the update is

$$(\mathbf{J}^T\mathbf{G}^{-1}\mathbf{J})\delta\hat{\mu}_{\Theta\mathcal{X}\mathcal{L}} = -\mathbf{J}^T\mathbf{G}^{-1}\mathbf{r}(\hat{\mu}_{\Theta\mathcal{X}\mathcal{L}}), \tag{5.4}$$

where $\mathbf{G}$ is the error covariance matrix built from diagonal blocks of $\mathbf{N}_k$ and $\mathbf{r}(\hat{\mu}_{\Theta\mathcal{X}\mathcal{L}})$ is the error evaluated at the current state estimate. At convergence the quantity $\mathbf{J}^T\mathbf{G}^{-1}\mathbf{J}$ is the *Fischer Information Matrix* (FIM) and also the inverse of the estimate covariance matrix, $\hat{\Sigma}_{\Theta\mathcal{X}\mathcal{L}}$. This is a key aspect of this work, since as will be described below, the numerical rank of the FIM provides information about the numerical observability of the parameters for a given batch of data. Specifically for computing the posterior distribution Θ we use the cost terms and Jacobians for both camera intrinsics and camera-to-IMU extrinsics as defined in [115, 66].

5.4.1 Observability

There exists a solution to Eq. (7.4) iff the FIM is invertible, i.e. it is of full rank. The link between the rank of the FIM and observability of the parameters being estimated is well established in [60]. A singular FIM corresponds to some unobservable directions in the parameter space given the current set of observations. Classical observability analysis, for example the method of Hermann and Krener [52], proves structural observability—that there exists some dataset for which the parameters are observable—but it does not guarantee that the parameters are observable for any dataset.

Using singular-value decomposition (SVD) on the FIM we can identify a numerically rank-deficient matrix by analyzing its singular values and consequently the numerical observability of the system [50] [45] [90]. The *numerical rank* r of a matrix is defined as the index of the smallest singular value σ_r which is larger than a pre-defined tolerance ϵ,

$$r = \arg\max_i \sigma_i \geq \epsilon. \tag{5.5}$$

It is important to note that the numerical rank corresponds to the algebraic rank of the unperturbed matrix within a neighborhood defined by the parameter ϵ proportional to the magnitude of the perturbation matrix, which in this case can be interpreted as the noise affecting the measurements. When the noise affecting the matrix entries has the same scale (by using column or row scaling) then the numerical rank can be determined by the singular values.

Specifically we decompose the error covariance matrix $\mathbf{G}$ from Eq. (7.4) into its square root form by using Cholesky decomposition, $\mathbf{G}^{-1} = \mathbf{L}^T\mathbf{L}$, we can re-write Eq (7.4) in standard form:

$$(\mathbf{LJ})^T(\mathbf{LJ})\delta\hat{\mu}_{\Theta\mathcal{X}\mathcal{L}} = -(\mathbf{LJ})^T\mathbf{Lr}(\hat{\mu}_{\Theta\mathcal{X}\mathcal{L}}), \tag{5.6}$$

which are the normal equations for the linear system $(\mathbf{LH})\delta\hat{\mu}_{\Theta\mathcal{X}\mathcal{L}} = -\mathbf{Lr}(\hat{\mu}_{\Theta\mathcal{X}\mathcal{L}})$. Thus we can directly use a *rank-revealing* decomposition to estimate the numerical rank of the FIM, and consequently the numerical observability of the system. Both SVD [46] and QR decomposition [46] are rank-revealing; here we use SVD to demonstrate the method, though we use the more computationally efficient QR decomposition in practice. Let $(\mathbf{LJ})$ be a $m \times n$ matrix with the following SVD decomposition:

$$\mathbf{LJ} = \mathbf{USV^T}, \tag{5.7}$$

where $\mathbf{U}$ is $m \times n$ and orthogonal, $\mathbf{S} = diag(\sigma_1, ..., \sigma_n)$ the singular values and $\mathbf{V}$ an $n \times n$ matrix, also orthogonal. From Eq. (7.6) and the orthogonality of $\mathbf{U}$ and $\mathbf{V}$ we can solve (7.4) as

$$\delta\hat{\mu}_{\Theta\mathcal{X}\mathcal{L}} = -\mathbf{VS^{-1}ULr}(\hat{\mu}_{\Theta\mathcal{X}\mathcal{L}}) \tag{5.8}$$

In order to only update the observable directions of the parameter space, we apply the truncated SVD (or truncated QR decomposition) which establishes the rank of the system by analyzing its singular values [50], only using the first r rows of $\mathbf{S}$, as defined in Eq. (5.5). This allows us to only update the observable directions of the parameter space and maintain the other directions at their initial value. Establishing the value of ϵ to use is specific to the amount of noise expected in the measurements and is treated in Section 5.5. Using the method described in this section we are able to identify which subset of the calibration parameters are observable, and when the full parameters space is observable, which is a key element of the algorithm described in the following section.

5.4.2 Learning Motions

The methodology described up to this point allows us to optimize for the calibration parameters while identifying unobservable directions in the parameter space. We will now describe using that result in order to learn which sequence of motions lead to the regression of the full parameter space.

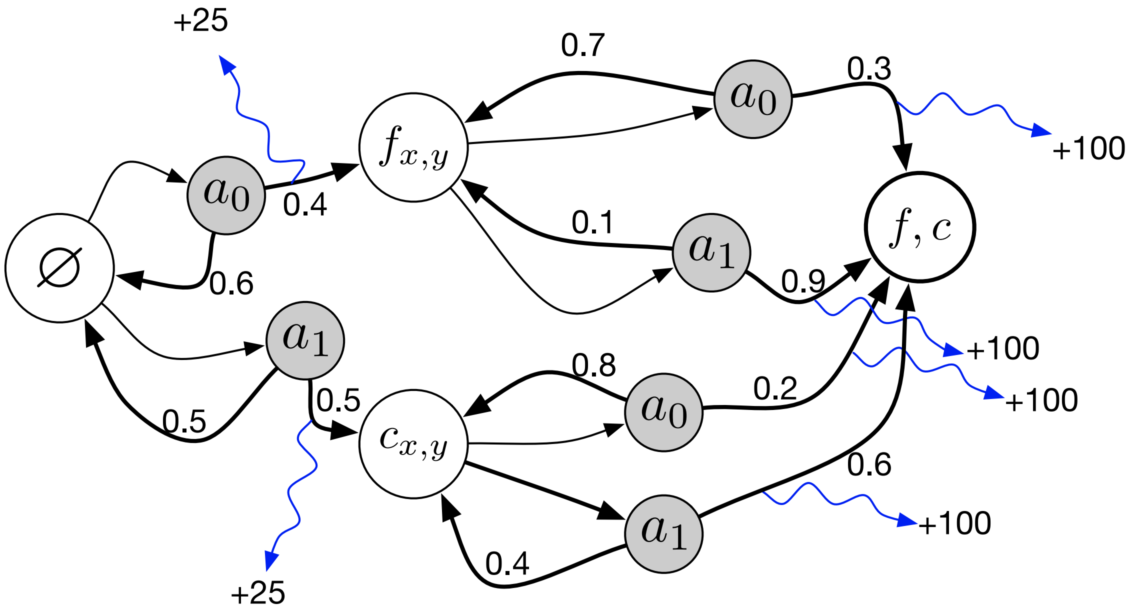


Figure 5.2: Example of discretized state space and actions for camera intrinsic calibration (pinhole camera model). Dark circles correspond to possible stochastic actions $\{a_1, a_2\}$. Blue arrows indicate a reward for that transition. We have grouped $\{f_x, f_y\}$ and $\{c_x, c_y\}$ for simplicity. All possible transitions are not shown for readability.

We consider a robot moving through space (or manipulated by an operator) as an *agent* situated in some feature-rich *environment* comprised of both the position of the robot in the environment and the latent calibration variables. The agent can perform certain actions in the environment (e.g. move to a new location, make new measurements). These actions may result in a *reward* (e.g. information on a latent variable). Actions can thus transform the environment (e.g. new configuration of latent variables) and lead to a new state, which the agent can perform another action on, and so forth. The rules for how to chose which action to take given the current state is called a *policy*, which must consider the stochasticity of the environment (e.g. the choice of action, such as moving the robot forward, may or may not obtain information about a latent variable depending on the structure of the world). The set of states and actions along with the rules for transitioning make up a Markov decision process, and one *episode* of this process corresponds to a sequence of state, action, rewards. The episode ends when the *terminal* state is reached.

The calibration problem can be interpreted as an MDP in which each state represents the regression of each direction of the calibration parameter space (see Figure 5.2), and the actions are a discretized set of movements that can be performed by the operator. Drawing a parallel to a game, we wish to discover the optimal policy, and therefore sequence of actions, for reaching our final state taking into account not only the immediate reward of an action (i.e. learning about a specific parameter) but the future rewards. To illustrate how this applies to calibration, consider a simplified example: If the robot knows nothing about its camera-to-IMU calibration, but it knows that given a certain movement it will learn the camera-IMU translation along the x direction, and given another movement, it will learn the rotation about the x-axis. Note that we assume here that no feasible movement will learn both simultaneously. From an immediate reward standpoint it may seem arbitrary, but by first regressing rotation we are able to reach the full final calibration in fewer movements. We wish to learn the optimal policy for calibration. The state discretization is based on the choice of ϵ as described in Section 5.4.1 which is the threshold that determines the rank deficiency and therefore the observable directions of the parameter space. We have empirically set $\epsilon = 0.015$, however it does need to be adjusted if we were to use a system with a different noise profile.

Q-Learning is well suited for the class of problem. Briefly, within reinforcement learning, Q-Learning is a model-free technique which can be used to find an optimal action-selection policy for a finite MDP. The basic principle is to maximize the discounted future reward. The discounted aspect is due to the stochastic nature of the environment and this to down-weigh the uncertain future rewards. This method essentially consists of establishing a discrete set of actions (Section 5.4.2.1) and states, a reward table for state transitions, and a Q-table which contains one row for each state and a column for each possible action, which encodes the "quality" of a certain action in a given state. Given our finite state space and discretized action space we are able to iteratively approximate the Q function using the *Bellman equation*:

$$Q(s,a) = r + \gamma max_{a'} Q(s',a'), \tag{5.9}$$

Which is the reward for the state transition plus the maximum possible future reward for the next state. The idea behind Q-learning is that we can iteratively converge on the Q-table using the Bellman equation, see Algorithm 1. In practice we also use a learning rate parameter $\alpha = 0.1$ which essentially limits the *step size* for each iteration. Once the Q-table has converged we can simply follow the learned policy π, given the current state: $\pi(s) = \arg\max_a Q(s,a)$ and suggest that action to the user.

Algorithm 1: Observability-aware calibration Q-Learning

```
Data: sensor measurements, $\gamma$, $\alpha$, Reward matrix
Result: converged Q-table
Initialize Q-table $[states, actions] \leftarrow 0$;
while Q-table not converged do
    select random initial state;
    while goal state not reached do
        select possible action, $a$, given current state;
        display action $a$ to be carried out to user;
        compute $\delta\hat{\mu}_{\Theta\mathcal{X}\mathcal{L}} \leftarrow -\mathbf{V}\mathbf{S}^{-1}\mathbf{U}\mathbf{L}\mathbf{r}(\hat{\mu}_{\Theta\mathcal{X}\mathcal{L}})$ according to (7.7);
        compute observable directions according to 5.5;
        if at least one calibration direction is observable then
            go to corresponding state $s'$ and observe reward $r$;
        else
            stay in current state and set $r \leftarrow 0$;
        end
        update Q-table: $Q[s,a] \leftarrow Q[s,a] + \alpha\,(r + \gamma max_{a'} Q[s',a'] - Q[s,a])$;
    end
end
```

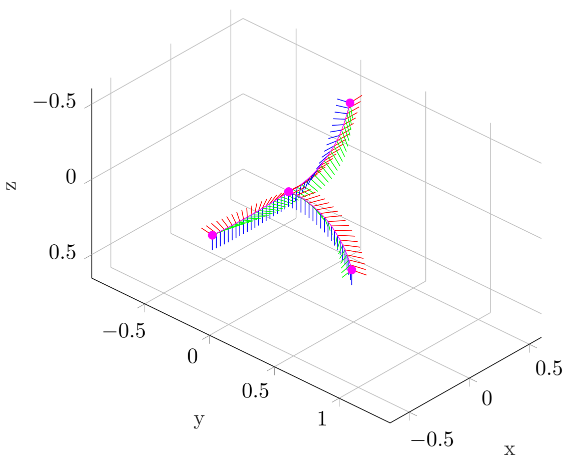
We apply this to both regressing camera intrinsics and extrinsics. Note that the important data association, outlier rejection, residual and Jacobian calculation steps are delegated to a sparse visual-inertial keyframe-based SLAM system which we will not go into detail here. In both intrinsic and extrinsic cases the reward table is straightforward: a reward of 25 is given for non-final state transitions and a reward of 100 is given for any transition to the final state (full calibration). Figure 5.2 shows a simple example where only two actions are possible, and the blue arrows indicate the reward value for that state transition. These values were obtained empirically. We found that using a constant $\alpha = 0.1$ works well, although experimenting with variable step sizes could lead to quicker convergence. We initialize the Q-table to zero and use $\gamma = 0.9$. We experimented with a variable discount factor [40] but found no practical benefit. The outer loop is executed until the convergence criteria is met. We use a relative change metric to quit the learning process, with a maximum number of iterations $max\_iter = 1000$.

5.4.2.1 Action Discretization

Given that this work is focused on providing intuitive and simple calibration instructions to a inexperienced operator, we discretize the initially continuous action space (possible movements performed by the rig) into a set of movements that can be easily conveyed and performed by the operator. This consists of translation along one degree of freedom combined with rotation around a single axis. Empirically we found this to be the optimal trade-off for basis motions that are both sufficiently informative and simple enough for the operator to execute. Degenerate motions such as pure rotation or translation were not included. This results in a tractable set of only 18 actions. The motions are also limited to the range of motion of the average human arm, between 60 and 80cm. Given the goal of providing easy-to-understand motion suggestions we chose simpler basis motions which may lead to more actions being performed (i.e. a more complex motion could contain more information than the simple motions suggested here, but would be harder to execute). Figure 5.3 shows a few learned motions from the discrete set of actions. The motion discretization described requires that the platform execute those motions when training. The training procedure consists of either simulations, as shown in section 5.5.2 or the the rig being moved around the environment (either by an operator or autonomously) in the latter case we suggest the discretized motions to be executed by the operator.

5.5 Experiments

We evaluate the proposed framework on several fronts. First we describe results from our simulated experiments, using synthetic visual and inertial data to train our system and evaluate the performance of calibration with the synthetically trained motions vs. trained with real data.



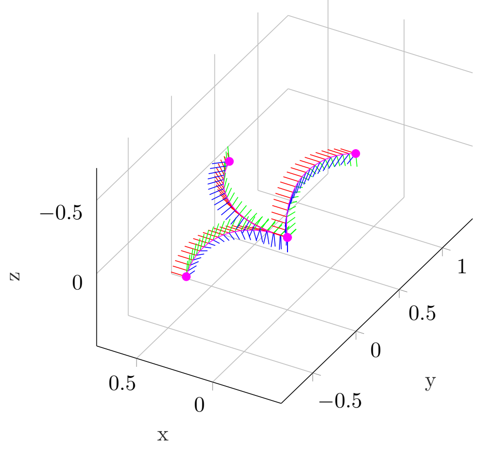


Figure 5.3: Left image shows learned policy for regressing camera intrinsics for a radial-tangential distortion model, with a narrow field of view lens ($f = 460$px) and central point in the middle of the image. Changing to a fisheye lens and a much wider field of view ($f = 220$px) changes the learned optimal action sequence (right figure). Each graph depicts the three suggested motions for that camera.

We then demonstrate that the system can be successfully applied to both camera intrinsics and camera-to-IMU extrinsics, and that our method performs equally or better than publicly available calibration tools. Finally we demonstrate that this system can be used for life-long learning by adapting to changes in hardware configuration that impact the calibration routine.

5.5.1 Experimental Setup

The proposed method was implemented in C++ and integrated into our existing sparse keyframe-based visual-inertial SLAM pipeline which uses *BRISK* [?] feature descriptors and ceres-solver [?] as the non-linear solver. In order to evaluate the proposed method, experiments were run on a sensor platform known as "rig." The rig was equipped with a monocular camera and a commercial grade MEMS-based IMU. The camera is equipped with a wide field-of-view lens at 2040×1080 resolution downsampled to 640×480 coupled with a MEMS IMU, sampled at 200Hz. The camera captures images at 30 frames per second.

In order to generate simulated measurements visual and inertial data corresponding to each of the 18 basis motions was generated. A differentiable quaternion spline through the SO(3) element of each pose was used to obtain smooth gyroscope measurements, and a cubic spline was run through the euclidean positions for accelerometer measurements. Simulated visual data was generated by creating a virtual world in OpenGL and simulating movement corresponding to the accelerometer measurements. Visual data was captured at 15fps with IMU updates at 100Hz. For each projected feature point from the simulated images (640×480 resolution), independent zero-mean Gaussian noise with $\sigma = 0.5$ was added to the (u, v) pixel coordinates. Zero-mean Gaussian noise with $\sigma = 10^{-3}$ was also added to the IMU accelerometer and gyroscope measurements and biases.

5.5.2 Simulations

We generate noisy simulated measurements corresponding to the 18 basis functions and run through the training procedure in Algorithm 1 until the relative change in values for the Q-table is $< 1e^{-3}$.

An experiment was run to determine if different calibration parameters would suggest different motions. To that end we generated synthetic data to regress the Q-table for narrow (radial-tangential distortion) and wide field of view fisheye (fov distortion) camera, which resulted in three considerably distinct set motions (see Figure5.3). We then used the suggested motions to execute the calibration procedure with simulated measurements for both cameras. Using the suggested motions corresponding to the camera which was used for training yielded on average a 22% better mean and 8% lower variance, suggesting a correlation between the learned motions and the camera distortion model and field of view.

5.5.3 Real-world Data

Experiments with the rig described in Section 5.5 require an interface for suggesting motions to the user. Due to the simplicity of the motions we resort to a textual representation, indicating what direction the rig should be moved in and rotation about which axis. A graphical interface with visual feedback is in development. The main objectives of this experiment are to verify that an inexperienced user can use the suggested motions to easily and reliably calibrate a robotic

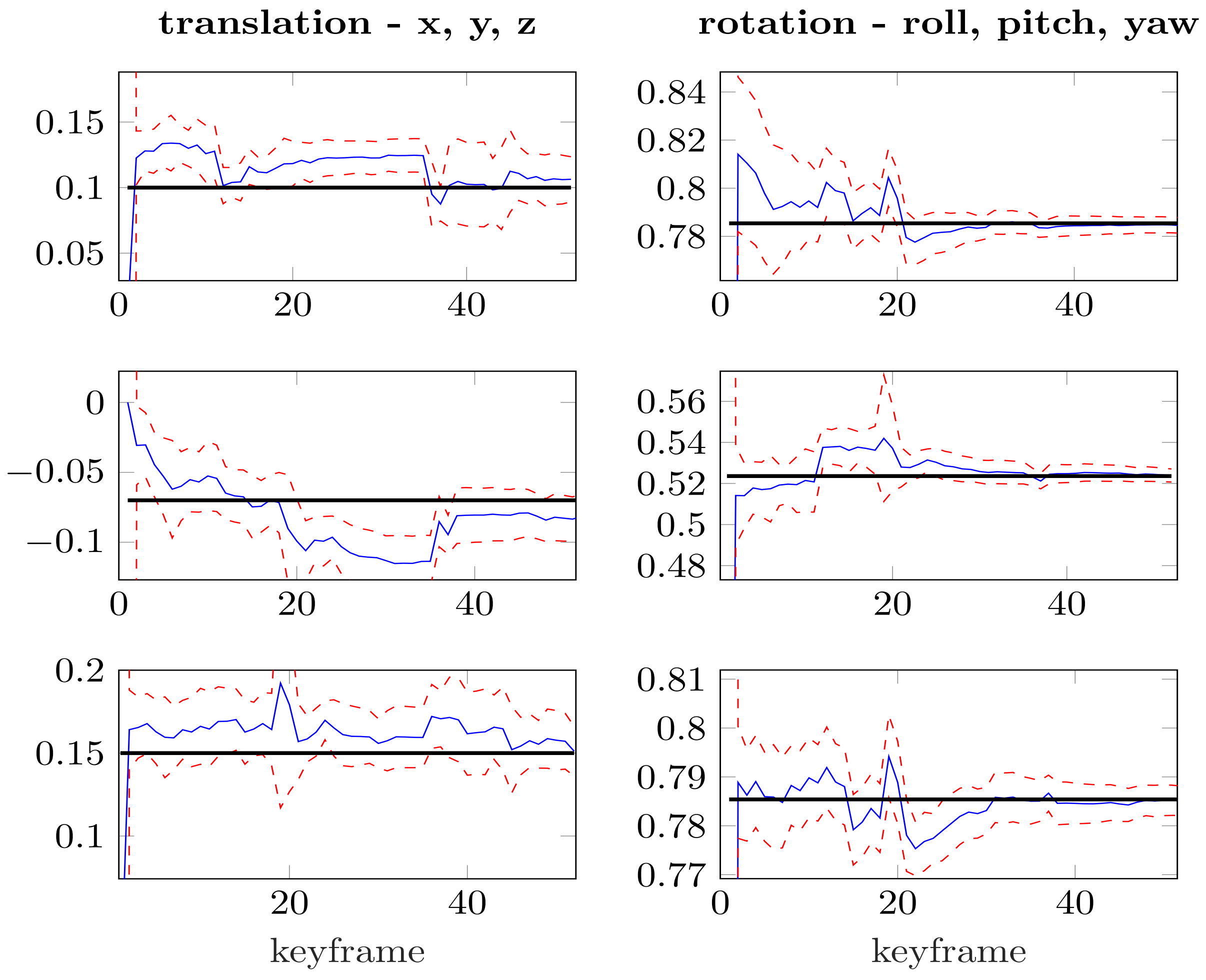


Figure 5.4: Convergence of extrinsic parameters by an inexperienced operator following suggested motions. The solid black line represents the ground truth as obtained by a batch optimization with a target-based routine, the blue line represents the estimate as the rig is moved and the dotted red line the 3σ bounds. The jumps in values correspond to the moment a new segment is processed. A total of four motions were suggested for this calibration run.

system and obtain lower variance compared to a publicly available standard calibration pipeline. To that end we manually trained algorithm, then calibrated the same camera with a fisheye lens by following the motions suggested by our system and also by waving a calibration target in front of the camera and using vicalib [8], a publicly available calibration library, to optimize over the camera intrinsics. This experiment was repeated 50 times for each calibration method in order to obtain statistically significant results. The users selected to perform the calibration ranged from lab members to random volunteers that did not have experience calibrating cameras. The results

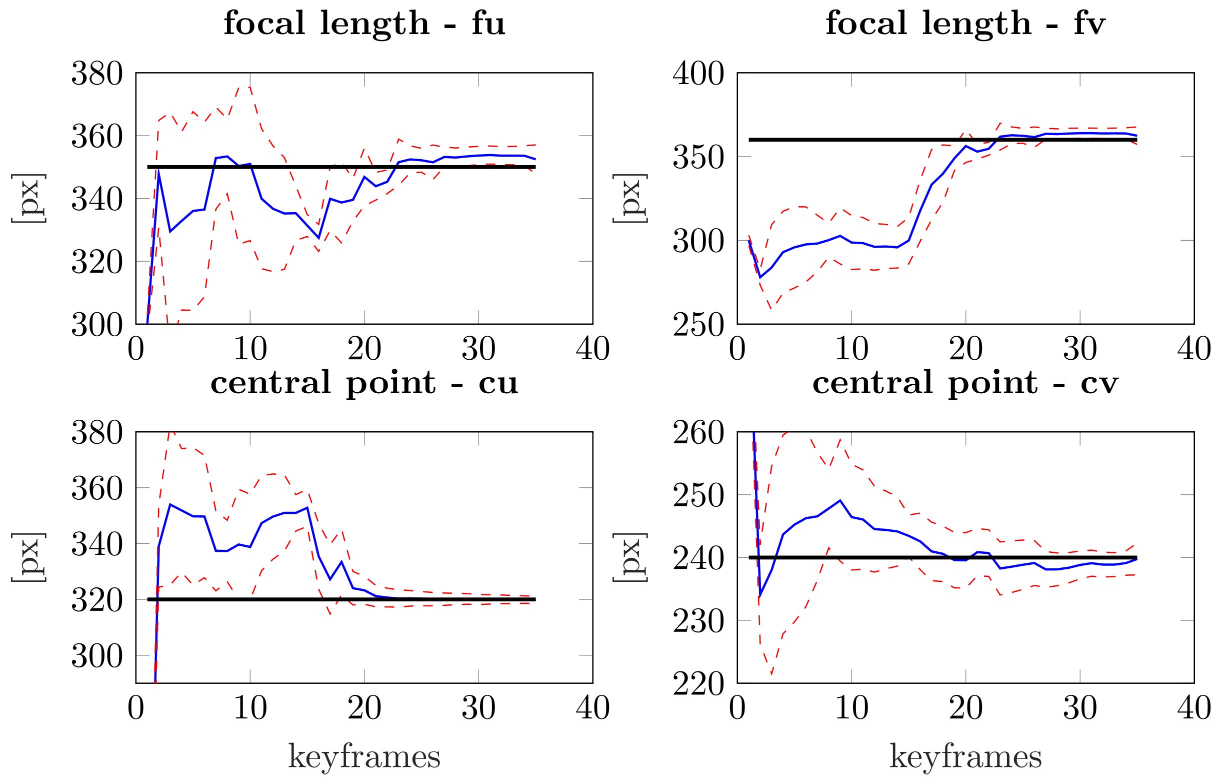
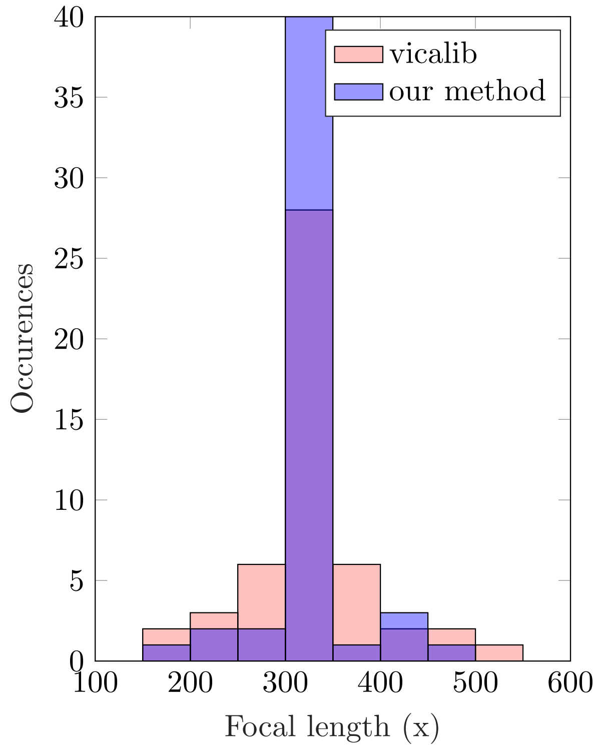


Figure 5.5: Convergence of camera intrinsic calibration parameters with real-data by following motion suggestions. Dotted red line is the 3σ bound, solid black line is the ground truth value.

can be seen in Figure 5.6 which clearly show that both methods converge to the same mean, but the distribution is tighter around our method. Furthermore both extrinsic (Figure 5.4) and intrinsic (Figure 5.5) calibration using motion suggestions converges to within 3σ of the ground truth in as few as 40 keyframes. Finally Figure 5.1 shows an example of a sequence of motions suggested for extrinsics calibration and the corresponding convergence of translation and rotation. It is important to note that a feature-rich environment assumption is made: there must be enough salient features for visual tracking. This is the most common scenario so we are not sacrificing much by forgoing an analysis on the effects of the structure of the environment on calibration.

5.6 Discussion

In this chapter, we have presented a novel calibration tool that uses reinforcement learning to provide live feedback on the state of calibration and produces accurate calibration parameters



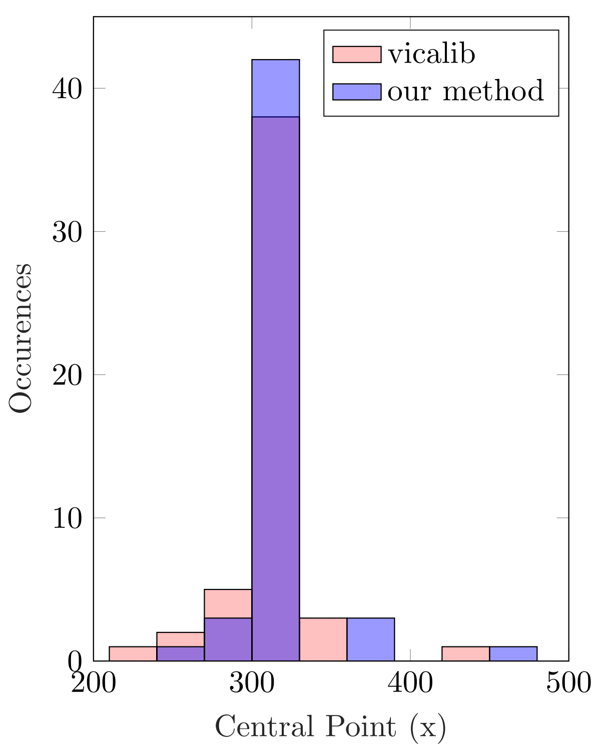


Figure 5.6: Distribution of focal lengths and central point for all trials. The mean parameter values between our method and vicalib are similar (focal length: 332.1px for vicalib and 334.8px for our method), the standard deviations are much higher (32.1 vs. 21.2) indicating more consistent and repeatable results.

even when used by inexperienced operators. We have leveraged truncated SVD/QR decomposition to deal with unobservable directions and reinforcement learning for motion suggestions to perform model-free calibration for both camera intrinsics and camera-to-IMU extrinsics. We have evaluated the proposed system in a variety of scenarios and shown that it can be used as a replacement for currently available calibration toolkits. Our principal question was to assess the suitability of reinforcement learning when applied to the calibration problem. We have shown that through the discretization of both the action and state we are able to leverage Q-learning to improve the calibration experience. An argument could be made that a much larger state space would be possible when using a deep network [103] instead of the Q-table, but we argue that for the calibration problem as stated here, there is little practical advantage. A considerable challenge in the proposed framework is designing the human interface so that the user will follow the instructed motions. This is an area in which warrants further development, however in the projected fully autonomous mode where motion suggestions are the input to a controller, the human interface can be removed from the pipeline. An interesting result that lends itself to long term autonomy is the ability to capture different sensor configurations. As shown in Figure 5.3 having radically different calibration parameters results in a different set of optimal motions, reinforcing the point that the “SLAM wobble” or any pre-determined calibration maneuver is not guaranteed to provide acceptable parameter inference. If a robot is expected to calibrate itself autonomously it cannot have a pre-determined routine or hope that random navigation will render its calibration parameters observable. A system that is robust to sensor configuration changes by re-learning how to calibrate itself is a step in the direction of both “power-on-and-go” robotics and long term autonomy.

Chapter 6

Online Probabilistic Change Detection in Feature-Based Maps

6.1 Problem Statement

In the context of mobile robots and autonomous driving, accurate high resolution *spatial awareness* is necessary for successfully navigating an environment. This can be cast as a *data fusion* problem in which noisy measurements from sensors undergoing uncertain dynamic motions must be combined into a single underlying state estimate. This problem is typically addressed through a procedure known as SLAM. Some applications, such as autonomous driving, require high fidelity state estimates which need to be robust to sensor and environmental changes. Current SLAM algorithms can be fragile in two aspects: algorithmic foundations and hardware robustness. The former includes failure modes induced by limitations in current SLAM algorithms (i.e. difficulty handling dynamic environments); the latter includes failures due to sensor degradation. We focus on one of these algorithmic limitations: the reliance on static maps. In general, one cannot make simplifying assumptions such as the existence of a static world on arbitrary real-world environments; stop signs are removed, buildings change appearance, and road construction is pervasive. Explicitly addressing this failure mode is critical for safe long-term operation.

It is impossible to talk about algorithmic failures without mentioning data association. Data association matches each measurement to the portion of the state the measurement refers to (e.g. associating a visual feature to a specific landmark in visual SLAM). Incorrect associations can quickly cause the SLAM estimate to diverge [109]. This is especially critical in feature-based maps which lack a notion of appearance (i.e visual or structural descriptor).

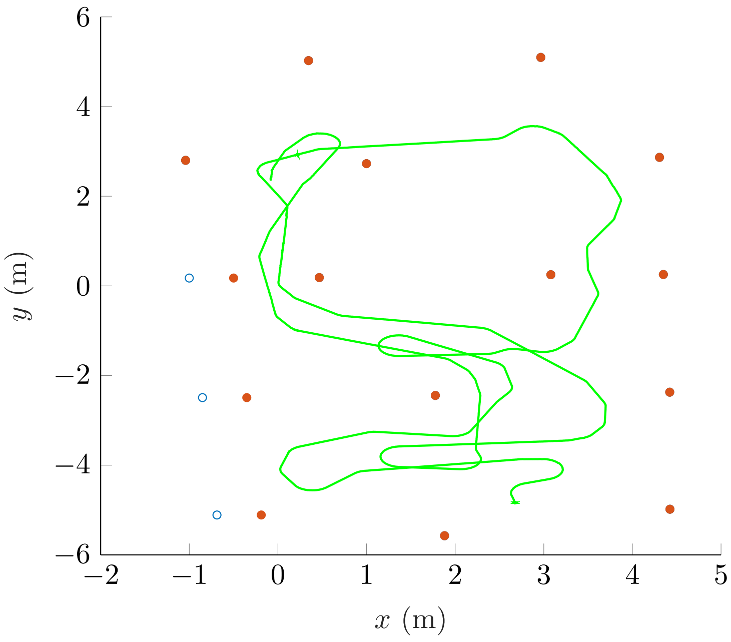


Figure 6.1: Localizing against a stale map: the green trajectory shows the estimated poses from range/bearing and odometry measurements, given a prior map which is no longer current (circles represent incorrect map elements).

In the static-world case, perceptual aliasing makes data association a challenging problem; this problem is worsened by the presence of unmodeled dynamics in the environment, which include both short-term and seasonal changes. It is fairly common for current SLAM approaches to make a *static world assumption*, which holds true for independent, short mapping runs in small-scale environments. However, when mapping in large environments over long periods of time, change is inevitable.

Change in the environment can be especially difficult to detect when dealing with an increased amount of clutter or when the change is subtle. For example, lane markings may be re-painted a few centimeters from the original position due to construction, or traffic signs may be slightly re-located. These scenarios are especially susceptible to incorrect data associations resulting in localization error. We present a robust solution to detecting changes in feature-based maps which leverages information about both single features and neighboring features to produce a globally consistent belief over individual and joint feature persistence. Figure 6.1 shows an example of

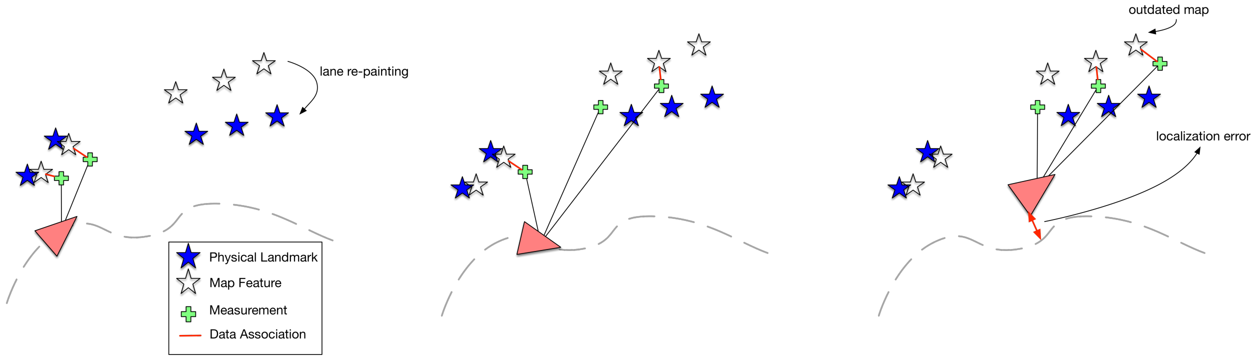


Figure 6.2: Graphical example of the problem: in the left figure, the measurements are correctly associated to map features; in the middle figure, some landmarks have moved, for example lane markings that were re-painted (blue stars), and the map is outdated (white stars). Here, one measurement (green cross) is correctly identified as a new measurement, while the other is incorrectly associated to the outdated map feature due to landmark clutter, pose and measurement uncertainties. In the right figure, the incorrect data association causes the least-squares solution to converge to an incorrect estimate. Our proposed solution leverages the correlation between the three landmarks which moved to estimate the joint feature persistence and improve data associations.

robust localization against a stale map: the localization estimate remains consistent even though some elements of the map have moved.

We extend the work in [126], which introduced the notion of a Bayesian filter to model feature persistence in a time-varying feature-based environmental model. We differ from [126] by proposing a general formulation for persistence which takes into account correlation between features. We focus on sparse feature based maps with no assumptions on sensor-specific feature descriptors (i.e visual descriptors) which is broadly applicable to any sparse feature-based representation. We are interested in estimating the existence of each feature in the map and bounding the localization error due to incorrect data associations, we show that by capturing the underlying structure of the environment we are able to make better informed decisions on data associations. In summary, we propose a novel joint probabilistic formulation over feature persistence. We show that a joint formulation over feature persistence can be made informative by imposing or learning the structure of the environment. We further show that the joint and marginal persistence estimates are amenable to constant-time operation. Finally, we demonstrate that by incorporating the joint belief over feature persistence in the data-association step, we are able to perform robust localization even in the presence of hard-to-detect changes (e.g. small changes). We demonstrate the benefits of estimating map persistence in a graph-SLAM [30, 64] implementation, potentially enabling long-term autonomous applications which are robust to arbitrarily small map changes.

6.2 Prior Work

The challenge of dealing with dynamic and semi-static environments is a recurring problem in the robotics community and has been addressed from multiple fronts. The principal challenges with semi-static environments are the need to *detect* a change in the environment and *update* an existing map so that it reflects the most current state of the world. One way to tackle this problem is to use environmental representations that are suited for dynamic environments. One such representation is the seminal work by Biber and Duckett [13, 12] who update a sparse map built from 2D laser scans by randomly selecting a fixed fraction of the scans every revisit to update the prior map. Morris et al. presents a multiple-map approach [105] where many map instances are stored and the one best fitting the current set of sensor measurements is used, an approach suitable for environments with a discrete set of possible configurations. Other approaches [20] model each "place" as a set of experiences which has proven to be robust to drastic seasonal changes. The main difference between these methods and what we propose is that while one targets *localization* we are interested in producing a geometrically and temporally consistent representation of the environment suitable for continued localization.

Other approaches [146, 73, 74] are capable of recovering a geometrically-consistent map robust to dynamic environments. However these solutions are tailored to specific sensor modalities, such as the work in [74] which models places as a collection of camera images, connected by 6-DOF transformations between camera poses. Another example is [146], which proposes the *Dynamic Pose Graph SLAM* but limits its use to 2D laser scanners and lacks an underlying probabilistic model for reasoning about change while abstracting the sensor modalities.

The Occupancy Grid is also a common choice for environmental representation, as proposed by Meyer-Delius et al. [101]. This work uses a *dynamic occupancy grid* which adapts the classic occupancy grids [104] to dynamic environments by modeling each cell as a stationary two-state Markov process. Other works such as [127] also propose a form of dynamic occupancy grids; [143] incorporated the occupancy grid model into a particle filtering framework for a Bayesian model-based mapping solution. These methods have in common the restriction of representing the environment by its volumetric geometry, which limits their accuracy in cases of interest such as when using visual appearance information for mapping.

Recently there has been some work on semi-static feature-abstracted environments such as Krajnk et al. [77] with Fourier analysis for predicting future states but that is designed for *periodic* activity. The work most similar to this chapter was presented by Rosen et al. [126] which proposes an information-theoretic formulation for feature persistence taking into account sensor errors. However, this approach ignores potential correlations between feature persistence by assuming each feature persistence is marginally independent. This is not validated with any real data; furthermore, the impact of the persistence model on improving the map and data associations is not addressed. In contrast to these prior works, we present a general unified formulation for feature persistence that captures potential correlation between features, while maintaining a tractable posterior for constant-time estimation and showing the necessity of modeling feature persistence jointly for robust localization.

6.3 Methodology

We cast feature persistence estimation within the context of probabilistic SLAM. The quantities of interest that are directly useful for higher-level tasks are the robot's pose in time and the positions of the map features; these quantities will be represented by the state vector $\mathcal{X} = [\mathbf{x_p}, \mathbf{x_{l_{1:M}}}]$ where $\mathbf{x_p}$ are all the poses and $\mathbf{x_l}$ are the M map features. Given a set of sensor measurements $\mathcal{Z}$, the Maximum-a-Posteriori (MAP) SLAM problem is to maximize the posterior $p(\mathcal{X}|\mathcal{Z})$. However this problem as stated is intractable since it requires summing over all possible data associations, an intractably large problem due to its combinatorial nature [28]. Letting J be the vector of all data association hypothesis one might then wish to estimate the optimal data association vector

$\arg\max_J p(J|Z)$; however the difficulty in evaluating the likelihood of a spurious measurement (i.e. the likelihood of seeing a new feature) makes this approach undesirable. The usual solution is to solve for data associations with a search over possible associations, using techniques such as Joint Compatibility Branch and Bound (JCBB) [109] and then condition the state estimate on data associations: $p(\mathcal{X}|\mathcal{Z}, J)$ where J is the vector of data associations that assign a feature in the state vector to a measurement. If we drop the static-world assumption and allow features to have an associated "survival time," an additional set of discrete random variables $\Theta^t \in \{0, 1\}$ which represent if a feature exists at a specified time t. An appealing approach would be to jointly estimate data associations and feature persistence $p(J, \Theta|\mathcal{Z})$, however we run into the same problem of evaluating the likelihood of a spurious measurement. Therefore we take a similar approach to estimating the state vector $\mathcal{X}$ and condition the feature persistence on the output of the data association step $p(\Theta^t|J)$. This is done by first estimating J using a data association technique such as JCBB, which is then used to estimate Θ.

6.3.1 Feature Persistence Model

We follow the survivability formulation introduced in [126], which we will briefly describe here. Each feature i in the map has a latent "survival-time" $T_i \in [0, \infty)$ which represents the time when feature i ceases to exist, as well as a persistence variable Θ_i^t:

$$\begin{aligned} T_i &\sim p_{T_i}(\cdot) \\ \Theta_i^t|T_i &= \begin{cases} 1, & t \le T_i \\ 0, & t > T_i \end{cases}, \end{aligned} \tag{6.1}$$

where Θ_i^t is a boolean random variable representing whether feature i exists at time t, and $p_{T_i}(\cdot)$ encodes some prior distribution over the survival time T_i. We are interested in estimating for each feature i, its marginal persistence probability $p(\Theta_i^t = 1|J^{1:N})$, where $J^{1:N}$ are all the feature detections collected until time t_N. Note that for a map with M features $J^{1:N} = \{J_1^{1:N}, ..., J_M^{1:N}\}$ with $J_k^{1:N} \triangleq \{j_k^{t_i}\}_{i=1}^N$, $k \in [1, M]$, $j \in [0, 1]$. The feature detections are the output of the data-association step, indicating if feature k was detected at time t_i.

6.3.2 Estimating Feature Persistence

We are interested in estimating the full joint feature persistence posterior at a certain time t, given all data association decisions from time t_1 to t_N:

$$p(\Theta^t = 1 | J^{1:N}), \tag{6.2}$$

where Θ^t is the joint persistence over all M map features $\Theta^t \triangleq \{\Theta_1^t, ..., \Theta_M^t\}$ at time $t \in [t_N, \infty)$. This implies that we are interested in estimating the joint posterior probability over feature existence in the present and future, given the sequence of detections for all features. It is important to note that we only estimate the persistence probability for times equal to or greater than the last received measurement t_N. Noting that $p(\Theta^t = 1|J^{1:N}) = p(T \geq t|J^{1:N})$, with $T \triangleq \{T_i\}_{i=1}^M$ the vector of survival times for all M map features and using Bayes' Rule to compute the posterior probability in (6.2)

$$p(\Theta^t = 1|J^{1:N}) = \frac{p(J^{1:N}|T \geq t)p(T \geq t)}{p(J^{1:N})}. \tag{6.3}$$

We will now derive a closed-form expression for evaluating each of the joint posterior terms, starting with the joint detection likelihood $P(J^{1:N}|T \geq t)$. We make the assumption that a sequence of detections $J^{1:N}$ of feature i depend only on the feature i itself; that is, $p(J_i^{1:N}|T \geq t) = p(J_i^{1:N}|T_i \geq t)$

$$\begin{aligned} p(J^{1:N}|T \geq t) &= \prod_{i=1}^{M} p(J_i^{1:N}|T_i \geq t) \\ &= \prod_{i=1}^{m} \prod_{k=1}^{n} p(j_i^{t_k}|T_i \geq t), \end{aligned} \tag{6.4}$$

where we are also making the assumption that the sequence of detections $\{j_i^{t_1}, j_i^{t_2}, \ldots, j_i^{t_N}\}$ for feature i is conditionally independent from other detections, given the persistence Θ_i^t. The intuition behind this is that given existence of a feature, its sequence of detections should not depend on other features. We are still left with evaluating the individual measurement likelihood $P(j_i^t|T_i)$ which is the probability of detecting a feature given its survival time T_i. If data-associations were always perfect this would simply be 1 if $T_i \geq t$ and 0 if $T_i < t$. Since that is not the case, we follow the formulation in [126] and define a *probability of missed detections* P_M and *probability of false alarm* P_F. Using the model defined in (6.1)

$$p(j_i^t|T_i) = \begin{cases} P_M^{(1-j_i^t)}(1-P_M)^{j_i^t}, & T_i \geq t \\ P_F^{j_i^t}(1-P_F)^{(1-j_i^t)}, & T_i < t \end{cases}, \tag{6.5}$$

where P_M models the probability that the feature exists but was not detected, and P_F the probability that the feature no longer exists but was detected, which may happen due to incorrect data associations or spurious measurements. These quantities are dependent on a series of factors such as the amount of clutter in the environment, the data-association process, and therefore the state uncertainty. We present P_M and P_F as constants, but they could be modified per observation, e.g., to include occlusions.

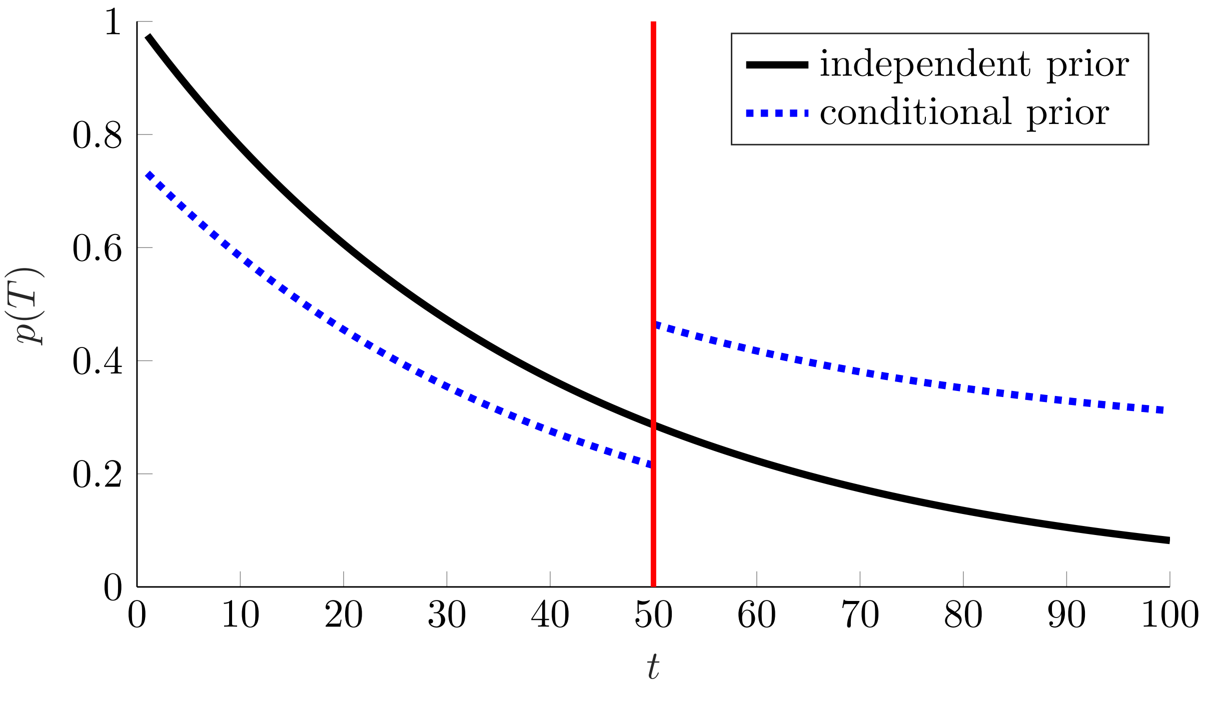


Figure 6.3: $p(T_i)$ (solid black line) is the independent prior on the survival time T_i modeled as an exponential decaying function. Given some correlated feature k with $T_k \geq 50$ we model $p(T|T_k \geq 50)$ (dotted blue line) with weights $\pi_i = 0.75\, \pi_k = 0.25$, which re-enforces our prior belief over T_i if $T_i \geq 50$

The core contribution of this work is in the modeling and evaluation of $p(\Theta^t)$, the joint prior distribution over feature persistence (at time t). In the trivial case each feature is independent and $p(\Theta^t) = \prod_{i=1}^{M} p(\Theta_i^t)$ however we cannot realistically make that independence assumption. In many environments the existence of one feature is clearly correlated to the existence of other features and exploiting that correlation is a crucial aspect estimating jointly consistent persistence. However, tracking the full joint distribution $P(\Theta^t)$ over all features is intractable as it grows 2^M with M map features. However, if we impose some structure to the environment (e.g. a feature drawn from a curb in the road is not affected if a sign-post is removed, however it is strongly correlated to other features in the same curb) the complexity of computing the joint prior is bounded by the maximum number of correlated features. Such structure is justified based on the intuition that the existence of a feature is only strongly correlated to a subset of the map. The formulation proposed in [126] makes the assumption that the persistence for each feature is marginally independent, which is a specific instance which falls out of the general formulation in (6.3). We propose exploiting the underlying structure of the environment to leverage the correlation between features while still maintaining a tractable posterior. Imposing some structure on the feature map such that there exists a set of $L \leq M$ cliques $\{\tau_i\}_{i=1}^{L}$, with $\tau_i^t \subset \Theta^t$ and features $\mathfrak{z} \in \tau$, the joint posterior factors into

$$p(\Theta^t) \approx \prod_{i=1}^{L} p(\tau_i^t). \tag{6.6}$$

Given the assumption that features associated in a clique τ_i have strongly correlated persistence, the joint prior $p(\tau_i^t)$ can be approximately decomposed into:

$$p(\tau_i^t) \approx \prod_{\mathfrak{z} \in \tau} p(\theta_{\mathfrak{z}}^t | \theta_k^t) = \prod_{\mathfrak{z} \in \tau} p(T_{\mathfrak{z}} | T_k \geq t) \; \forall k \in \tau, \tag{6.7}$$

which states that for any feature k in a clique τ_i, the joint distribution $p(\tau_i)$ can be approximately factored into a product of conditional distributions on feature $\mathfrak{z}_k \in \tau_i$. The intuition behind this approximation is that for a set of correlated features (e.g a set of features all drawn from the same rigid body), conditioning on a single feature from that rigid body adds approximately the same amount of information as conditioning on all the features. The conditional prior $p(T_{\mathfrak{z}} | T_k \geq t)$ is defined as

$$p(T_\mathfrak{z}|T_k \geq t) \triangleq \begin{cases} \pi_\mathfrak{z} p(T_\mathfrak{z} < t_i) & t_i < t \\ \\ \pi_\mathfrak{z} p(T_\mathfrak{z} \geq t_i) + \pi_k & t_i \geq t \end{cases}, \tag{6.8}$$

with $\sum_j \pi_j \triangleq 1$ pairwise weights associated to each feature pair in τ_i.

Having computed the likelihood and the prior from (6.3), we are left with computing the marginal measurement probability or evidence $P(J^{1:N})$. We leverage the fact that the detection likelihood is constant in the intervals between detections and the factorization of the full joint posterior into L cliques. Combined with a way to evaluate the cumulative distribution function of the survival time prior $p(T \leq t) \triangleq F_T(t)$, where $F_T(t)$ is the c.d.f. of the survival time prior as described in [126] to derive a closed-form expression for the evidence; defining $t_0 \triangleq 0$ and $t_{N+1} \triangleq \infty$

$$\begin{aligned} p(J^{1:N}) &= \prod_{i=1}^{L} p(J_{\tau_i}^{1:N}) \\ &= \prod_{i=1}^{L} \left(\prod_{\mathfrak{z} \in \tau_i} \left(\sum_{\mathfrak{u}=0}^{N} p(J_\mathfrak{z}^{1:N}|t_\mathfrak{u}) \int_{t_\mathfrak{u}}^{t_{\mathfrak{u}+1}} p(T_{\tau_i}) \right) \right), \end{aligned} \tag{6.9}$$

where we first decomposed the full joint evidence into the product of the cliques τ_i, then further decomposed each clique into the product of its individual terms, which are tied together by the joint prior $P(T_{\tau_i})$. We make use of (6.7) to write out the integral over joint prior survival times as a product of conditional distributions on one element of the clique $p(T_{\tau_i}) = \prod_{\mathfrak{z} \in \tau} p(T_\mathfrak{z}|T_k)$ where each conditional prior can be evaluated according to (6.8).

6.3.2.1 Marginal Formulation

Suppose we wish to estimate the marginal persistence for a feature a which is correlated to another feature b, given a sequence of N detections $J_a^{1:N}$, $J_b^{1:N}$ from time $t \in [t_1, t_N]$. We may make the reduction:

$$\begin{aligned}
p(\Theta_a^t = &1 | J_a^{1:N}, J_b^{1:N}) = \\
&p(T_a \geq t | J_a^{1:N}, J_b^{1:N}) \\
&= \int_0^\infty p(T_a \geq t, T_b | J_a^{1:N}, J_b^{1:N})\, dT_b \\
&= \int_0^\infty \frac{p(J_a^{1:N}, J_b^{1:N} | T_a \geq t, T_b) \cdot p(T_a \geq t, T_b)}{p(J_a^{1:N}, J_b^{1:N})}\, dT_b \\
&= \frac{p(J_a^{1:N} | t_N) p(T_a \geq t)}{p(J_a^{1:N}, J_b^{1:N})} \times \\
&\quad \int_0^\infty p(J_b^{1:N} | T_b) p(T_b | T_a \geq t)\, dT_b,
\end{aligned} \tag{6.10}$$

where we use the fact that $p(J^{1:N}|T)$ is constant in the intervals $[t_i, t_{i+1}]$ to define the integral in (6.10) as

$$\begin{aligned}
\int_0^\infty & p(J_b^{1:N} | T_b) p(T_b | T_a \geq t)\, dT_b \\
&= \sum_{i=0}^{N} p(J_b^{1:N} | t_i) \int_{t_i}^{t_{i+1}} p(T_b | T_a \geq t)\, dT_b.
\end{aligned} \tag{6.11}$$

Note that in the case where features a and b are independent, $p(T_b|T_a \geq t) = p(T_b)$ and (6.11) simply becomes the marginal $p(J_b^{1:N})$ which cancels part of the evidence in (6.10); this results in exactly the posterior defined in [126].

6.3.3 Feature Correlation Design

In this section we describe how to design the weights π in (6.8). Since there is no inherent structure to the sparse feature-based environmental representation, we design a prior structure that aims to capture the underlying structure of the environment. It is possible to learn feature correlations from the sensor data used to create the feature map (i.e image frames or point clouds) using a object detector to semantically segment the environment. However the original sensor data used to create the map is not always readily available. We deal with that scenario, where the only input to designing feature correlation is the sparse feature map itself.

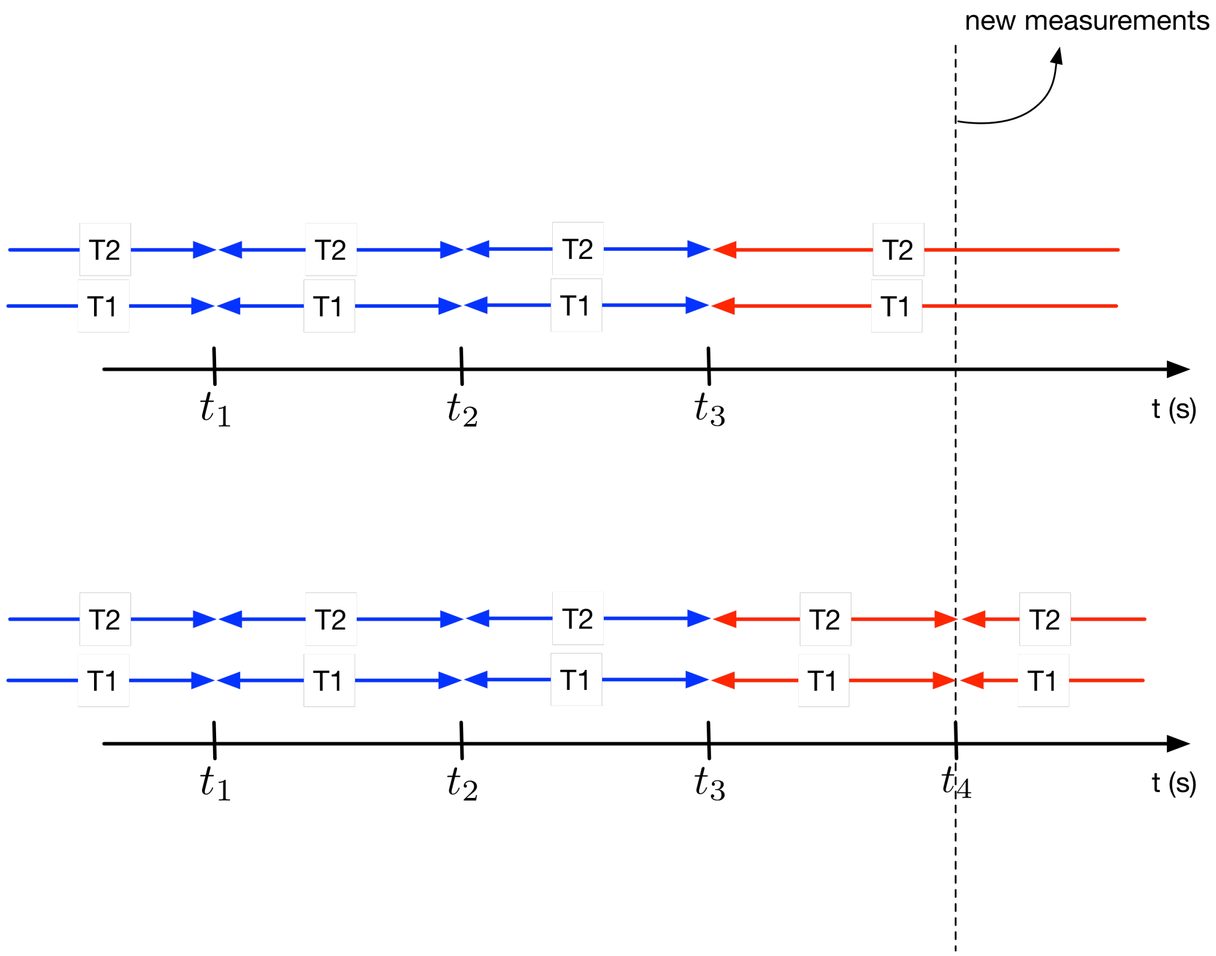


Figure 6.4: Terms which need to be computed for the marginal evidence with two features $[T_1, T_2]$. The segments represented by arrows are the the possibility that each feature's survival time T is within that range. Each t_i represents the time in which a new detection was received. Computing the joint evidence requires summing over the detection likelihood for all measurements given every configuration of T_1, T_2. When a new measurement is received (dotted vertical line) at time t_4 we only need to recompute terms associated with the segments highlighted in red.

We model features which where observed at a similar point in time, and are physically close to have correlated persistence. The intuition is that if a set of features is co-observed and geometrically close, the likelihood that they belong to the same semantic object (e.g. lane markings, sign post) is high. We define the set $\mathfrak{N}$ of all features that were observed within Δs, and within that set we employ a Euclidean nearest neighbors metric to group features in cliques of up to $n = 5$ features, which are within a maximum distance d_{max} to the center of the clique. When applied to sparse feature-based maps in which clutter is reduced this method is a general way of capturing the underlying scene structure. The weights π between features are then computed as the inverse Euclidean distance

$$\pi_{ij} = \frac{1}{\left\| \hat{\mathcal{X}}_i - \hat{\mathcal{X}}_j \right\|}, \tag{6.12}$$

where π_{ij} is the normalized weight such that $\sum_j \pi_{ij} = 1$. Computing the set of cliques and their corresponding weights can be performed offline. Figure 6.3 demonstrates the effect the weights π have on the prior distribution $p(T_i)$; The conditional distribution $p(T_i|T_j > t)$ with given weights π_{ij} is a mixture model with reduced probability mass before t. In the case of independent features $\pi_{ii} = 1$ and Figure 6.3 becomes the solid line.

6.3.4 Recursive Estimation

Computing the full posterior in (6.3) every time a new observation is included is computationally expensive, in this section we show how to compute both the joint distribution $P(\Theta^t|J^{1:N})$ and the marginal $P(\Theta_i^t|J^{1:N})$ as described in Section 6.3.2.1 in a recursive manner by re-using previously computed terms. This allows for a constant-time update when a new detection is received. When a new observation $j^{t_{N+1}}$ is appended to the observation vector $J^{1:N}$, $t_{N+1} > t_N$ and given the independence assumption in (6.4), the updated joint likelihood is

$$p(J^{1:N+1}|T \geq t) = \prod_{i=1}^{M} \prod_{k=1}^{N+1} p(j_i^{t_k}|T_i \geq t). \tag{6.13}$$

So the updated joint likelihood at time t_{N+1} can be written in terms of the previous at time t_N

$$p(J^{1:N+1}|T \geq t) = p(J^{1:N}|T \geq t) \prod_{i=1}^{M} p(j_i^{N+1}|T_i \geq t). \tag{6.14}$$

Updating the joint evidence is less straightforward due to having to integrate over all possible survival times for all features. However using the decomposition of the joint prior distribution in (6.7) where we have L cliques τ_i, when a new measurement at t_{N+1} is observed, we can compute the updated joint evidence as

$$p(J^{1:N+1}) = \prod_{i=1}^{L} p(J_{\tau_i}^{1:N+1}). \tag{6.15}$$

If a clique τ_k has τ_M features the clique evidence $p(J_{\tau_k}^{1:N})$ can be computed as

$$p(J_{\tau_k}^{1:N}) = \sum_{i=0}^{N-1} p(J_{\tau_k}^{1:N}|t_i)p(t_{i-1} \leq T_{\tau_k} < t_i) \\ + p(J_{\tau_i}^{1:N}|T_{\tau_i} \geq t_N), \tag{6.16}$$

which implies that the updated *partial evidence*, which excludes the last term in the sum (where the survival time is T_{N+1} after incorporating the measurement at time t_{N+1} is

$$p_L(J_{\tau_k}^{1:N+1}) \triangleq \left[\left(\sum_{i=0}^{N-1} p(J_{\tau_k}^{1:N}|t_i)p(t_{i-1} \leq T_{\tau_k} < t_i) \right) \right. \\ \left. + \big(p(J_{\tau_k}^{1:N}|t_N)p(t_N \leq T_{\tau_k} < t_{N+1})\big) \right] \\ \times \prod_{i=1}^{\tau_M} p(j_i^{N+1}|T_i < t_{N+1}). \tag{6.17}$$

The full updated evidence can be obtained by combining the partial evidence from (6.17) with the measurement probability given feature survival times at time $T = t_{N+1}$

$$p_(J_{\tau_k}^{1:N+1}) = p_L(J_{\tau_k}^{1:N+1}) + \prod_{i=1}^{\tau_M} p(j_i^{N+1}|T_i \geq t_{N+1}) \tag{6.18}$$

The updated joint distribution can thus be computed for each clique τ_i using Eqs. (6.13), (6.15) and computing the prior using (6.7). Figure 6.4 shows the terms that need to be updated every time a new measurement is incorporated. In summary, we use the fact that the measurement likelihoods are constant between observations (e.g. between t_i and t_{i+1}) to discretize the integral over survival times T, and that when a new measurement is incorporated at time t_{N+1} the terms corresponding to survival times before t_N can simply be updated my multiplying by the probability of seeing measurement j^{N+1} given that the survival time for that feature is before t_{N+1} (segments in blue in Figure 6.4). The marginal persistence probability can be updated in an analogous manner and is omitted due to space constraints.

6.4 Robust Data Associations and Localization

In this section we briefly present a common data association technique and show how our work estimating feature persistence can disambiguate the data association problem in the presence of semi-static scene elements. The *maximum likelihood* (ML) or *individual compatibility* solution to data association is based on probabilistic methods and can be summarized as taking into account the uncertainties between the robot's location and the landmark position in the map. It can be interpreted as a simple nearest neighbors data association, but with Euclidean distance replaced by the Mahalanobis distance. The standard approach is to model each measurement $z \sim \mathcal{N}(h(\mathcal{X}), \Gamma)$ as a Gaussian distribution with given covariance Γ. Taking the negative logarithm we obtain the maximum likelihood cost function

$$D^2 := ||h(\hat{\mathcal{X}}) - z||^2_{\Lambda}, \tag{6.19}$$

where D is the Mahalanobis distance and the covariance Λ is defined as

$$\Lambda := \frac{\partial h}{\partial \mathcal{X}} \Sigma \frac{\partial h}{\partial \mathcal{X}}^T + \Gamma, \tag{6.20}$$

with Σ the current state uncertainty. The hypothesis that a given measurement z was caused by the j^{th} landmark can be evaluated based on a chi-square acceptance decision $D < \chi^2_{d,\alpha}$ where α is the desired confidence level and d the dimension of the measurement. Instead of considering every map feature as a candidate for data association, we use the marginal persistence estimate for each feature to weight the data associations. We are specifically interested in avoiding incorrect data associations when there are multiple hypothesis with similar cost; in the scenario where we consider feature persistence independently and the pose uncertainty is non-negligible, it is impossible to distinguish between two hypothesis: associate the measurement to a map feature or consider it a new feature, given that the static world assumption no longer holds. However given some environmental structure, the joint persistence estimate can capture the difference, as shown in Figure 6.5. The only assumption is that at some point at least one feature in a clique of correlated features was independently determined to have low persistence. For example, if a sequence of consecutive and co-linear lane markings are represented by point features and have correlated joint prior distribution over persistence, and those lane markings have shifted slightly the assumption is that at some point the pose uncertainty over the vehicle location is small enough that we are able to determine that at least a single feature in that group no longer exists; then even if the pose uncertainty grows and we are not able to individually determine that every feature in the group no longer exists, the joint persistence model allows us to infer the non-existence of the other lane markings, an example of this is shown in Figure 6.2.

6.5 Experiments and Results

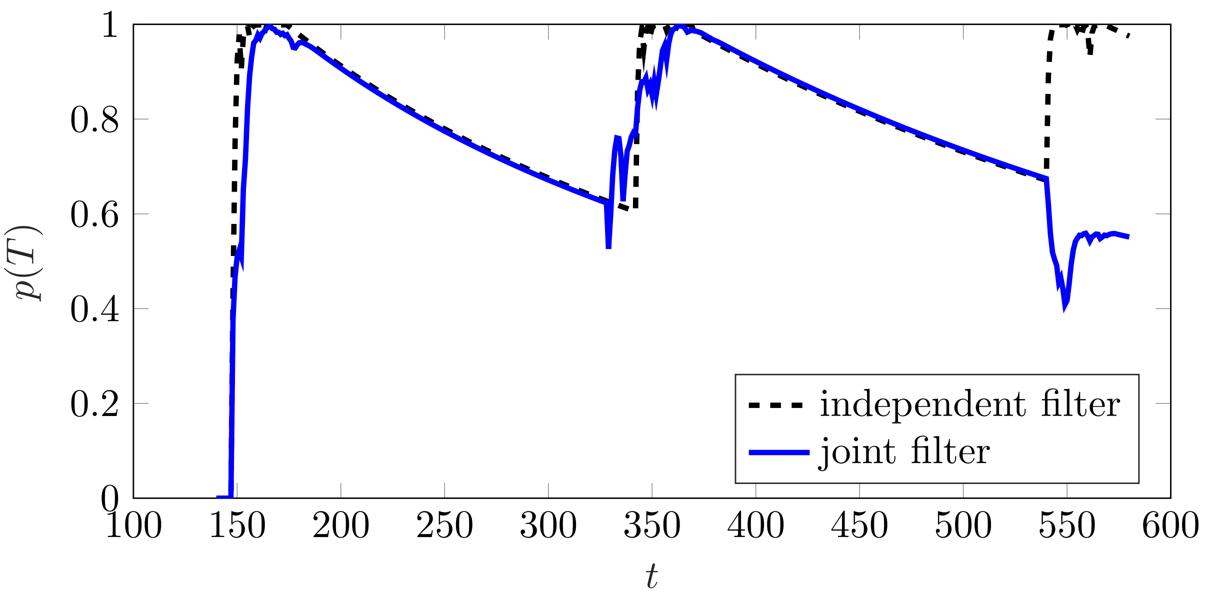


Figure 6.5: Comparison of independent filter and joint filter for a single point feature. With no changes to the environment ($t \in [0, 540]$) the filters are equivalent. At time 540 the feature is moved by 1m in the x direction, due to increased pose uncertainty the measurements from the moved feature are associated to the old map point, and the independent filter (dashed line) renews its belief over the feature's existence. The joint filter indicates a drop in belief due to the correlation to neighboring features which have been identified as removed.

Our evaluation of the proposed joint persistence formulation has two principal objectives: establish the *improved persistence estimate by modeling correlation between features* and how can we use the persistence for each feature to perform robust data associations and successfully navigate challenging dynamic environments. To that extent we use both simulated and real datasets with ground truth information to assess persistence and localization error. Due to the lack of a dataset with naturally occurring changes of a semi-static nature, we impose changes on the environment to varying degrees. In order to assess the usefulness of estimating feature persistence in a factor-graph SLAM setting, we implement a 2D SLAM system in which odometry and range and bearing measurements are added to a keyframe-based fixed-lag smoothing [135] back-end which uses ceres-solver [4] to solve the non-linear least squares problem.

Initially the map is created with known data associations, since landmark initialization is not the focus of this work. Once the map of 2D point features has been created, Mahalanobis-distance data association as described in Section 6.4 is used to chose a pairing between each range and bearing measurement and point feature in the map, using the covariance from the latest state estimate $\hat{\mathcal{X}}$ which can be efficiently recovered from the square-root information form using [62] and the covariance from the estimated map feature. This setup allows for realistic modeling of the feature detector (i.e data associations) in terms of the pose uncertainty, clutter in the environment and the type of change the landmark underwent (removed vs. moved).

6.5.1 Simulation

We generate a 2D map with N features drawn uniformly from a predefined set of environments (road, random landmarks, hallway) and simulate a robot traveling along a pre-determined path. Wheel odometry measurements are generated at 50Hz, composed of forward velocity (m/s) and steering angle (rad). The measurements are corrupted by zero-mean Gaussian noise with standard deviation of 0.1m/s and 0.09 rad ($\sim 10°$). Range and bearing measurements are generated for every landmark in the robot's field of view (90°, 10m) and corrupted by zero-mean Gaussian noise of 0.05m and 0.08 rad respectively. Visible landmark measurements are removed with probability P_M sampled uniformly $P_M \sim \mathcal{U}([0.01, 0.3])$ and spurious range and bearing measurements are sampled uniformly from the robot's field of view with probability $P_F \sim \mathcal{U}([0.01, 0.15])$. For each τ_i clique of features in the map we sample a survival time $T \sim [0, 1000]$, since we make the assumption that features in the same clique have correlated persistence. If we define a persistence threshold $P_d = 0.5$ in which features are removed from the map the performance of the removal classifier in terms of the amount of change Δ_i each feature undergoes can be assessed in terms of feature removal precision and recall. In a map with 50 features in which a single feature is changed by varying amounts we average 200 observation sequences for that feature. Figure 6.6 shows the average removal precision and recall for both the individual persistence filter as in [126] and this work. The recall for the individual filter is considerably worse than the joint filter for small changes in the feature, as expected since given the uncertainties in the measurements the data association step will associate the measurement from the moved landmark to the stale map feature, thus renewing its belief and flagging the feature for removal.

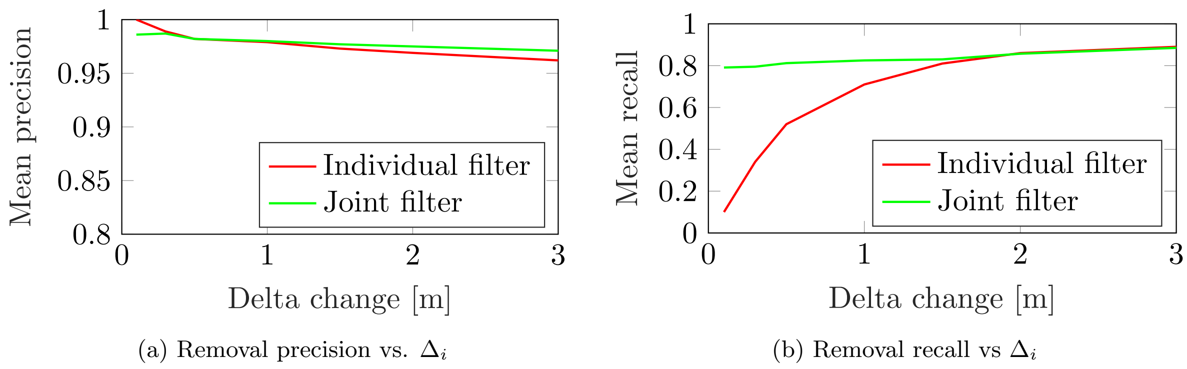
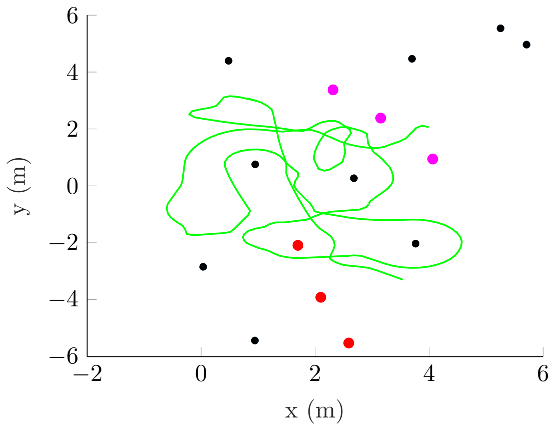


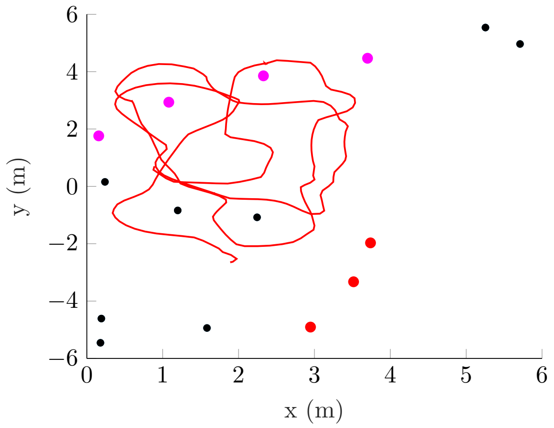
Figure 6.6: Precision and recall for feature removal with removal threshold set to $P_d = 0.5$ over 200 observations of a single feature which was moved by Δ_i as described in Section 6.5.1 for a total duration $T = 1000$. The feature is moved by $\Delta_i \in \{0.1, 0.3, 0.5, 1, 1.5, 2, 3\}$ meters. The persistence used for determining if a feature should be removed or not is computed using the marginal persistence $p(T_i > t|J^{1:N})$ where $J^{1:N}$ are all the detections for every feature in the map.

6.5.2 Real Data

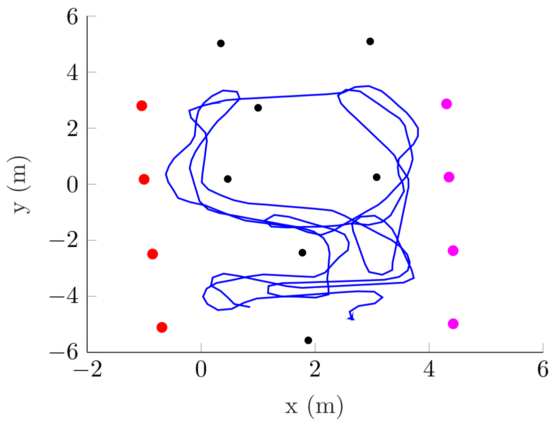
In order to validate the proposed method on real data we use the UTIAS [82] dataset which has ground truth poses sampled at 100Hz with accuracy in the order of $1e^{-3}$m. This dataset is composed of a 2D sparse point-feature based map with ground-truth data associations provided. Since there are no semi-static landmarks in the dataset, we impose change in the landmarks in a similar manner as described in Section 6.5.1. We wish to assess the robustness of the proposed method to various environmental configurations, in terms of the pose RMSE. Figure 6.7 shows three datasets in which feature cliques were created based on co-linearity of features. The cliques



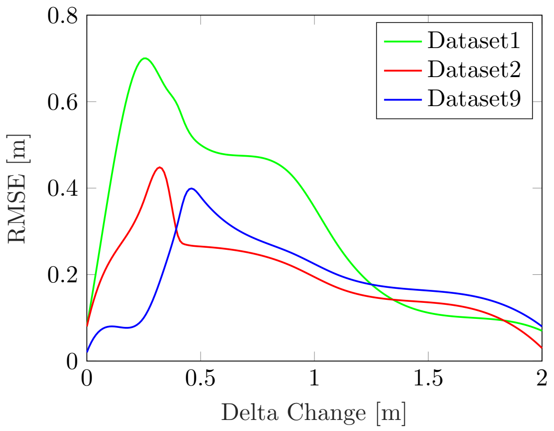
(a) Dataset 1 mean trajectory



(b) Dataset 2 mean trajectory



(c) Dataset 9 mean trajectory



(d) RMSE vs. Change

Figure 6.7: Evaluation of robust localization in the presence of small changes to landmarks in different configurations. Three scenarios from the UTIAS [82] dataset were used to assess the robustness of the proposed method to small changes. Individual features (black) do not have correlated persistence to any other features. Each scenario has two cliques (red and magenta) which have correlated persistence. In each of the three environments the features in a clique were moved in the x direction between $[0, 2]$m in increments of 0.01m. (a) - (c) represent the average estimated trajectory position over all change configurations, (d) shows the RMSE for each scenario vs. feature translation.

were then moved by increments of 0.01m and a batch estimate was computed for each increment. A feature removal threshold of $P_d = 0.6$ was used for data associations. Figure 6.7d shows the evolution of the RMSE for each dataset vs. the clique translation. The RMSE increases with very small ($[0, 0.15]$m) translations since all features in the clique are still detected and the incorrect map features are still included in the optimization. The error reducing when at least one feature in the clique is determined to be below the removal threshold P_d. It is interesting to note that the detection, and subsequent update to persistence for each feature is entirely dependent on the data association, which in turn is a function of the structure of the environment (i.e. clutter, number of features) and the pose and feature uncertainty at the moment of data association, regardless of the data association scheme used (e.g. JCBB, IC). The three datasets in Figure 6.7 show a drop in RMSE at a similar change point ($\sim 0.4m$) indicating a realistic minimum feature change for any kind of persistence aided localization to be effective.

6.6 Discussion

We present an general formulation for feature persistence which makes use of correlation between feature persistence imposed by a joint prior distribution which may be learned or engineered. The key insight of this work is proposing a joint distribution over feature persistence which is computationally tractable in constant time. Our proposed formulation improves upon prior work on modeling individual feature persistence by demonstrating the necessity of a joint model when the environment is subject to change. Our approach allows for the use of the survival time prior distributions discussed in [126] while incorporating information about environmental structure. We show approximated constant-time online inference over feature persistence in a graph-slam environment in both simulated and real scenarios, subject to landmark change. The use of persistence for informed data associations allows navigating through challenging dynamic environments where considering the joint feature persistence is essential.

Chapter 7

FastCal: Robust Online Self-Calibration for Robotic Systems

7.1 Problem Statement

Autonomous platforms destined for long-term applications equipped with multiple sensors such as cameras and Inertial Measurement Unit (IMU) have become increasingly ubiquitous. Generally these platforms must undergo sophisticated calibration routines to estimate extrinsic (sensor-to-sensor rigid body transform) parameters to high degrees of certainty before sensor data may be interpreted and fused. Once fielded, calibration parameters are generally fixed for the lifetime of the platform. However, in many applications these platforms experience gradual changes in calibration parameters due to e.g. non-rigid mounting, accidental bumps and temperature dilation that can change sensor extrinsic parameters. Self-calibration addresses this by inferring extrinsic parameters pertaining to proprioceptive and exteroceptive sensors without using a known calibration target or a specific calibration routine. The motivation behind self-calibration is to remove the explicit, tedious, and sometimes nearly impossible calibration procedure from robotic applications and to enable robust long-term autonomous operation. Self-Calibration is an essential part of any long-term robotic system, as such is under constant pressure to increase its accuracy, speed and robustness. A higher speed allows its inclusion into larger systems with extensive subsequent processing (*e.g.* localization, mapping, object/activity recognition, planning) and its deployment in computationally constrained scenarios (*e.g* planetary exploration, embedded systems). A robust self-calibration system should cope with unobservable directions in the parameter space (*e.g.* due to a nonholonomic platform, measurement noise which makes unobservable parameters *appear* observable) and changes and drift in calibration parameters. In our previous work [115] we have partially addressed constant-time self-calibration using a priority queue and informative segments. [116] addresses drift and slow changes in calibration parameters by attempting to regress the change point and retroactively correcting the state estimates and in [113] we propose an observability aware framework capable of updating only the observable directions of the parameter space, even in the presence of noisy measurements, which is then used in a reinforcement-learning framework to learn informative motions to be suggested to a human operator. In this chapter we leverage some individual aspects of previous works and propose significant novel contributions:

1 A novel formulation for regressing extrinsic calibration parameters which is suitable for integration with any existing SLAM system, while being considerably faster than the system used in [115] due to a loosely coupled formulation which optimizes over relative poses instead of jointly over raw sensor measurements. We also propose a novel criteria for adding informative segments to the estimation queue which minimizes the number of times the entire segment queue needs to be optimized.

2 We handle intrinsically degenerate scenarios in noisy nonholonomic systems by only updating the observable directions of the parameter space, demonstrating the necessity and usefulness in a real-world robotic application.

3 Slow changes over arbitrary periods of time are handled by continuously renewing the segment queue by use of time-decay on measurements, this approach is shown to be much more efficient and robust compared to [116], at the cost of immediate and local accuracy.

4 Integration of the proposed self-calibration system into a real-world robotic platform operating in challenging environments for extended periods of time[1] .

[1] 3 weeks of operation, approximately 170km driven.

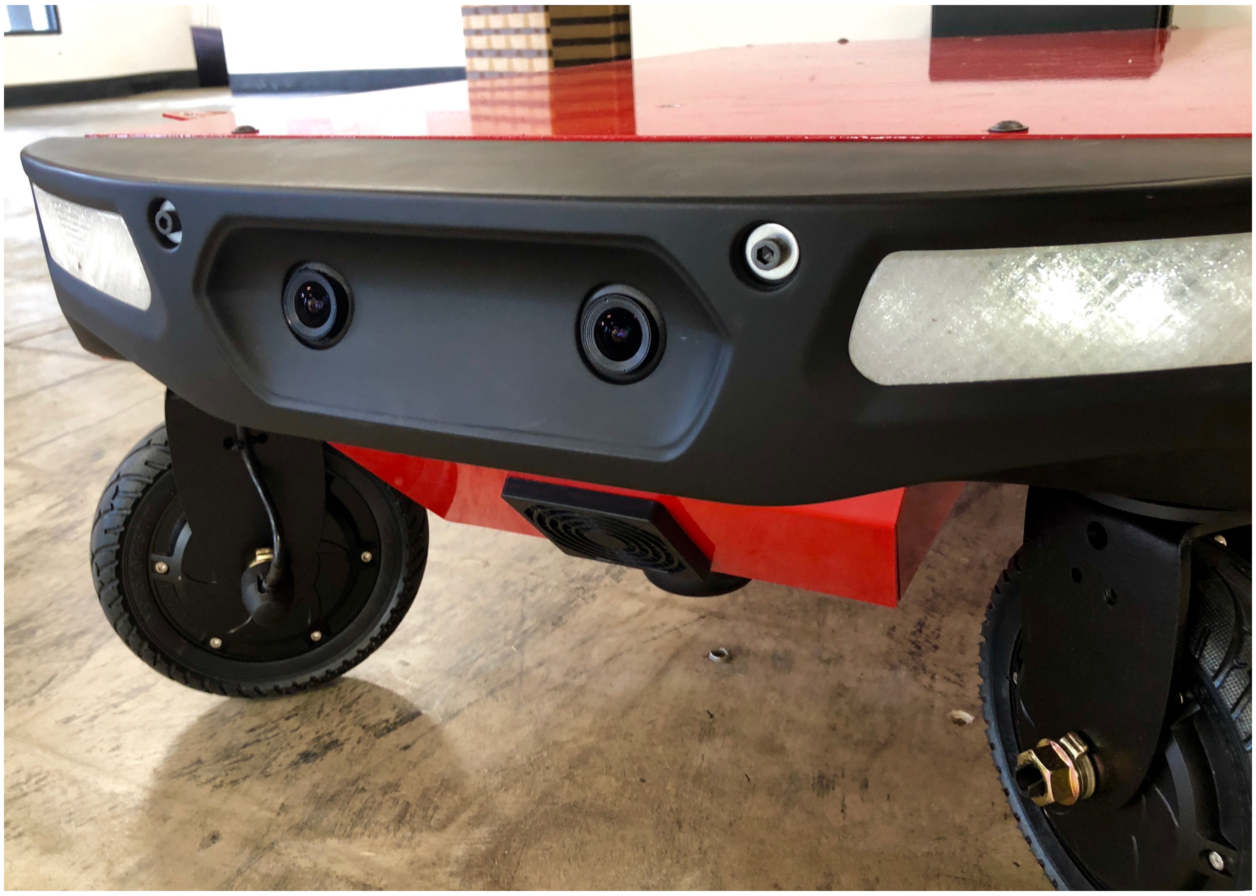


Figure 7.1: Robotic platform equipped with two camera stereo pairs, one in front and one in the back, used for experiments.

7.1.1 Related Work

Most current techniques for vision-aided inertial navigation use filtering approaches [61, 69, 106] or a smoothing formulation. In either case the estimation is made constant-time by rolling past information into a prior distribution. Filtering methods present the significant drawback of introducing inconsistencies due to linearization errors of past measurements which cannot be corrected post hoc, particularly troublesome for non-linear camera models. Some recent work has tackled these inconsistencies; see, e.g. [87, 55, 23, 88]. The state-of-the-art includes methods to estimate poses and landmarks along with calibration parameters, but these approaches do not output the marginals for the calibration parameters, which are desirable for long-term autonomy applications.

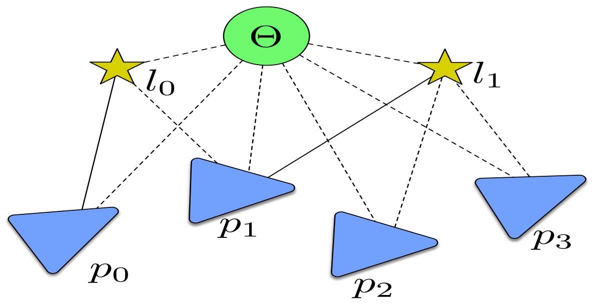
The use of a known calibration pattern such as a checkerboard coupled with nonlinear regression has become the most popular method for camera calibration in computer vision during the last decade; it has been deployed both for intrinsic camera calibration [140] and extrinsic calibration between heterogeneous sensors [151]. While being relatively efficient, this procedure still requires expert knowledge to reach a discerning level of accuracy. It can also be quite inconvenient on a mobile platform requiring frequent recalibration (e.g experimental platforms which undergo constant sensor changes). In an effort to automate the process in the context of mobile robotics, several authors have included the calibration problem in a state-space estimation framework, either with filtering [98] or smoothing [79] techniques. Filtering techniques based on the Kalman filter are appealing due to their inherently online nature. However, in case of nonlinear systems, smoothing techniques based on iterative optimization can be superior in terms of accuracy [139].

Our approach does not rely on formal observability analyses to identify degenerate paths of the calibration run as in [16], since these approaches still expect non-degenerate excitations.

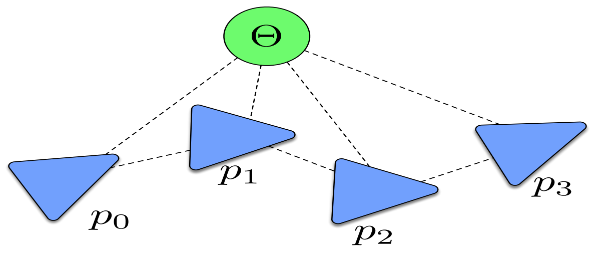
A last class of methods relies on an energy function to be minimized. For instance, Levinson and Thrun [85] have defined an energy function based on surfaces and Sheehan et al. [134] on an information theoretic quantity measuring point cloud quality.

7.2 Methodology

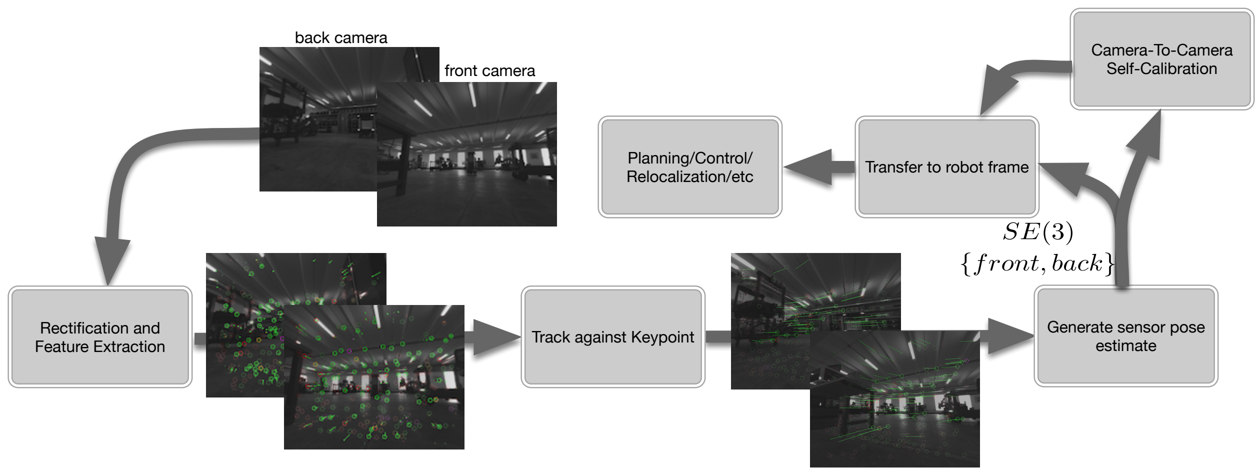
It is common for robotic platforms to have multiple sensors, such as cameras, wheel encoders, IMU. This creates the need to obtain the relative rigid body transform between sensors so that a fused position estimate may be obtained. This is what we refer to as *calibration parameters* in this work, represented by Θ. There are other calibration parameters which can be estimated, such as camera intrinsics (*e.g.*focal length, center point, distortion parameters) but these have been found to not vary considerably even in long term operation. Sensor extrinsics however, change frequently (Section 7.3) and have considerable impact on the resulting position estimation. For these reasons we focus on estimating sensor extrinsics. For camera intrinsic self-calibration refer to [115, 66]. The proposed FastCal algorithm can be divided into three major components, summarized in algorithm 2: 1) Selecting informative segments so as to bound the computation time. 2) Updating only the observable directions of the parameter space. 3) Considering drift in the calibration parameters over time by time-decaying measurements. Figure 7.2 shows the calibration problem posed as a factor graph, both as a tightly couple problem where the poses, landmarks and calibration parameters are estimated jointly and as a loosely coupled problem where the individual sensor pose-graphs are estimated independently, and the calibration parameters are obtained in a subsequent optimization. The block diagram for a typical SLAM system is also shown, where there is usually an odometry node which provides relative poses. We also focus on creating an algorithm which has as few tuning parameters as possible; the parameters for FastCal are summarized in table 7.1, the parameters in bold are the ones with highest impact on the performance of the algorithm.



(a) Tightly coupled problem.



(b) Loosely coupled problem.


back camera
front camera
Camera-To-Camera Self-Calibration
Planning/Control/ Relocalization/etc
Transfer to robot frame
$SE(3)$
$\{front, back\}$
Rectification and Feature Extraction
Track against Keypoint
Generate sensor pose estimate

(c) System diagram.

Figure 7.2: (a) Landmark measurements are used to jointly estimate camera poses and camera-to-camera extrinsics (Θ). (b) Camera poses are estimated independently; The calibration parameter Θ is estimated in a second step. (c) System diagram of a typical SLAM system, the input is each camera image, and the self-calibration block integrates easily into any existing odometry pipeline by subscribing to the existing odometry block.

Algorithm 2: FastCal Algorithm

```
Data: relative pose measurements for each of the $N$ sensors; reference sensor $N_{ref}$, initial
      guess on sensor placement.
Result: sensor-to-sensor extrinsic $SE(3)$: $\Theta$
Initialize $\Theta \leftarrow$ initial guess;
if $num\_measurements(\mathcal{D}^{candidate}) < \theta_{meas}$ then
    $\mathcal{D}^{candidate} \leftarrow$ new measurements;
else
    Estimate $\Theta|\mathcal{D}^{candidate}$ according to (7.3);
    if $num\_segments(\mathcal{D}^{info}) < \theta_{pq}$ then
        $\mathcal{D}^{info} \leftarrow \mathcal{D}^{candidate}$;
    else
        Check if $\mathcal{D}^{candidate}$ should be swapped into $\mathcal{D}^{info}$, according to 7.2.2 ;
        if $\mathcal{D}^{info} \leftarrow \mathcal{D}^{candidate}$ then
            Estimate $\Theta|\mathcal{D}^{info}$ according to (7.8) with TSVD;
        end
    end
end
for $i \leftarrow 0$ to $num\_segments(\mathcal{D}^{info})$ do
    if Time Decay for $\mathcal{D}_i^{candidate}$ according to (7.9) $< 0.001$ then
        remove $\mathcal{D}_i^{candidate}$ from $\mathcal{D}^{info}$
    end
end
```

7.2.1 Problem Formulation

We focus on estimating the sensor-to-sensor SE(3) rigid body transform between two cameras with no co-visible features. The calibration problem can be framed as an optimization problem in a Bayesian estimation framework, by including the calibration parameters in the standard SLAM

Table 7.1: FastCal Parameters

Function	Symbol	Default Value
TSVD threshold	$\theta_{\epsilon_{svd}}$	0.1
Maximum entropy	$\theta_{\Sigma_{max}}$	15
Number of segments in priority queue	θ_{pq}	10
Number of measurements in each candidate	θ_{meas}	10
Keyframing translation [m]	$\theta_{kf_{trans}}$	0.15
Keyframing rotation [rad]	$\theta_{kf_{rot}}$	0.1745
Time decay	θ_{λ}	0.04
Number of consecutive estimates at the same value	θ_{same}	3
Min total update	θ_{min_update}	0.008

formulation:

$$\hat{\mu}_{\Theta\mathcal{X}\mathcal{L}} = \underset{\Theta\mathcal{X}\mathcal{L}}{\arg\max}\, p(\Theta, \mathcal{X}, \mathcal{L}|\mathcal{Z}) = \underset{\Theta\mathcal{X}\mathcal{L}}{\arg\min} -\log p(\Theta, \mathcal{X}, \mathcal{L}|\mathcal{Z}); \tag{7.1}$$

Where the estimated parameter μ contains the robot pose ($\mathcal{X}$), landmarks ($\mathcal{L}$) and calibration (Θ) parameters. $\mathcal{Z}$ in this context are the sensor measurements, such as landmark observations. The advantages of solving the problem in this formulation is leveraging joint information from all measurements, at the price of higher computational complexity since we must solve a larger system comprised of N poses, M landmarks in addition to the calibration parameters.

Alternatively, we can leverage the fact that most robotics systems already estimate the reduced state:

$$\hat{\mu}_{\mathcal{X}\mathcal{L}} = \underset{\mathcal{X}\mathcal{L}}{\arg\max}\, p(\mathcal{X}, \mathcal{L}|\mathcal{Z}) = \underset{\mathcal{X}\mathcal{L}}{\arg\min} -\log p(\mathcal{X}, \mathcal{L}|\mathcal{Z}); \tag{7.2}$$

for each camera, where $\mathcal{X} = [\mathbf{x}_{s1}, \mathbf{x}_{s2}, ..., \mathbf{x}_{sn}]$ the world position for N sensors at time t. Finding the sensor-to-sensor extrinsics can then be posed as an alignment problem, using $\mathcal{X}$ as the measurement instead of the landmark observations as in (7.1).

$$\hat{\Theta} = \underset{\Theta}{\arg\max}\, p(\Theta|\mathcal{X}) = \underset{\Theta}{\arg\min} -\log p(\Theta|\mathcal{X}); \tag{7.3}$$

This formulation has a few advantages: the least squares solution to (7.3) is simply a 6×6

system (for a single pair of sensors) which can be solved very efficiently and allows for the use of more informative decompositions as will be discussed in 7.2.3. This formulation also allows for easy integration into an existing SLAM system which already provides the independent sensor position estimates; All that needs to be done is subscribe to the camera positions being estimated and efficiently solve (7.3).

7.2.2 Informative Segment Selection

We want to benefit from the robustness of offline methods, along with the possibility to deploy our algorithm in an online and long-term setting. To reach this goal, we process small batches of data sequentially and decide to keep them based on their utility for the calibration. Each new batch is merged to the old ones to refine our knowledge about the calibration parameters until we reach a satisfactory level of confidence. This notion was introduced in [66] and extended to multiple sensors in [115]. We briefly explain the informative segment selection into a priority queue, to then introduce our novel metric on adding new batches of measurements. We define the current set of data samples in a priority queue as $\mathcal{D}^{info} = \{\mathbf{x}_1, ..., \mathbf{x}_n\}$ which has led to the posterior marginal density in equation (7.3), associated to the random variable $\hat{\Theta}|\mathcal{D}^{info}$. The set, $\mathcal{D}^{info}$, contains the current informative measurements for the calibration variable Θ. Our sensors, S, continuously stream new data which are used for estimating their relative positions, that we then accumulate in another batch of size ΔN denoted by $\mathcal{D}^{candidate} = \{\mathbf{x}_{n+1}, ..., \mathbf{x}_{n+\Delta N}\}$. Intuitively, if the measurements in $\mathcal{D}^{candidate}$ are similar to those in $\mathcal{D}^{info}$, we are not really improving our knowledge about Θ and we can safely discard $\mathcal{D}^{candidate}$ to keep the computation tractable. There are multiple ways of evaluating the usefulness of $\mathcal{D}^{candidate}$: Given the covariance of the estimated calibration parameters Σ_Θ, which can be obtained quickly from the solution to (7.3) by inverting the 6×6 information matrix, the entropy of the distribution is then given by $h = \frac{1}{2}\ln|2\pi e\Sigma_\Theta|$, where the bars denote the matrix determinant. In [115], the update criteria was if the entropy $h_{candidate}$ associated to the measurements $\mathcal{D}^{candidate}$ was smaller than the worst scoring batch in $\mathcal{D}^{info}$ by a certain margin, the segment was swapped into the informative segment queue and a new estimate for Θ was obtained by optimizing over all the measurements in the queue. This approach performs well, as shown in [66, 115], however it causes an excessive number of estimations of the entire priority queue, every time a new candidate segment is swapped in.

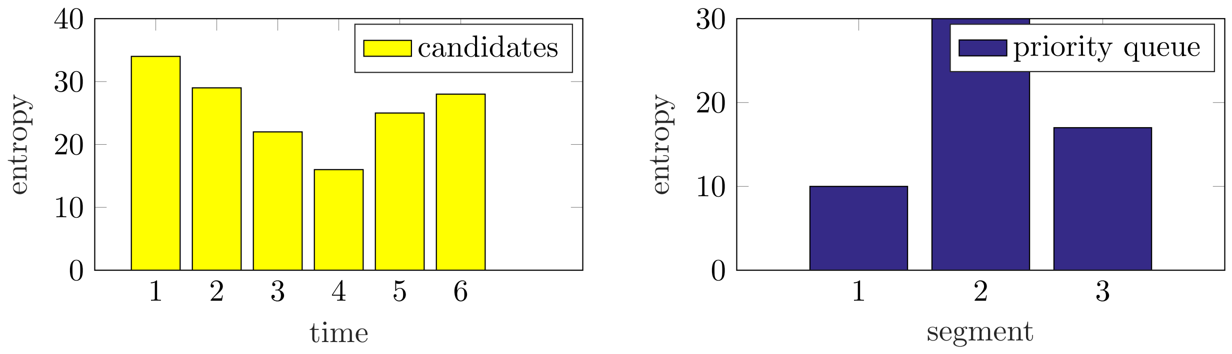


Figure 7.3: Left: Entropies of rolling candidate window over time. Candidate window at time $t = 2$ could be swapped into the priority queue in place of segment 2, however by waiting until time $t = 5$ we can instead swap in the candidate window at time $t = 4$, with much more information content. Right: Entropy of three informative batches in the priority queue.

We propose a different metric on adding batches to the priority queue: a candidate measurement batch is swapped into the queue iff. its entropy beats the worst scoring segment in the priority queue *and* the rolling candidate window has achieved a local entropy minimum. This ensures that we dont needlessly swap in segments to the priority queue and ensures that we are adding the best possible local window to the queue, not just the one that beat the worst scoring segment. Figure 7.3 shows on the left the entropy for a rolling window of $N = 10$ candidate measurements over time, on the right the entropies for the three segments in the priority queue. In the previous formulation the candidate window at time $t = 2$ has an entropy of 27 which beats segment 2 in the priority queue (with an entropy of 30) and would be swapped in. However by holding off on adding the candidate window to the priority queue, at time $t = 5$ we notice that the candidate entropy is increasing and as such we have reached a local minimum at $t = 4$, which is then swapped into the priority queue. This approach reduces the priority queue entropy much faster and leads to quicker convergence on Θ, as well as greatly reducing the number times the priority queue needs to be estimated.

The measurements which are added to each segment are the relative pose measurements generated independently for each sensor (each camera in our case) as shown in Figure 7.2. Most information based-approaches add every measurement to the candidate window, and re-estimate the calibration parameter $\Theta|\mathcal{D}^{candidate}$, however since most systems can generate relative pose measurements at over 20Hz, the information gain from each new individual measurement is small and results in several evaluations of $\mathcal{D}^{candidate}$. We aim to reduce the number of candidate window evaluations. Instead of adding each new measurement we take a keyframing based approach to self-calibration and accumulate relative pose measurements until the total relative transform reaches a threshold. In our case we use $\theta_{kf_{trans}} = 0.15m$ and $\theta_{kf_{rot}} = 0.1745rad$. This results in both much faster convergence and reduces the number of times the candidate window and priority queue need to be estimated.

7.2.3 Observability-Aware Estimation

We wish to reliably estimate the sensor extrinsics even in nonholonomic platforms which can never excite all degrees of freedom, and thus will not render the full calibration parameter space observable. We leverage our insights from [113] to use the Truncated SVD (TSVD) decomposition of the Fischer Information Matrix (FIM) and determine the observable directions in the parameter space, and only update those, even in the presence of noise. This is essential for platforms which, for example, only move on a planar surface and never excite roll, pitch or translation along the direction normal to the ground plane. In these cases the system (7.3) is ill conditioned and we have to either manually regularize the unobservable directions in order to make the system well conditioned or automatically detect the observable components and only update those directions. This is the approach we take in this work; We argue that manually regularizing explicitly throws away information which could be useful and implicitly biases the solution. Furthermore, by adopting the simplified parameter space, we can perform the SVD decomposition on the FIM very quickly, since it is of reduced size (6×6 for a pair of sensors). The least squares solution to (7.3) is

$$(\mathbf{J}^T\mathbf{G}^{-1}\mathbf{J})\delta\hat{\Theta} = -\mathbf{J}^T\mathbf{G}^{-1}\mathbf{r}(\hat{\Theta}), \tag{7.4}$$

Where J is the Jacobian matrix, G the measurement covariance matrix obtained from the first step and r the residual. There exists a solution to Eq. (7.4) iff the FIM is invertible, i.e. it is of full rank. The link between the rank of the FIM and observability of the parameters being estimated is well established in [60]. A singular FIM corresponds to some unobservable directions in the parameter space given the current set of observations. Classical observability analysis, for example the method of Hermann and Krener [52], proves structural observability—that there exists some dataset for which the parameters are observable—but it does not guarantee that the parameters are observable for any dataset.

Using singular-value decomposition (SVD) on the FIM we can identify a numerically rank-deficient matrix by analyzing its singular values and consequently the numerical observability of the system [50]. The *numerical rank* r of a matrix is defined as the index of the smallest singular value σ_r which is larger than a pre-defined tolerance $\theta_{\epsilon_{svd}}$, $r = \arg\max_i \sigma_i \geq \theta_{\epsilon_{svd}}$. When the noise affecting the matrix entries has the same scale (by using column or row scaling) then the numerical rank can be determined by the singular values. The scaling matrix S can be computed as:

$$S = \text{diag}\left\{\frac{1}{||J(:,1)||}, ..., \frac{1}{||J(:,n)||}\right\} \tag{7.5}$$

Where $||J(:,n)||$ denotes the column norm of the Jacobian matrix, for column n. Specifically we decompose the error covariance matrix $\mathbf{G}$ from Eq. (7.4) into its square root form by using Cholesky decomposition, $\mathbf{G}^{-1} = \mathbf{L}^T\mathbf{L}$, we can re-write Eq (7.4) in standard form:$(\mathbf{LJ})^T(\mathbf{LJ})\delta\hat{\Theta} = -(\mathbf{LJ})^T\mathbf{Lr}(\hat{\Theta})$ which are the normal equations for the linear system $(\mathbf{LH})\delta\hat{\Theta} = -\mathbf{Lr}(\hat{\Theta})$. Thus we can directly use a *rank-revealing* decomposition to estimate the numerical rank of the FIM, and consequently the numerical observability of the system. Let $(\mathbf{LJ})$ be a $m \times n$ matrix with the following SVD decomposition:

$$\mathbf{LJ} = \mathbf{USV^T}, \tag{7.6}$$

where $\mathbf{U}$ is $m \times n$ and orthogonal, $\mathbf{S} = diag(\varsigma_1, ..., \varsigma_i)$ the singular values and $\mathbf{V}$ an $n \times n$ matrix, also orthogonal. From Eq. (7.6) and the orthogonality of $\mathbf{U}$ and $\mathbf{V}$ we can solve (7.4) as

$$\delta\hat{\Theta} = -\mathbf{VS^{-1}ULr}(\hat{\Theta}), \tag{7.7}$$

Specifically, according to [50], we can efficiently obtain the update as:

$$\delta\hat{\Theta} = \sum_{i=1}^{r_\epsilon} \frac{\mathbf{u}_i^T \mathbf{r}_\Theta}{\varsigma_i} \mathbf{v}_i, \tag{7.8}$$

where $\mathbf{u}$ and $\mathbf{v}$ are the colmn vectors of $\mathbf{U}$ and $\mathbf{V}$. This allows us to only update the observable directions of the parameter space and maintain the other directions at their initial value. Establishing the value of ϵ to use is specific to the amount of noise expected in the measurements and is treated in Section 7.3.

7.2.4 Drift Correction

In order to correct for the inevitable drift over time on the calibration parameters due to physical shocks, maintenance, etc. We adopt a simpler strategy than what was used in [116]; We associate a exponential time-decay with each batch in the priority queue $\mathcal{D}info$. This is done by using a exponential distribution

$$p(t;\lambda) = \lambda e^{-\lambda t}, t \in [0, \infty] \tag{7.9}$$

Which has an expectation $E = \lambda^{-1}$, so we see that the parameter λ in (7.9) encodes the expected time that a set of measurements remains informative. There are several potential methods for selecting this parameter in practice, among them *class-conditional learning* could use machine learning techniques to learn class-conditional decay rates for certain environments (*e.g.* a warehouse where things move around a lot vs. a building where things rarely change). In this chapter we set the decay rate λ to a value empirically shown to balance continuously refreshing segments and number of priority queue estimations.

7.3 Experiments and Results

Our experimental setup consists of a robotic platform (Figure 7.1) designed to transport material autonomously in warehouses, deployed in diverse real-world scenarios. This is an example of a challenging long term autonomy deployment in adverse industrial scenarios, with constrained

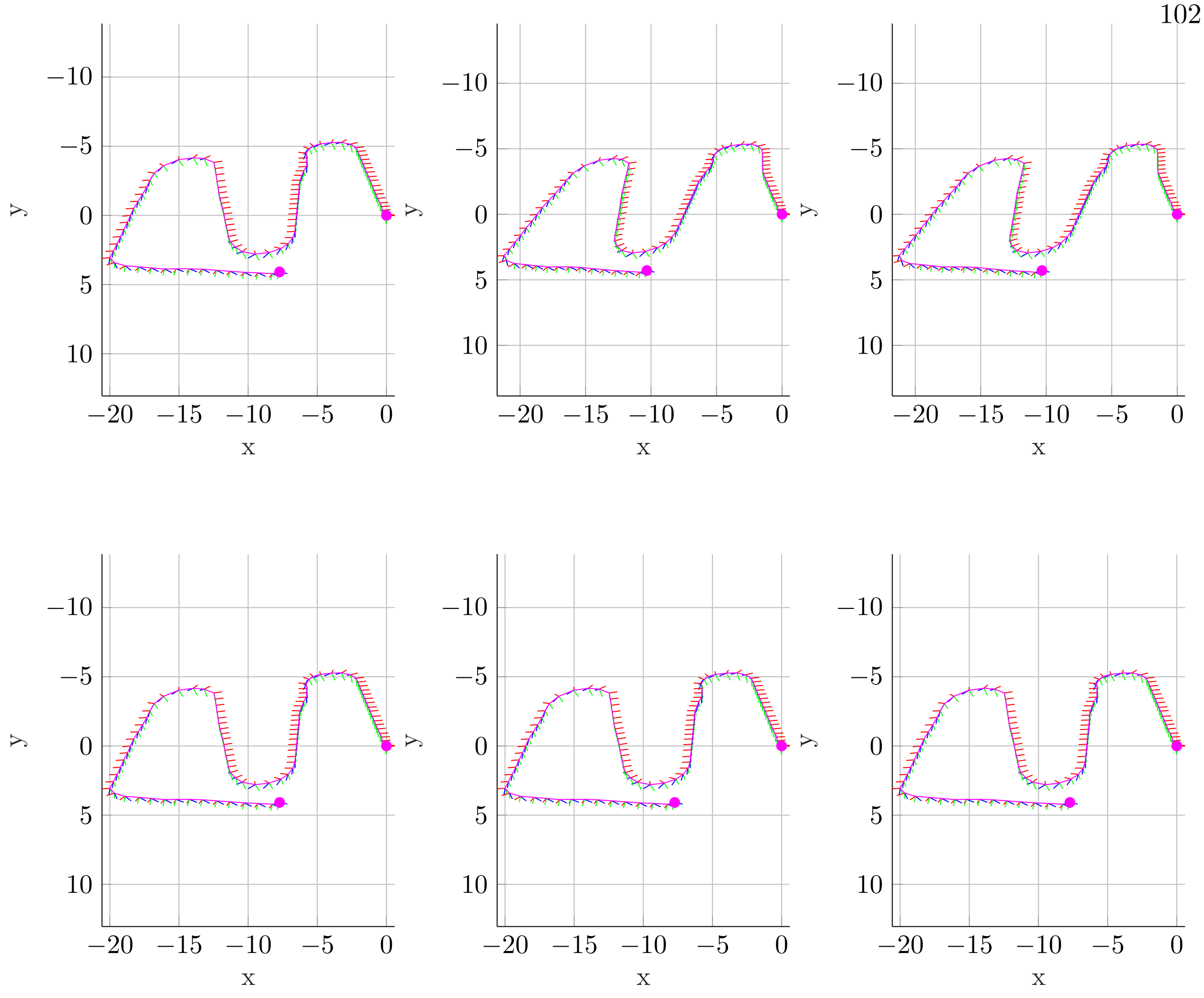


Figure 7.4: Same route performed over the span of multiple weeks. Top row: no self-calibration, using initial offline calibration only. Bottom row: FastCal enabled. Each column represents a day in which that route was performed. Columns are approximately 1 week apart.

resources and subject to physical impacts. It proves to be an excellent test-bed for the proposed FastCal algorithm which integrates into the SLAM module seamlessly by subscribing to sensor pose updates and registering a calibration update callback.

The robotic platform is equipped with two pairs of global shutter stereo camera pairs, one facing forward and one facing backward, with no overlapping field of view. We focus on estimating the front-to-back camera extrinsics, given an initial rough guess obtained with a measuring tape. We implement the proposed FastCal algorithm in C++, utilizing the parameters defined in Table 7.1. We wish to assess how often and by how much the extrinsics parameters actually change in

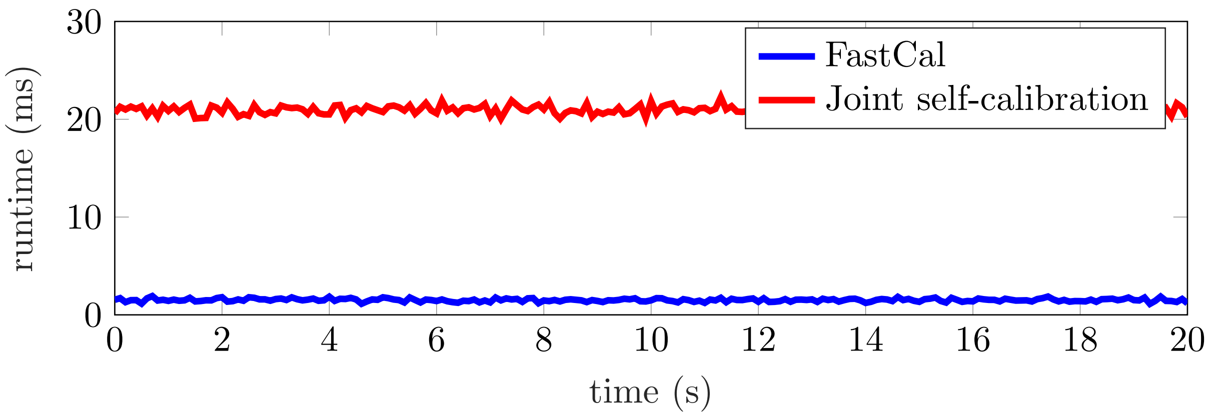


Figure 7.5: Timing comparison of the proposed FastCal algorithm vs a reference implementation of tightly coupled, priority queue based self-calibration, for the same size priority queue, candidate segment, number of iterations and fair parameter settings.

practice. For that end we run FastCal on the robot for a period of 14 days. In this time the robot was subject to cargo loads up to 150kg, was transported in a truck, had it's front bumper removed and re-attached due to maintenance reasons and finally suffered one accidental head-first collision while on joystick mode. These circumstances provide valuable data points on the usefulness and necessity of a robust self-calibration algorithm. Figure 7.4 shows the difference between using self-calibration vs. not using calibration over a relatively long period of time. The top row shows the estimated poses for a robot traversing the same path each time, over a period of three weeks. The bottom row shows a different robot which had FastCal enabled traversing the same path, also over three weeks. The position error is hard to determine exactly due to a lack of ground truth. We use the variance in poses between trajectories as an error metric, with the assumption that the robot should perform approximately the same path each time. The robot which did not have FastCal suffers considerable drift in its pose estimates ($\sim$ 4mσ) whereas the FastCal robot triggered 2 changes in calibration parameters, and has a much lower variance ($\sim$ 1mσ).

In order to compare the time complexity of FastCal vs. previous approaches [115] we compare the overhead that the self-calibration algorithm adds to the SLAM pipeline in two scenarios: using the proposed FastCal algorithm and a refernce implementation of the tightly coupled approach as described in [115]. The results are shown in figure 7.5; FastCal is about 4 times faster,

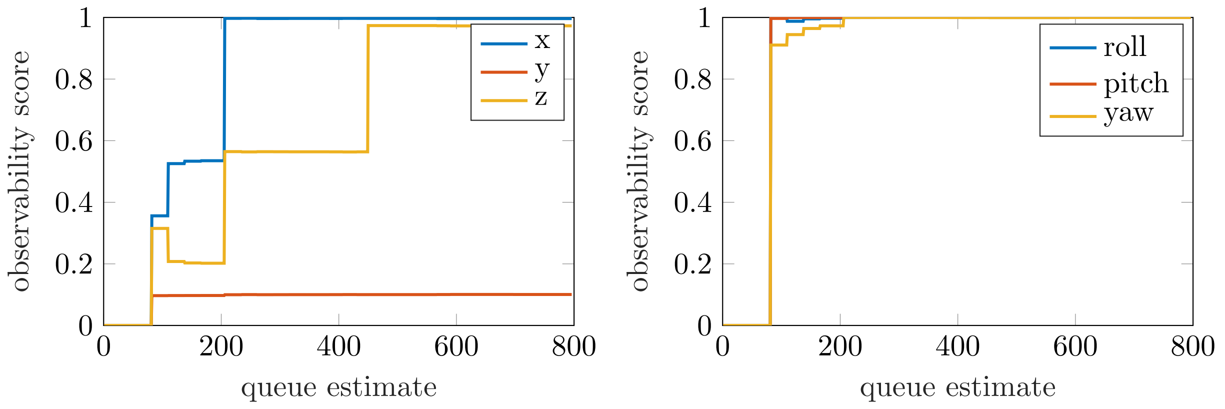


Figure 7.6: Observability score for each direction of the calibration parameter (SE(3) transform between cameras). Note that due to the planar motion, the y component is completely unobservable, even in the presence of noise; the unobservable direction remains clamped at its original value as a natural consequence of FastCal, with no need for explicit regularization.

while converging to the calibration parameters sooner and providing robustness to unobservable directions.

The observability of the parameter space and robustness to noise and unobservable directions is one of the principal components of FastCal, Figure 7.6 shows the evolution of observability over translation (left) and rotation (right) over a trajectory of 600ft. The robotic platform only moves on planar surfaces, as such there is no excitation on roll or pitch, as well as along the axis perpendicular to the ground (y). Classic observability analysis [52] tells us that only the y component (vertical component) between the cameras is unobservable; x, z, roll, pitch and yaw are observable. Figure 7.6 shows how rotation quickly becomes observable, but translation requires the robot to perform a few turns, and the y component is permanently unobservable, as predicted. The observability score here is computed using the formulation in 7.2.3, obtaining the nullspace of the information matrix and obtaining the maximum nullspace component along each direction. In Figure 7.3 we compare the entropy of the priority queue over time, compared to the entropy of the reference implementation priority queue. FastCal's entropy monotonically decreases, in part due to the novel segment adding criteria which aims to find local minimum in the candidate window search. There is no guarantee that the priority queue entropy will reduce by swapping a single candidate

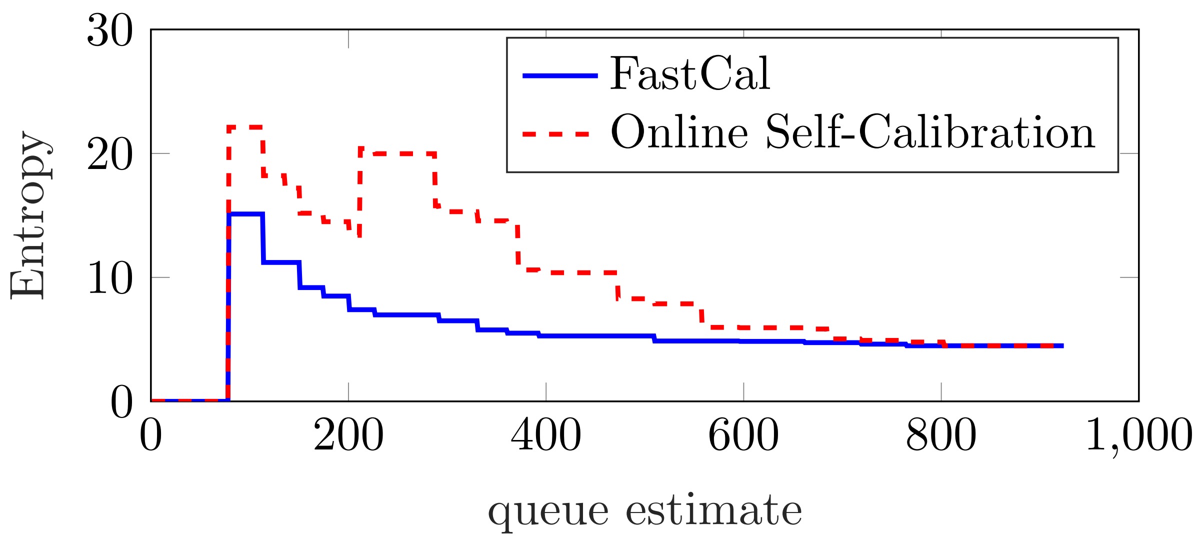


Figure 7.7: Priority Queue entropy comparison of the proposed FastCal algorithm vs a implementation of [115]. The novel criteria on adding segments to the queue, coupled with the keyframing on measurements results in a much faster reduction in entropy and thus convergence on the calibration parameters.

segment, as that would require a costly mutual information estimate every time a new candidate segment is assessed. In the reference implementation of [115] the smaller measurement sizes and the more frequent swaps in the priority queue result in much slower convergence.

7.4 Summary

In this chapter we presented a low time complexity extrinsics self-calibration algorithm that uses an information theoretic measure to add only the most locally informative measurement batches to the estimation queue, the work from [115] is improved upon by leveraging novel queue update techniques which drastically reduce the time it takes to converge on the calibration parameters. We have leveraged truncated SVD/QR decomposition to deal with unobservable directions and operate on nonholonomic platforms, similar to the algorithm presented in [113], however not applied to reinforcement learning and user motion suggestions, but to calibrating nonholonomic robotic platforms. Finally, we have incorporated time-decay as a mechanism to address drift in calibration parameters. We draw from the lessons learned from [116] to implement a simpler, more effective mechanism of dealing with drift. We have evaluated the proposed system in a variety of long term scenarios spanning 15 days and over 120km driven and shown that it can be used as a low-overhead add-on to existing perception systems, while achieving similar accuracy to off-line calibration techniques. Our principal goal was to develop a robust, low time complexity self-calibration algorithm for sensor extrinsics, and show its usefulness in practical long term robotic applications. We have shown that through the decoupling of the estimation problem into two steps, and selectively adding new segments to the priority queue we are able to achieve robust and accurate calibration results with minimal compute overhead. An argument could be made that calibration parameters do not vary enough to justify the addition of self-calibration, but we argue that for true long-term autonomy applications, robust self-calibration is essential, as even in relatively short experiments there was significant change in sensor extrinsics.

Chapter 8

Summary and Future Work

This chapter presents a short summary of the contributions emanating from this thesis, together with references to papers published describing this research. We then conclude by suggesting promising directions for future work.

8.1 Contributions

Chapter 3 presents online, constant-time self-calibration and change detection with re-calibration for joint estimation of camera-to-IMU transform and camera intrinsic parameters, using only naturally occurring features. The system is evaluated with experimental data and shown to converge to offline calibration estimates with centimeter level accuracy for camera-to-IMU translation, and sub-degree accuracy for rotation. The statistical change detection framework presented in [68] and summarized in Section 3.2.3 has been extended to the camera-to-IMU transform, including a statistical comparison of distributions over candidate segments for a SE(3) pose.

The use of an adaptive conditioning window for re-estimation of past poses allows this framework to operate in long-term applications where the accumulation of linearization errors in a prior distribution would lead to significant drift. We presented a framework that supports adding additional sensors while maintaining online operation. To the authors' best knowledge this is the first application of multi-sensor self-calibration with automatic change detection and re-estimation of parameters.

In Chapter 4 the work from Chapter 3 is extended to deal explicitly with the case of drifting calibration parameters over long trajectories. The system is evaluated with experimental and simulated data and shown to converge to offline calibration estimates even in the presence of slowly drifting calibration parameters. The statistical change detection framework presented initially in [68] is used to detect change regions for drifting parameters and estimate the calibration parameters in the drift region.

The use of a drift correcting self-calibrating framework coupled with adaptive conditioning window for re-estimation of past poses allows this framework to operate in long-term applications where the accumulation of linearization errors in a prior distribution and the accumulation of incorrectly estimated calibration parameters over change periods would lead to significant drift. We present an analysis on the effects of inappropriate modeling of calibration parameters over long trajectories, and show how the use of a multivariate probabilistic change detection framework can greatly reduce the drift even in the presence of hard-to-detect incremental changes over time in calibration parameters.

Chapter 5 further builds on the self-calibration pipeline by presenting a novel user-assisted calibration tool that uses reinforcement learning to provide live feedback on the state of calibration and produces accurate calibration parameters even when used by inexperienced operators. We have leveraged truncated SVD/QR decomposition to deal with unobservable directions and reinforcement learning for motion suggestions to perform model-free calibration for both camera intrinsics and camera-to-IMU extrinsics. An evaluation of the proposed system in a variety of scenarios has shown that it can be used as a replacement for currently available calibration toolkits. Our principal question was to assess the suitability of reinforcement learning when applied to the calibration problem. We have shown that through the discretization of both the action and state we are able to leverage Q-learning to improve the calibration experience. An interesting result that lends itself to long term autonomy is the ability to capture different sensor configurations. As shown in Figure 5.3 having radically different calibration parameters results in a different set of optimal motions, reinforcing the point that the “SLAM wobble” or any pre-determined calibration maneuver is not guaranteed to provide acceptable parameter inference. If a robot is expected to calibrate itself autonomously it cannot have a pre-determined routine or hope that random navigation will render its calibration parameters observable. A system that is robust to sensor configuration changes by re-learning how to calibrate itself is a step in the direction of both “power-on-and-go” robotics and long term autonomy.

Chapter 6 changes the perspective from intrinsic robustness (such as calibration parameters) to extrinsic robustness: changes in the environment the robot operates in. A general formulation for feature persistence is presented which makes use of correlation between feature persistence imposed by a joint prior distribution which may be learned or engineered. The key insight of this work is proposing a joint distribution over feature persistence which is computationally tractable in constant time. Our proposed formulation improves upon prior work on modeling individual feature persistence by demonstrating the necessity of a joint model when the environment is subject to change. Our approach allows for the use of the survival time prior distributions discussed in [126] while incorporating information about environmental structure. We show approximated constant-time online inference over feature persistence in a graph-slam environment in both simulated and real scenarios, subject to landmark change. The use of persistence for informed data associations allows navigating through challenging dynamic environments where considering the joint feature persistence is essential.

Finally Chapter 7 Draws from and extends Chapters 3, 4 and 5 to work in union on a robotic platform deployed in a long-term autonomy setting. We present a low time complexity extrinsics self-calibration algorithm that uses an information theoretic measure to add only the most locally informative measurement batches to the estimation queue, the work from Chapter 3 is improved upon by leveraging novel queue update techniques which drastically reduce the time it takes to converge on the calibration parameters. We have leveraged truncated SVD/QR decomposition to deal with unobservable directions and operate on nonholonomic platforms, similar to the algorithm presented in 5, however not applied to reinforcement learning and user motion suggestions, but to calibrating nonholonomic robotic platforms. Finally, we have incorporated time-decay as a mechanism to address drift in calibration parameters. We draw from the lessons learned from Chapter 4 to implement a simpler, more effective mechanism of dealing with drift. We have evaluated the proposed system in a variety of long term scenarios spanning 15 days and over 120km driven and shown that it can be used as a low-overhead add-on to existing perception systems, while achieving similar accuracy to off-line calibration techniques. Our principal goal was to develop a robust, low time complexity self-calibration algorithm for sensor extrinsics, and show its usefulness in practical long term robotic applications. We have shown that through the decoupling of the estimation problem into two steps, and selectively adding new segments to the priority queue we are able to achieve robust and accurate calibration results with minimal compute overhead. An argument could be made that calibration parameters do not vary enough to justify the addition of self-calibration, but we argue that for true long-term autonomy applications, robust self-calibration is essential, as even in relatively short experiments there was significant change in sensor extrinsics.

8.2 Future Work

Here we discuss some of the open problems and research questions related to robustness in life-long SLAM which are natural extensions of the contributions in this thesis:

Metric Relocalization: On the topic of loop-closures and relocalization, appearance-based methods [21, 102] are able to close loops between day and night sequences, or even between seasons [110], the resulting loop closure is topological in nature, which means that while they are able to identify that the same location is being re-visited, they do not estimate a relative pose with respect to the previously build map. For metric relocalization, feature-based approaches are still the norm [44], but cannot be extended to the circumstances described above. Vision has become the sensor of choice for most loop-closing applications, which has become a sensor signature matching problem [25, 26, 44, 142], other sensors might be exploited. For instance, Brubaker **et al.** [17] uses trajectory matching to circumvent the shortcomings of cameras. Other approaches map with one sensor modality but localize in the same map with a different sensor modality. Wolcott **et al.** [149] and Foster **et al.** [37] take steps in that direction. [37] studies vision-based localization in a lidar generated map. Majdik **et al.** [96] studies how to localize a drone in a cadastral 3D model textured from Google Street View images, Behzadin **et al.** [11] shows how to localize in hand-drawn maps using a laser scanner and finally Winterhalter **et al.** [148] does RGB-D localization given 2D floor plans. A relocalization approach that provides metric loop closure on deformable, time-varying maps is still an important research topic and is essential for life-long SLAM.

Deformable Maps: As described in section 2.2, most mainstream SLAM algorithms have been designed with a rigid and static world assumption; however the real world is very much non-rigid due to dynamics and the inherent deformability of most objects. A SLAM solution for long term applications should be able to handle dynamics in the environment, such as non-rigidity. The computer vision community has work dating back to the 80s attempting to recover shape from non-rigid objects, but still with restrictive applicability. For example, Pentland **et al.** [120, 119] requires prior knowledge of some mechanical properties of objects, Bregler **et al.** [15, 144] relies on a restricted deformation of the object, and shows examples using the human face. More recent work in non-rigid Structure from Motion (SfM) such as [6, 5, 47] are less restrictive in their assumptions, but still only work in small scenarios. In the SLAM community, Newcombe **et al.** [111] addressed the non-rigid case for small-scale reconstructions. Even though considerable progress has been made towards non-rigid maps, considerable work needs to be done on addressing **non-rigid maps at a large scale**, which is still largely unexplored.

Failsafe SLAM and recovery: Despite the considerable progress made on the SLAM back-end, modern SLAM solvers are still vulnerable in the presence of outliers. This is mainly due to the fact that all SLAM techniques are based on iterative optimization over a non-convex cost surface. This causes the outlier rejection outcome to depend on the quality of the initial guess, and the system is also inherently fragile: despite robust techniques, the inclusion of a single outlier can degrade the quality of the estimate, which has the snowball effect of degrading the capability of discerning outliers later on. An ideal SLAM solution for long-term applications needs to be **fail-safe** and **failure-aware**, i.e., the SLAM system needs to be aware of imminent failure and provide recovery mechanisms that can re-establish normal operation. These capabilities are not present in any existing SLAM approach.

Hardware Failure: If the accuracy of the sensor degrades due to malfunction, aging, etc, the quality of the sensor measurements (e.g., the noise, bias) will not match the noise model used in the back-end (2.3) which leads to poor estimates. This leads to two important research questions: “How can we detect degraded sensor operation?” and “How can we dynamically adjust the sensor noise statistics accordingly?”.

Automatic parameter tuning: Most SLAM systems, in particular the data association module, require extensive parameter tuning in order to work correctly for a given scenario. For feature-based visual-slam, for example, the number of features in an image, the number of pyramid levels, the normalized cross-correlation (NCC) threshold are just a few of the possible tuning parameters that need to be adjusted for optimal performance in a given scenario. RANSAC parameters and the criteria to decide when to add new factors (keyframe) to the graph, or when to trigger a loop closure detection are a few other examples in a myriad of parameters. In the ideal ”power-on-and-go” SLAM system, for arbitrary scenarios, methods for automatic tuning of the involved parameters need to be developed, especially if the system is expected to operate for long periods of time over varying conditions.

Bibliography

[1] An observability-constrained sliding window filter for SLAM. IEEE, 2011. 18

[2] P A Absil, R Mahony, and R Sepulchre. Optimization algorithms on matrix manifolds, 2009. 16

[3] Pratik Agarwal, Gian Diego Tipaldi, Luciano Spinello, Cyrill Stachniss, and Wolfram Burgard. Robust map optimization using dynamic covariance scaling. In International Conference on Robotics and Automation, pages 62–69. IEEE, 2013. 6

[4] Sameer Agarwal, Keir Mierle, et al. Ceres solver, 2012. 16, 85

[5] A Agudo, F Moreno-Noguer, and B Calvo. Real-Time 3D Reconstruction of Non-Rigid Shapes with a Single Moving Camera. Computer Vision and ..., 2016. 111

[6] A Agudo, F Moreno-Noguer, and B Calvo. Sequential non-rigid structure from motion using physical priors. Transactions on Pattern Analysis and Machine Intelligence, 38(5):979–994, 2016. 111

[7] J Aulinas, Y R Petillot, J Salvi, and X Lladó. The SLAM problem: a survey. CCIA, 2008. 11

[8] Autonomous Robotics and Perception Group (ARPG). VICalib visual-inertial calibration suite. `https://github.com/arpg/vicalib`, 2016. 51, 65

[9] T Bailey and H Durrant-Whyte. Simultaneous localization and mapping (SLAM): Part II. Robotics & Automation, 2006. 10, 11, 18

[10] Chris Beall, Brian J Lawrence, Viorela Ila, and Frank Dellaert. 3d reconstruction of underwater structures. In Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on, pages 4418–4423. IEEE, 2010. 5

[11] Bahram Behzadian, Pratik Agarwal, Wolfram Burgard, and Gian Diego Tipaldi. Monte carlo localization in hand-drawn maps. In Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on, pages 4291–4296. IEEE, 2015. 111

[12] Peter Biber and Tom Duckett. Experimental analysis of sample-based maps for long-term slam. The International Journal of Robotics Research, 28(1):20–33, 2009. 72

[13] Peter Biber, Tom Duckett, et al. Dynamic maps for long-term operation of mobile service robots. In Robotics: science and systems, pages 17–24, 2005. 72

[14] F Bonin-Font, A Ortiz, and G Oliver. Visual navigation for mobile robots: A survey. Journal of Intelligent & Robotic Systems, 53(3):263–296, 2008. 11

[15] Christoph Bregler, Aaron Hertzmann, and Henning Biermann. Recovering non-rigid 3d shape from image streams. In Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on, volume 2, pages 690–696. IEEE, 2000. 111

[16] Jonathan Brookshire and Seth Teller. Extrinsic calibration from per-sensor egomotion. Robotics: Science and Systems VIII, Jul, pages 504–512, 2013. 54, 92

[17] M A Brubaker, A Geiger, and Raquel Urtasun. Lost! leveraging the crowd for probabilistic visual self-localization. In Proceedings of the IEEE . . . , pages 3057–3064. IEEE, 2013. 110

[18] Luca Carlone, Andrea Censi, and Frank Dellaert. Selecting good measurements via 1 relaxation: A convex approach for robust estimation over graphs. In 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 2667–2674. IEEE, 2014. 6

[19] Luca Carlone, Roberto Tron, Kostas Daniilidis, and Frank Dellaert. Initialization techniques for 3D SLAM: A survey on rotation estimation and its use in pose graph optimization. In International Conference on Robotics and Automation, pages 4597–4604. IEEE, 2015. 11, 22, 36

[20] Winston Churchill and Paul Newman. Practice makes perfect? managing and leveraging visual experiences for lifelong navigation. In Robotics and Automation (ICRA), 2012 IEEE International Conference on, pages 4525–4532. IEEE, 2012. 72

[21] Winston Churchill and Paul Newman. Experience-based navigation for long-term localisation. The International Journal of Robotics Research, 32(14):1645–1661, 2013. 6, 110

[22] J Civera, DR Bueno, AJ Davison, and JMM Montiel. Camera self-calibration for sequential bayesian structure from motion. pages 403–408. IEEE, 2009. 8, 34

[23] Javier Civera, Diana R Bueno, Andrew J Davison, and J M MartÍnez Montiel. Camera self-calibration for sequential Bayesian structure from motion. In International Conference on Robotics and Automation, pages 403–408. IEEE, 2009. 8, 20, 34, 92

[24] Peter Corke, Jorge Lobo, and Jorge Dias. An Introduction to Inertial and Visual Sensing. International Journal of Robotics Research, 26(6):519–535, June 2007. 7

[25] Mark Cummins and Paul Newman. FAB-MAP: Probabilistic localization and mapping in the space of appearance. The International Journal of Robotics . . . , 2008. 5, 110

[26] Mark Cummins and Paul Newman. Highly scalable appearance-only slam-fab-map 2.0. In Robotics: Science and Systems, volume 5, page 17. Seattle, USA, 2009. 110

[27] F Dayoub, G Cielniak, and T Duckett. Long-term experiments with an adaptive spherical view representation for navigation in changing environments. Robotics and Autonomous Systems, 2011. 7

[28] Frank Dellaert. Monte-Carlo EM for data-association and its applications in computer vision. PhD thesis, Carnegie Mellon University Pittsburgh, PA, 2001. 73

[29] Frank Dellaert. Factor Graphs and GTSAM: A Hands-on Introduction. September 2012. 16

[30] Frank Dellaert. Factor graphs and GTSAM: A hands-on introduction. Technical report, Georgia Institute of Technology, 2012. 72

[31] G Dissanayake, S Huang, and Z Wang. A review of recent developments in simultaneous localization and mapping. 2011 6th International ..., pages 477–482, 2011. 10, 11

[32] Tue-Cuong Dong-Si and Anastasios I Mourikis. Estimator initialization in vision-aided inertial navigation with unknown camera-IMU calibration. In Intelligent Robots and Systems, pages 1064–1071. IEEE, 2012. 7, 22, 23, 36, 37, 38

[33] H Durrant-Whyte and T Bailey. Simultaneous localization and mapping: part I. Robotics & Automation, 13(2):99–110, June 2006. 10, 11, 18

[34] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part i. IEEE robotics & automation magazine, 13(2):99–110, 2006. 54, 55

[35] Ryan M Eustice, Hanumant Singh, John J Leonard, and Matthew R Walter. Visually Mapping the RMS Titanic: Conservative Covariance Estimates for SLAM Information Filters. International Journal of Robotics Research, 25(12):1223–1242, December 2006. 5

[36] Nathaniel Fairfield, George Kantor, and David Wettergreen. Real-time slam with octree evidence grids for exploration in underwater tunnels. Journal of Field Robotics, 24(1-2):03–21, 2007. 5

[37] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. Air-ground localization and map augmentation using monocular dense reconstruction. In 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 3971–3978. IEEE, 2013. 110

[38] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. SVO: Fast semi-direct monocular visual odometry. In International Conference on Robotics and Automation, pages 15–22. IEEE, 2014. 41

[39] J-M Frahm and Reinhard Koch. Camera calibration with known rotation. In Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on, pages 1418–1425. IEEE, 2003. 7, 32

[40] Vincent François-Lavet, Raphael Fonteneau, and Damien Ernst. How to discount deep reinforcement learning: Towards new dynamic strategies. arXiv preprint arXiv:1512.02011, 2015. 62

[41] F Fraundorfer and D Scaramuzza. Visual odometry: Part i: The first 30 years and fundamentals. IEEE Robotics and Automation Magazine, 18(4):80–92, 2011. 6, 19

[42] Udo Frese. Interview: Is slam solved? KI-Künstliche Intelligenz, 24(3):255–257, 2010. 3

[43] J Fuentes-Pacheco and J Ruiz-Ascencio. Visual simultaneous localization and mapping: a survey. Artificial Intelligence Review, 2012. 11

[44] Dorian Gálvez-López and Juan D Tardos. Bags of binary words for fast place recognition in image sequences. IEEE Transactions on Robotics, 28(5):1188–1197, 2012. 5, 110

[45] Bruce P Gibbs. Advanced Kalman filtering, least-squares and modeling: a practical handbook. John Wiley & Sons, 2011. 56

[46] Gene H Golub and Charles F Van Loan. Matrix computations, volume 3. JHU Press, 2012. 57

[47] Oscar G Grasa, Ernesto Bernal, Santiago Casado, Ismael Gil, and JMM Montiel. Visual slam for handheld monocular endoscope. IEEE transactions on medical imaging, 33(1):135–146, 2014. 111

[48] Giorgio Grisetti, Rainer Kummerle, Cyrill Stachniss, and Wolfram Burgard. A Tutorial on Graph-Based SLAM. IEEE Intelligent Transportation Systems Magazine, 2(4):31–43, 2010. 11

[49] J S Gutmann and K Konolige. Incremental mapping of large cyclic environments. In Computational Intelligence in Robotics and Automation, 1999. CIRA '99. Proceedings. 1999 IEEE International Symposium on, pages 318–325. IEEE, 1999. 12

[50] Per Christian Hansen. Rank-deficient and discrete ill-posed problems: numerical aspects of linear inversion. SIAM, 1998. 56, 57, 100, 101

[51] Karol Hausman, James Preiss, Gaurav S Sukhatme, and Stephan Weiss. Observability-Aware Trajectory Optimization for Self-Calibration with Application to UAVs. arXiv.org, April 2016. 51

[52] Robert Hermann and Arthur Krener. Nonlinear controllability and observability. IEEE Transactions on automatic control, 22(5):728–740, 1977. 56, 100, 104

[53] J A Hesch, D G Kottas, and S L Bowman. Camera-IMU-based localization: Observability analysis and consistency improvement. ... International Journal of ..., 2013. 18

[54] J A Hesch, D G Kottas, S L Bowman, and Stergios I Roumeliotis. Camera-IMU-based localization: Observability analysis and consistency improvement. International Journal of Robotics Research, 33(1):182–201, January 2014. 18

[55] Joel A Hesch, Dimitrios G Kottas, Sean L Bowman, and Stergios I Roumeliotis. Towards Consistent Vision-Aided Inertial Navigation. Algorithmic Foundations of Robotics, 86(Chapter 34):559–574, 2013. 8, 20, 34, 92

[56] A. Heyden and K. Astrom. Euclidean reconstruction from image sequences with varying and unknown focal length and principal point. In IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pages 438–443. Institute of Electrical Engineers, Inc (IEEE), 1997. 7, 32

[57] Kin Leong Ho and Paul Newman. Loop closure detection in slam by combining visual and spatial appearance. Robotics and Autonomous Systems, 54(9):740–749, 2006. 6

[58] S Huang and G Dissanayake. A critique of current developments in simultaneous localization and mapping. International Journal of Advanced Robotic Systems, 13(5):1–13, September 2016. 11

[59] P J Huber and E M Ronchetti. Robust Statistics. 2009. Wiley. 15

[60] Claude Jauffret. Observability and fisher information matrix in nonlinear regression. IEEE Transactions on Aerospace and Electronic Systems, 43(2), 2007. 52, 56, 100

[61] Eagle S Jones and Stefano Soatto. Visual-inertial navigation, mapping and localization: A scalable real-time causal approach. International Journal of Robotics Research, 30(4):407–430, April 2011. 7, 20, 22, 32, 37, 92

[62] Michael Kaess and Frank Dellaert. Covariance recovery from a square root information matrix for data association. Robotics and autonomous systems, 57(12):1198–1210, 2009. 85

[63] Michael Kaess, Hordur Johannsson, Richard Roberts, Viorela Ila, John J Leonard, and Frank Dellaert. iSAM2: Incremental smoothing and mapping with fluid relinearization and incremental variable reordering. In International Conference on Robotics and Automation, pages 3281–3288. IEEE, 2011. 16

[64] Michael Kaess, Hordur Johannsson, Richard Roberts, Viorela Ila, John J Leonard, and Frank Dellaert. isam2: Incremental smoothing and mapping using the Bayes tree. The International Journal of Robotics Research, 31(2):216–235, 2012. 72

[65] Michael Kaess, Ananth Ranganathan, and Frank Dellaert. iSAM: Incremental Smoothing and Mapping. Transactions on Robotics, 24(6):1365–1378, December 2008. 16

[66] Nima Keivan and Gabe Sibley. Constant-time monocular self-calibration. Robotics and Biomimetics (ROBIO), pages 1590–1595, 2014. 7, 22, 23, 34, 56, 93, 97, 98

[67] Nima Keivan and Gabe Sibley. Asynchronous adaptive conditioning for visual-inertial SLAM. International Journal of Robotics Research, 34(13):1573–1589, 2015. 22, 26, 35, 41

[68] Nima Keivan and Gabe Sibley. Online SLAM with any-time self-calibration and automatic change detection. In International Conference on Robotics and Automation, pages 5775–5782. IEEE, 2015. 7, 8, 22, 25, 30, 31, 34, 35, 38, 40, 44, 46, 107, 108

[69] Jonathan Kelly and Gaurav S Sukhatme. Visual-Inertial Sensor Fusion: Localization, Mapping and Sensor-to-Sensor Self-calibration. International Journal of Robotics Research, 30(1):56–79, January 2011. 7, 20, 22, 32, 37, 92

[70] Jonathan Kelly and Gaurav S Sukhatme. Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration. The International Journal of Robotics Research, 30(1):56–79, 2011. 49

[71] Ayoung Kim and Ryan M Eustice. Real-time visual slam for autonomous underwater hull inspection using visual saliency. IEEE Transactions on Robotics, 29(3):719–733, 2013. 5

[72] Georg Klein and David Murray. Parallel Tracking and Mapping for Small AR Workspaces. In International Symposium on Mixed and Augmented Reality, pages 225–234. IEEE, 2007. 35

[73] Kurt Konolige and James Bowman. Towards lifelong visual maps. In Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on, pages 1156–1163. IEEE, 2009. 72

[74] Kurt Konolige, James Bowman, JD Chen, Patrick Mihelich, Michael Calonder, Vincent Lepetit, and Pascal Fua. View-based maps. The International Journal of Robotics Research, 29(8):941–957, 2010. 7, 72

[75] Dimitrios G Kottas, Joel A Hesch, Sean L Bowman, and Stergios I Roumeliotis. On the Consistency of Vision-Aided Inertial Navigation. In Experimental Robotics, pages 303–317. Springer International Publishing, Heidelberg, 2013. 18

[76] T Krajník, J P Fentanes, and O M Mozos. Long-term topological localisation for service robots in dynamic environments using spectral maps. Intelligent Robots and ..., pages 4537–4542, 2014. 7

[77] Tomas Krajnik, Jaime Pulido Fentanes, Grzegorz Cielniak, Christian Dondrup, and Tom Duckett. Spectral analysis for long-term robotic mapping. In Robotics and Automation (ICRA), 2014 IEEE International Conference on, pages 3706–3711. IEEE, 2014. 73

[78] Frank R Kschischang, Brendan J Frey, and H-A Loeliger. Factor graphs and the sum-product algorithm. IEEE Transactions on information theory, 47(2):498–519, 2001. 12

[79] Rainer Kümmerle, Giorgio Grisetti, and Wolfram Burgard. Simultaneous calibration, localization, and mapping. In Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on, pages 3716–3721. IEEE, 2011. 54, 92

[80] Rainer Kummerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. G¡sup¿2¡/sup¿o: A general framework for graph optimization. In International Conference on Robotics and Automation, pages 3607–3613. IEEE, 2011. 16

[81] Y Latif, C Cadena, and J Neira. Robust loop closing over time. Robotics: Science and Systems VIII, 2013. 6

[82] Keith YK Leung, Yoni Halpern, Timothy D Barfoot, and Hugh HT Liu. The utias multi-robot cooperative localization and mapping dataset. The International Journal of Robotics Research, 30(8):969–974, 2011. 86, 87

[83] Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart, and Paul Furgale. Keyframe-based visual–inertial odometry using nonlinear optimization. The International Journal of Robotics Research, 34(3):314–334, 2015. 54

[84] Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart, and Paul Furgale. Keyframe-based visual–inertial odometry using nonlinear optimization. International Journal of Robotics Research, 34(3):314–334, March 2015. 23

[85] Jesse Levinson and Sebastian Thrun. Unsupervised calibration for multi-beam lasers. In Experimental Robotics, pages 179–193. Springer, 2014. 54, 92

[86] Mingyang Li and Anastasios I Mourikis. 3-D motion estimation and online temporal calibration for camera-IMU systems. In International Conference on Robotics and Automation, pages 5709–5716. IEEE, 2013. 23

[87] Mingyang Li and Anastasios I Mourikis. High-precision, consistent EKF-based visual–inertial odometry. International Journal of Robotics Research, 32(6):690–711, May 2013. 8, 20, 34, 92

[88] Mingyang Li, Hongsheng Yu, Xing Zheng, and Anastasios I Mourikis. High-fidelity sensor modeling and self-calibration in vision-aided inertial navigation. In International Conference on Robotics and Automation, pages 409–416. IEEE, 2014. 8, 20, 23, 34, 92

[89] Mingyang Li, Hongsheng Yu, Xing Zheng, and Anastasios I Mourikis. High-fidelity sensor modeling and self-calibration in vision-aided inertial navigation. In 2014 IEEE International Conference on Robotics and Automation (ICRA), pages 409–416. IEEE, 2014. 8, 34

[90] Kin-Huat Low. Industrial robotics: programming, simulation and applications. I-Tech, 2007. 56

[91] Stephanie M Lowry, Niko Sunderhauf, Paul Newman, John J Leonard, David D Cox, Peter I Corke, and Michael Milford. Visual Place Recognition: A Survey. Transactions on Robotics, 32(1):1–19, 2016. 6, 11

[92] Feng Lu and Evangelos Milios. Globally consistent range scan alignment for environment mapping. Autonomous robots, 4(4):333–349, 1997. 12

[93] Todd Lupton and Salah Sukkarieh. Removing scale biases and ambiguity from 6DoF monocular SLAM using inertial. In International Conference on Robotics and Automation, pages 3698–3703. IEEE, 2008. 7

[94] S Lynen, T Sattler, M Bosse, and J A Hesch. Get Out of My Lab: Large-scale, Real-Time Visual-Inertial Localization. In Robotics Science and Systems, 2015. 7

[95] Ian Mahon, Stefan B Williams, Oscar Pizarro, and Matthew Johnson-Roberson. Efficient View-Based SLAM Using Visual Loop Closures. Transactions on Robotics, 24(5):1002–1014, 2008. 11

[96] András L Majdik, Damiano Verda, Yves Albers-Schoenberg, and Davide Scaramuzza. Air-ground matching: Appearance-based gps-denied urban localization of micro aerial vehicles. Journal of Field Robotics, 32(7):1015–1039, 2015. 111

[97] Agostino Martinelli. Vision and IMU Data Fusion: Closed-Form Solutions for Attitude, Speed, Absolute Scale, and Bias Determination. Transactions on Robotics, 28(1):44–60, 2012. 7

[98] Agostino Martinelli, Davide Scaramuzza, and Roland Siegwart. Automatic self-calibration of a vision system during robot motion. In Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on, pages 43–48. IEEE, 2006. 54, 92

[99] Jérôme Maye, Hannes Sommer, Gabriel Agamennoni, Roland Siegwart, and Paul Furgale. Online self-calibration for robotic systems. International Journal of Robotics Research, 35(4):0278364915596232–380, October 2015. 7

[100] Christopher Mei, Gabe Sibley, Mark Cummins, Paul Newman, and Ian Reid. RSLAM: A System for Large-Scale Mapping in Constant-Time Using Stereo. International Journal of Computer Vision, 94(2):198–214, June 2010. 41

[101] Daniel Meyer-Delius, Maximilian Beinhofer, and Wolfram Burgard. Occupancy grid models for robot mapping in changing environments. In AAAI, 2012. 72

[102] Michael J Milford and Gordon F Wyeth. Seqslam: Visual route-based navigation for sunny summer days and stormy winter nights. In Robotics and Automation (ICRA), 2012 IEEE International Conference on, pages 1643–1649. IEEE, 2012. 6, 110

[103] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. Nature, 518(7540):529–533, 2015. 68

[104] Hans Moravec and Alberto Elfes. High resolution maps from wide angle sonar. In Robotics and Automation. Proceedings. 1985 IEEE International Conference on, volume 2, pages 116–121. IEEE, 1985. 73

[105] Timothy Morris, Feras Dayoub, Peter Corke, Gordon Wyeth, and Ben Upcroft. Multiple map hypotheses for planning and navigating in non-stationary environments. In Robotics and Automation (ICRA), 2014 IEEE International Conference on, pages 2765–2770. IEEE, 2014. 72

[106] Anastasios I Mourikis and Stergios I Roumeliotis. A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation. In International Conference on Robotics and Automation, pages 3565–3572. IEEE, 2007. 7, 18, 20, 23, 32, 92

[107] Richard M Murray, Zexiang Li, S Shankar Sastry, and S Shankara Sastry. A mathematical introduction to robotic manipulation. CRC press, 1994. 130, 131, 133

[108] Jos Neira, Andrew J Davison, and John J Leonard. Guest Editorial Special Issue on Visual SLAM. Transactions on Robotics, 24(5):929–931, 2008. 11

[109] José Neira and Juan D Tardós. Data association in stochastic mapping using the joint compatibility test. IEEE Transactions on robotics and automation, 17(6):890–897, 2001. 7, 69, 74

[110] P Neubert and N Sünderhauf. Appearance change prediction for long-term navigation across seasons. Mobile Robots (ECMR), pages 198–203, 2013. 110

[111] Richard A Newcombe, Dieter Fox, and Steven M Seitz. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 343–352, 2015. 111

[112] David Nister and H Stewénius. Scalable Recognition with a Vocabulary Tree. In Computer Vision and Pattern Recognition, pages 2161–2168. IEEE, 2006. 5

[113] Fernando Nobre and Christoffer Heckman. Reinforcement learning for assisted visual-inertial robotic calibration. In International Symposium on Robotics Research (ISRR)., 2017. 9, 90, 99, 105

[114] Fernando Nobre, Christoffer Heckman, Paul Ozog, Ryan Wolcott, and Jeffrey Walls. Probabilistic change detection in feature-based maps. In International Conference on Robotics and Automation (ICRA). IEEE, 2018. 9

[115] Fernando Nobre, Christoffer Heckman, and Gabe Sibley. Multi-sensor slam with online self-calibration and change detection. In International Symposium on Experimental Robotics (ISER)., 2016. 7, 8, 9, 32, 34, 35, 44, 52, 56, 90, 93, 97, 98, 103, 105

[116] Fernando Nobre, Michael Kasper, and Christoffer Heckman. Drift-correcting self-calibration for visual-inertial slam. In International Conference on Robotics and Automation (ICRA). IEEE, 2017. 9, 90, 101, 105

[117] Edwin Olson and Pratik Agarwal. Inference on networks of mixtures for robust robot mapping. The International Journal of Robotics Research, 32(7):826–840, 2013. 6

[118] Edwin Olson, Matthew R Walter, Seth J Teller, and John J Leonard. Single-cluster spectral graph partitioning for robotics applications. In Robotics: Science and Systems, pages 265–272, 2005. 7

[119] Alex Pentland and John Williams. Good vibrations: Modal dynamics for graphics and animation, volume 23. ACM, 1989. 111

[120] Alex Petland and Bradley Horowitz. Recovery of nonrigid motion and structure. IEEE Transactions on Pattern Analysis and Machine Intelligence, 13(7):730–742, 1991. 111

[121] Marc Pollefeys, Reinhard Koch, and Luc Van Gool. Self-calibration and metric reconstruction in spite of varying and unknown internal camera parameters. In International Journal of Computer Vision, pages 7–25, 1999. 7, 32

[122] L Polok, M Solony, Viorela Ila, P Smrz, and P Zemcik. Incremental Cholesky factorization for least squares problems in robotics. IFAC Proceedings Volumes, 2013. 16

[123] Andrew Richardson, Johannes Strom, and Edwin Olson. Aprilcal: Assisted and repeatable camera calibration. In Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on, pages 1814–1821. IEEE, 2013. 52

[124] G Ros, A Sappa, D Ponsa, and A M Lopez. Visual slam for driverless cars: A brief survey. In Intelligent Vehicles Symposium, 2012. 11

[125] D M Rosen, J Mason, and John J Leonard. Towards Lifelong Feature-Based Mapping in Semi-Static Environments. ... Conference on Robotics ..., pages 1063–1070, 2016. 7

[126] David M Rosen, Julian Mason, and John J Leonard. Towards lifelong feature-based mapping in semi-static environments. In Robotics and Automation (ICRA), 2016 IEEE International Conference on, pages 1063–1070. IEEE, 2016. 71, 73, 74, 76, 77, 78, 79, 86, 88, 109

[127] Jari Saarinen, Henrik Andreasson, and Achim J Lilienthal. Independent markov chain occupancy grid maps for representation of dynamic environment. In Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on, pages 3489–3495. IEEE, 2012. 73

[128] Deon Sabatta, Davide Scaramuzza, and Roland Siegwart. Improved appearance-based matching in similar and dynamic environments using a vocabulary tree. 2010. 7

[129] Sajad Saeedi, Michael Trentini, Mae Seto, and Howard Li. Multiple-Robot Simultaneous Localization and Mapping: A Review. Journal of Field Robotics, 33(1):3–46, January 2016. 11

[130] Renato F Salas-Moreno, Richard Newcombe, Hauke Strasdat, Paul H J Kelly, and Andrew J Davison. SLAM++: Simultaneous Localisation and Mapping at the Level of Objects. In Computer Vision and Pattern Recognition, pages 1352–1359. IEEE, May 2013. 16

[131] Joao Machado Santos, Tomáš Krajník, Jaime Pulido Fentanes, and Tom Duckett. Lifelong information-driven exploration to complete and refine 4-d spatio-temporal maps. IEEE Robotics and Automation Letters, 1(2):684–691, 2016. 7

[132] Davide Scaramuzza and F Fraundorfer. Visual odometry: Part II: Matching, robustness, optimization, and applications. IEEE Robotics & Automation Magazine, 19(2):78–90, 2012. 11

[133] Davide Scaramuzza and Friedrich Fraundorfer. Visual Odometry [Tutorial]. IEEE Robotics & Automation Magazine, 18(4):80–92, December 2011. 5, 11

[134] Mark Sheehan, Alastair Harrison, and Paul Newman. Self-calibration for a 3d laser. The International Journal of Robotics Research, 31(5):675–687, 2012. 54, 92

[135] Gabe Sibley, Larry Matthies, and Gaurav Sukhatme. Sliding window filter with application to planetary landing. Journal of Field Robotics, 27(5):587–608, 2010. 85

[136] Gabe Sibley, Larry Matthies, and Gaurav S Sukhatme. Sliding window filter with application to planetary landing. Journal of Field Robotics, 27(5):587–608, July 2010. 16

[137] J Sivic and A Zisserman. Video Google: a text retrieval approach to object matching in videos. In International Conference on Computer Vision, pages 1470–1477 vol.2. IEEE, 2003. 5

[138] Hauke Strasdat, J M MartÍnez Montiel, and Andrew J Davison. Real-time monocular SLAM: Why filter? In International Conference on Robotics and Automation, pages 2657–2664. IEEE, 2010. 18

[139] Hauke Strasdat, JMM Montiel, and Andrew J Davison. Real-time monocular slam: Why filter? In Robotics and Automation (ICRA), 2010 IEEE International Conference on, pages 2657–2664. IEEE, 2010. 54, 92

[140] Peter F Sturm and Stephen J Maybank. On plane-based camera calibration: A general algorithm, singularities, applications. In Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on., volume 1, pages 432–437. IEEE, 1999. 53, 92

[141] Niko Sünderhauf and Peter Protzel. Towards a robust back-end for pose graph slam. In Robotics and Automation (ICRA), 2012 IEEE International Conference on, pages 1254–1261. IEEE, 2012. 6

[142] Gian Diego Tipaldi and Kai O Arras. Flirt-interest regions for 2d range data. In Robotics and Automation (ICRA), 2010 IEEE International Conference on, pages 3616–3622. IEEE, 2010. 6, 110

[143] Gian Diego Tipaldi, Daniel Meyer-Delius, and Wolfram Burgard. Lifelong localization in changing environments. The International Journal of Robotics Research, 32(14):1662–1678, 2013. 73

[144] Lorenzo Torresani, Aaron Hertzmann, and Chris Bregler. Nonrigid structure-from-motion: Estimating shape and motion with hierarchical priors. IEEE transactions on pattern analysis and machine intelligence, 30(5):878–892, 2008. 111

[145] B Triggs, P F McLauchlan, and R I Hartley. Bundle adjustment—a modern synthesis. Workshop on Vision, 1999. 15

[146] Aisha Walcott-Bryant, Michael Kaess, Hordur Johannsson, and John J Leonard. Dynamic pose graph SLAM: Long-term mapping in low dynamic environments. In Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on, pages 1871–1878. IEEE, 2012. 7, 72

[147] Chieh-Chih Wang, Charles Thorpe, Sebastian Thrun, Martial Hebert, and Hugh Durrant-Whyte. Simultaneous localization, mapping and moving object tracking. The International Journal of Robotics Research, 26(9):889–916, 2007. 7

[148] Wera Winterhalter, Freya Fleckenstein, Bastian Steder, Luciano Spinello, and Wolfram Burgard. Accurate indoor localization for rgb-d smartphones and tablets given 2d floor plans. In Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on, pages 3138–3143. IEEE, 2015. 111

[149] Ryan W Wolcott and Ryan M Eustice. Visual localization within lidar maps for automated urban driving. In 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 176–183. IEEE, 2014. 110

[150] Zhenfei Yang, Tianbo Liu, and Shaojie Shen. Self-Calibrating Multi-Camera Visual-Inertial Fusion for Autonomous MAVs . In Intelligent Robots and Systems, pages 259–277. Springer US, 2016. 7

[151] Qilong Zhang and Robert Pless. Extrinsic calibration of a camera and laser range finder (improves camera calibration). In Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on, volume 3, pages 2301–2306. IEEE, 2004. 53, 92

Appendix A

Lie Groups

The optimization methods presented in the previous sections are applicable for scalar fields which are defined on Euclidean vector spaces $\mathbb{R}^n$. When performing optimization, we calculate an incremental update $\delta \in \mathbb{R}^n$ which is added to the current estimate $\mathbf{x}^{(k)} \in \mathbb{R}^n$

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \delta$$

however when dealing with rotations, this vector addition breaks down, as we will see in Section A.1. The problem lies in modeling rotations as an Euclidean vector space, which they are clearly not a part of, since in general performing a rotation by ω and then a rotation by δ is not equivalent to performing a rotation by $\omega + \delta$. Thus rotations must be modeled as **Lie Groups**, as will be seen in the following sections.

A.1 Motivation

We will start with a small example of a robot moving in a plane, parameterized by a **2D pose** (x, y, θ). When we give it a small forward velocity v_x, we know that the location changes as

$$\dot{x} = v_x$$

The solution to this trivial differential equation is, with x_0 the initial x-position of the robot,

$$x_t = x_0 + v_x t$$

A similar story holds for translation in the y direction, and in fact for translations in general:

$$(x_t, y_t, \theta_t) = (x_0 + v_x t, y_0 + v_y t, \theta_0)$$

Similarly for rotation we have

$$(x_t, y_t, \theta_t) = (x_0, y_0, \theta_0 + \omega t)$$

where ω is angular velocity, measured in rad/s in counterclockwise direction.

However, if we combine translation and rotation, the story breaks down! We cannot write

$$(x_t, y_t, \theta_t) = (x_0 + v_x t, y_0 + v_y t, \theta_0 + \omega t)$$

The reason is that, if we move the robot a tiny bit according to the velocity vector (v_x, v_y, ω), we have (to first order)

$$(x_\delta, y_\delta, \theta_\delta) = (x_0 + v_x \delta, y_0 + v_y \delta, \theta_0 + \omega \delta)$$

but now the robot has rotated, and for the next incremental change, the velocity vector would have to be rotated before it can be applied. In fact, the robot will move on a **circular** trajectory as illustrated in Figure A.1.

The reason is that **translation and rotation do not commute**: if we rotate and then move we will end up in a different place than if we moved first, then rotated.

To make progress, we have to be more precise about how the robot behaves. Specifically, let us define composition of two poses T_1 and T_2 as

$$T_1 T_2 = (x_1, y_1, \theta_1)(x_2, y_2, \theta_2) = (x_1 + \cos\theta_1 x_2 - \sin\theta y_2, y_1 + \sin\theta_1 x_2 + \cos\theta_1 y_2, \theta_1 + \theta_2)$$

This is a bit clumsy, so we resort to a trick: embed the 2D poses in the space of 3×3 matrices, so we can define composition as matrix multiplication:

$$T_1 T_2 = \begin{bmatrix} R_1 & t_1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R_2 & t_2 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} R_1 R_2 & R_1 t_2 + t_1 \\ 0 & 1 \end{bmatrix}$$

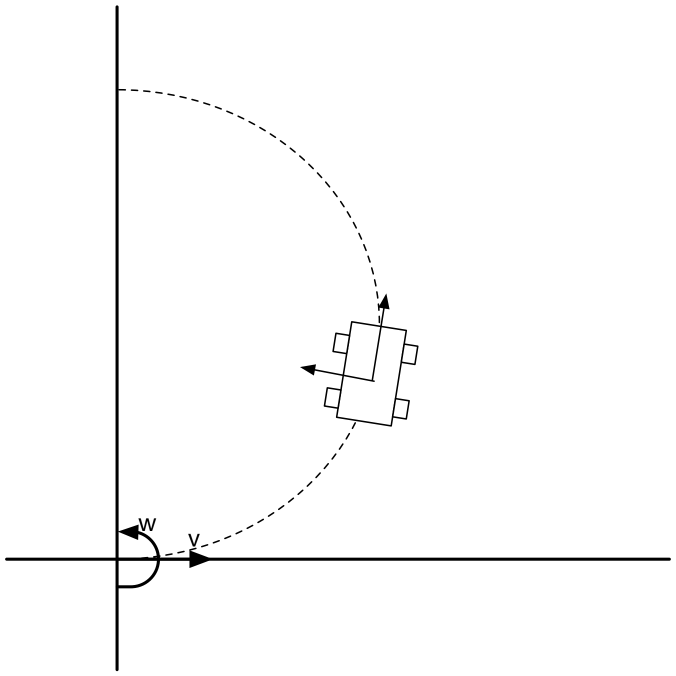


Figure A.1: A sketch of a robot moving in a circular trajectory.

where the matrices R are 2D rotation matrices defined as

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

Now a "tiny" motion of the robot can be written as

$$T(\delta) = \begin{bmatrix} \cos\omega\delta & -\sin\omega\delta & v_x\delta \\ \sin\omega\delta & \cos\omega\delta & v_y\delta \\ 0 & 0 & 1 \end{bmatrix} \approx \begin{bmatrix} 1 & -\omega\delta & v_x\delta \\ \omega\delta & 1 & v_y\delta \\ 0 & 0 & 1 \end{bmatrix} = I + \delta \begin{bmatrix} 0 & -\omega & v_x \\ \omega & 0 & v_y \\ 0 & 0 & 0 \end{bmatrix}$$

Let us define the **2D twist** vector $\xi = (v, \omega)$, and the matrix above as

$$\hat{\xi} \triangleq \begin{bmatrix} 0 & -\omega & v_x \\ \omega & 0 & v_y \\ 0 & 0 & 0 \end{bmatrix}$$

If we wanted t to be large, we could split up t into smaller timesteps, say n of them, and compose them as follows, as shown in Figure A.2:

$$T(t) \approx \left(I + \frac{t}{n}\hat{\xi}\right) \ldots \text{n times} \ldots \left(I + \frac{t}{n}\hat{\xi}\right) = \left(I + \frac{t}{n}\hat{\xi}\right)^n$$

Of course, the perfect solution would be obtained if we take n to infinity:

$$T(t) = \lim_{n\to\infty}\left(I + \frac{t}{n}\hat{\xi}\right)^n$$

For real numbers, this series is familiar and is actually a way to compute the exponential function:

$$e^x = \lim_{n\to\infty}\left(1 + \frac{x}{n}\right)^n = \sum_{k=0}^{\infty}\frac{x^k}{k!}$$

The series can be similarly defined for square matrices, and the final result is that we can write the motion of a robot along a circular trajectory, resulting from the 2D twist $\xi = (v, \omega)$ as the **matrix exponential** of $\hat{\xi}$:

$$T(t) = e^{t\hat{\xi}} \triangleq \lim_{n\to\infty}\left(I + \frac{t}{n}\hat{\xi}\right)^n = \sum_{k=0}^{\infty}\frac{t^k}{k!}\hat{\xi}^k$$

We call this mapping from 2D twists matrices $\hat{\xi}$ to 2D rigid transformations the **exponential map.**

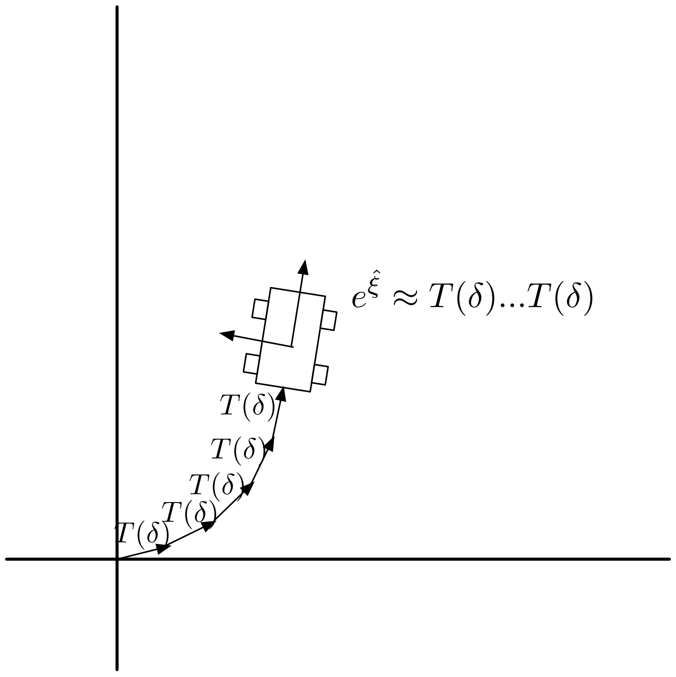


Figure A.2: A robot moving in a circular trajectory, approximated as n infinitesimal steps.

The above has all elements of Lie group theory. We call the space of 2D rigid transformations, along with the composition operation, the **special Euclidean group** $SE(2)$. It is called a Lie group because it is simultaneously a topological group and a manifold, which implies that the multiplication and the inversion operations are smooth. The space of 2D twists, together with a special binary operation to be defined below, is called the Lie algebra $\mathfrak{se}(2)$ associated with $SE(2)$.

A.2 Introduction to Lie Groups

We now define the concepts illustrated above, introduce some notation, and see what we can say in general.

A.2.1 A Manifold and a Group

A **Lie group** G is both a group **and** a manifold that possesses a smooth group operation. Associated with it is a **Lie Algebra** $\mathfrak{g}$ which, loosely speaking, can be identified with the tangent space at the identity and completely defines how the groups behaves around the identity. There is a mapping from $\mathfrak{g}$ back to G, called the **exponential map**

$$\exp : \mathfrak{g} \to G$$

which is typically a many-to-one mapping. The corresponding inverse can be define locally around the origin and hence is a "logarithm"

$$\log : G \to \mathfrak{g}$$

that maps elements in a neighborhood of id in G to an element in $\mathfrak{g}$.

An important family of Lie groups are the matrix Lie groups, whose elements are $n \times n$ invertible matrices. The set of all such matrices, together with the matrix multiplication, is called the general linear group $GL(n)$ of dimension n, and any closed subgroup of it is a **matrix Lie group**. Most if not all Lie groups we are interested in will be matrix Lie groups.

A.2.2 Lie Algebra

The Lie Algebra $\mathfrak{g}$ is called an algebra because it is endowed with a binary operation, the **Lie bracket** $[X, Y]$, the properties of which are closely related to the group operation of G. For example, for algebras associated with matrix Lie groups, the Lie bracket is given by $[A, B] \triangleq AB - BA$.

The relationship of the Lie bracket to the group operation is as follows: for commutative Lie groups vector addition $X + Y$ in $\mathfrak{g}$ mimics the group operation. For example, if we have $Z = X + Y$ in $\mathfrak{g}$, when mapped backed to G via the exponential map we obtain

$$e^Z = e^{X+Y} = e^X e^Y$$

However, this does **not** hold for non-commutative Lie groups:

$$Z = \log(e^X e^Y) \neq X + Y$$

Instead, Z can be calculated using the Baker-Campbell-Hausdorff (BCH) formula:

$$Z = X + Y + [X, Y]/2 + [X - Y, [X, Y]]/12 - [Y, [X, [X, Y]]]/24 + \dots$$

For commutative groups the bracket is zero and we recover $Z = X + Y$. For non-commutative groups we can use the BCH formula to approximate it.

A.2.3 Exponential Coordinates

For n-dimensional matrix Lie groups, as a vector space the Lie algebra $\mathfrak{g}$ is isomorphic to $\mathbb{R}^n$, and we can define the hat operator [107, page 41],

$$\hat{} : x \in \mathbb{R}^n \to \hat{x} \in \mathfrak{g}$$

which maps n-vectors $x \in \mathbb{R}^n$ to elements of $\mathfrak{g}$. In the case of matrix Lie groups, the elements $\hat{x}$ of $\mathfrak{g}$ are also $n \times n$ matrices, and the map is given by

$$\hat{x} = \sum_{i=1}^{n} x_i G^i \tag{A.1}$$

where the G^i are $n \times n$ matrices known as Lie group generators. The meaning of the map $x \to \hat{x}$ will depend on the group G and will generally have an intuitive interpretation.

A.2.4 Actions

An important concept is that of a group element acting on an element of a manifold M. For example, 2D rotations act on 2D points, 3D rotations act on 3D points, etc. In particular, a **left action** of G on M is defined as a smooth map $\Phi : G \times M \to M$ such that [107, Appendix A]:

(1) The identity element e has no effect, i.e., $\Phi(e, p) = p$

(2) Composing two actions can be combined into one action: $\Phi(g, \Phi(h, p)) = \Phi(gh, p)$

The (usual) action of an n-dimensional matrix group G is matrix-vector multiplication on $\mathbb{R}^n$,

$$q = Ap$$

with $p, q \in \mathbb{R}^n$ and $A \in G \subseteq GL(n)$.

A.2.5 The Adjoint Map and Adjoint Representation

Suppose a point p is specified as p' in the frame T, i.e., $p' = Tp$, where T transforms from the global coordinates p to the local frame p'. To apply an action A we first need to undo T, then apply A, and then transform the result back to T:

$$q' = TAT^{-1}p'$$

The matrix TAT^{-1} is said to be conjugate to A, and this is a central element of group theory.

In general, the **adjoint map** $\mathbf{Ad}_g$ maps a group element $a \in G$ to its **conjugate** gag^{-1} by g. This map captures conjugacy in the group G, but there is an equivalent notion in the Lie algebra $\mathfrak{g}$,

$$\mathbf{Ad}_g e^{\hat{x}} = g \exp(\hat{x})\, g^{-1} = \exp(Ad_g \hat{x})$$

where $Ad_g : \mathfrak{g} \to \mathfrak{g}$ is a map parameterized by a group element g, and is called the **adjoint representation**. The intuitive explanation is that a change $\exp(\hat{x})$ defined around the origin, but applied at the group element g, can be written in one step by taking the adjoint $Ad_g \hat{x}$ of $\hat{x}$.

In the special case of matrix Lie groups the adjoint can be written as

$$Ad_T\hat{x} \triangleq T\hat{x}T^{-1}$$

and hence we have

$$Te^{\hat{x}}T^{-1} = e^{T\hat{x}T^{-1}} \tag{A.2}$$

where both $T \in G$ and $\hat{x} \in \mathfrak{g}$ are $n \times n$ matrices for an n-dimensional Lie group.

A.3 3D Rotations

A.3.1 Basics

The Lie group $SO(3)$ is a subgroup of the general linear group $GL(3)$ of 3×3 invertible matrices. Its Lie algebra $\mathfrak{so}(3)$ is the vector space of 3×3 skew-symmetric matrices $\hat{\omega}$. Since $SO(3)$ is a three-dimensional manifold, $\mathfrak{so}(3)$ is isomorphic to $\mathbb{R}^3$ and we define the map

$$\hat{} : \mathbb{R}^3 \to \mathfrak{so}(3)$$

$$\hat{} : \omega \to \hat{\omega} = [\omega]_\times$$

which maps 3-vectors ω to skew-symmetric matrices $[\omega]_\times$:

$$[\omega]_\times = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} = \omega_x G^x + \omega_y G^y + \omega_z G^z$$

Here the matrices G^i are the generators for $SO(3)$,

$$G^x = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix} G^y = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix} G^z = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

corresponding to a rotation around X, Y, and Z, respectively. The Lie bracket $[x, y]$ in $\mathfrak{so}(3)$ corresponds to the cross product $x \times y$ in $\mathbb{R}^3$.

Hence, for every 3-vector ω there is a corresponding rotation matrix

$$R = e^{[\omega]_\times}$$

which defines a canonical parameterization of $SO(3)$, with ω known as the canonical or exponential coordinates. It is equivalent to the axis-angle representation for rotations, where the unit vector ω/θ defines the rotation axis, and its magnitude the amount of rotation θ.

The exponential map can be computed in closed form using **Rodrigues' formula** [107, page 28]:

$$e^{\hat{\omega}} = I + \frac{\sin\theta}{\theta}\hat{\omega} + \frac{1-\cos\theta}{\theta^2}\hat{\omega}^2 \tag{A.3}$$

where $\hat{\omega}^2 = \omega\omega^T - I$, with $\omega\omega^T$ the outer product of ω. Hence, a slightly more efficient variant is

$$e^{\hat{\omega}} = (\cos\theta)\, I + \frac{\sin\theta}{\theta}\hat{\omega} + \frac{1-cos\theta}{\theta^2}\omega\omega^T \tag{A.4}$$

A.3.2 Diagonalized Form

Because a 3D rotation R leaves the axis ω unchanged, R can be diagonalized as

$$R = C \begin{pmatrix} e^{-i\theta} & 0 & 0 \\ 0 & e^{i\theta} & 0 \\ 0 & 0 & 1 \end{pmatrix} C^{-1}$$

with $C = \begin{pmatrix} c_1 & c_2 & \omega/\theta \end{pmatrix}$, where c_1 and c_2 are the complex eigenvectors corresponding to the 2D rotation around ω. This also means that, by (A.2),

$$\hat{\omega} = C \begin{pmatrix} -i\theta & 0 & 0 \\ 0 & i\theta & 0 \\ 0 & 0 & 0 \end{pmatrix} C^{-1}$$

In this case, C has complex columns, but we also have

$$\hat{\omega} = B \begin{pmatrix} 0 & -\theta & 0 \\ \theta & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} B^T \tag{A.5}$$

with $B = \begin{pmatrix} b_1 & b_2 & \omega/\theta \end{pmatrix}$, where b_1 and b_2 form a basis for the 2D plane through the origin and perpendicular to ω. Clearly, from Section **??**, we have

$$c_1 = B \begin{pmatrix} 1 \\ i \\ 0 \end{pmatrix} \quad \text{and} \quad c_2 = B \begin{pmatrix} i \\ 1 \\ 0 \end{pmatrix}$$

and when we exponentiate (A.5) we expose the 2D rotation around the axis ω/θ with magnitude θ:

$$R = B \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} B^T$$

The latter form for R can be used to prove Rodrigues' formula. Expanding the above, we get

$$R = (\cos\theta)\left(b_1 b_1^T + b_2 b_2^T\right) + (\sin\theta)\left(b_2 b_1^T - b_1 b_2^T\right) + \omega\omega^T/\theta^2$$

Because B is a rotation matrix, we have $BB^T = b_1 b_1^T + b_2 b_2^T + \omega\omega^T/\theta^2 = I$, and using (A.5) it is easy to show that $b_2 b_1^T - b_1 b_2^T = \hat{\omega}/\theta$, hence

$$R = (\cos\theta)\,(I - \omega\omega^T/\theta^2) + (\sin\theta)\,(\hat{\omega}/\theta) + \omega\omega^T/\theta^2$$

which is equivalent to (A.4).

A.3.3 The Adjoint Map

For rotation matrices R we can prove the following identity:

$$R[\omega]_\times R^T = [R\omega]_\times \tag{A.6}$$

Hence, given property (A.6), the adjoint map for $\mathfrak{so}(3)$ simplifies to

$$Ad_R[\omega]_\times = R[\omega]_\times R^T = [R\omega]_\times$$

and this can be expressed in exponential coordinates simply by rotating the axis ω to $R\omega$.

As an example, to apply an axis-angle rotation ω to a point p in the frame R, we could:

(1) First transform p back to the world frame, apply ω, and then rotate back:

$$q = Re^{[\omega]_\times} R^T$$

(2) Immediately apply the transformed axis-angle transformation $Ad_R[\omega]_\times = [R\omega]_\times$:

$$q = e^{[R\omega]_\times} p$$

A.3.4 Actions

In the case of $SO(3)$ the vector space is $\mathbb{R}^3$, and the group action corresponds to rotating a point

$$q = Rp$$

We would now like to know what an incremental rotation parameterized by ω would do:

$$q(\omega) = Re^{[\omega]_\times} p$$

hence the derivative is:

$$\frac{\partial q(\omega)}{\partial \omega} = R\frac{\partial}{\partial \omega}\left(e^{[\omega]_\times} p\right) = R\frac{\partial}{\partial \omega}\left([\omega]_\times p\right) = R[-p]_\times$$

To show the last equality note that

$$[\omega]_\times p = \omega \times p = -p \times \omega = [-p]_\times \omega$$