UNIVERSITY OF CALIFORNIA
RIVERSIDE

Visual-Inertial Odometry on Resource-Constrained Systems

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Electrical Engineering

by

Mingyang Li

December 2014

The Dissertation of Mingyang Li is approved:

Committee Chairperson

University of California, Riverside

Acknowledgements

First of all, I would like to express my deepest appreciation to my advisor, Prof. Anastasios Mourikis, for his constant guidance, support, and encouragement during the past five years. He continually instilled the requirements of being a good engineer and scientist into my mind, which will have a life-long influence on me.

Secondly, I would like to thank Prof. Jay Farrell, Prof. Wei Ren, Prof. Hamed Mohsenian-Rad, Prof. Christian Shelton, and Prof. Amit K. Roy-Chowdhury, for their valuable help and thoughtful suggestion for my graduate study and research. I would like to also express my gratitude to Prof. Ertem Tuncel, Prof. Jie Chen, Prof. Yingbo Hua, and Prof. Matthew Barth, for their efforts and insights in teaching the graduate courses, from which I have learned a lot.

In addition, my thanks go to my fellow graduate students at the robotics and computer vision group, Anning Chen, Arvind Ramanandan, Akshay More, Dongfang Zheng, Haiyu Zhang, Hongsheng Yu, Tue-Cuong Dong-Si, Sheng Zhao, Suvarna Sarath, Xing Zheng, and Yiming Chen, with all of whom I shared hours of discussion, study, work, play, food, and travel. Also I would like to thank my girlfriend Tianran Jia, and my friends at Riverside, Shu Zhang, Yulin Zhang, Songfan Yang, Qichi Yang, Xiaoyin Ma, Zheng Qin, for making my graduate life enriched and memorable.

To my parents for all the support.

ABSTRACT OF THE DISSERTATION

Visual-Inertial Odometry on Resource-Constrained Systems

by

Mingyang Li

Doctor of Philosophy, Graduate Program in Electrical Engineering
University of California, Riverside, December 2014
Dr. Anastasios Mourikis, Chairperson

In this work, we focus on the problem of pose estimation in unknown environments, using the measurements from an inertial measurement unit (IMU) and a single camera. We term this estimation task visual-inertial odometry (VIO), in analogy to the well-known visual-odometry (VO) problem. Our focus is on developing VIO algorithms for high-precision, consistent motion estimation in real time. The majority of VIO algorithms proposed to date have been developed for systems which are equipped with high-end processors and high-quality sensors. By contrast, we are interested in tracking the motion of systems that are small and inexpensive, and are equipped with limited processing and sensing resources. Such *resource-constrained* systems are common in application areas such as micro aerial vehicles, mobile phones, and augmented reality (AR) devices. Endowing such systems with the capability to accurately track their poses will create a host of new opportunities for commercial applications, and lower the barrier to entry in robotics research and development.

Performing accurate motion estimation on resource-constrained systems requires novel methodologies to address the challenges caused by the limited sensing and processing capabilities, and to provide guarantees for the optimal utilization of these resources. To this end, in this work, we focus on developing novel, resource-adaptive VIO algorithms based on the extended Kalman filter (EKF) formulation. Specifically, we (i) analyze the properties and performance of existing EKF-based VIO approaches, and propose a novel estimator design method, which ensures the correct observability properties of the linearized system models to improve the estimates' accuracy and consistency, (ii) present a methodology for minimizing the computational cost of the EKF-VIO algorithms, which relies on online optimization of the estimator's parameters based on the properties of the environment, (iii) propose an algorithm for joint online calibration of the spatial and temporal relationship between the visual and inertial sensors, and (iv) propose high-fidelity sensor models that enable us to process the measurements captured by rolling-shutter cameras and low-cost inertial sensors. We evaluate our estimators with various simulated and real-world data sets, which demonstrate that our proposed formulations are able to consistently and accurately track the pose of resource-constrained systems in real time.

Contents

List of Figures

List of Tables

Nomenclature

$\hat{\mathbf{x}}$	An estimate of the variable $\mathbf{x}$.
$\tilde{\mathbf{x}}$	Errors in the estimate $\hat{\mathbf{x}}$.
$\hat{\mathbf{x}}_{i\|j}$	The estimate of the variable $\mathbf{x}$ at the timestep i given sensor measurements up to the timestep j.
${}^{A}_{B}\mathbf{R}$	The rotation matrix rotating vectors from the coordinate frame {B} to the coordinate frame {A}.
${}^{A}_{B}\bar{\mathbf{q}}$	The unit quaternion vector corresponding to the rotation matrix ${}^{A}_{B}\mathbf{R}$.
${}^{A}\mathbf{c}$	The vector $\mathbf{c}$ expressed in the coordinate frame {A}.
${}^{A}\mathbf{p}_{B}$	Position of the origin of frame {B} expressed in {A}.
$\mathbf{I}_n$	The $n \times n$ identity matrix.
$\mathbf{0}_n$	The $n \times n$ zero matrix.
$\mathbf{0}_{m\times n}$	The $m \times n$ zero matrix.
$\otimes$	The quaternion multiplication operator.
$\lfloor \mathbf{a}\times \rfloor$	The skew-symmetric matrix corresponding to the vector $\mathbf{a}$.

Chapter 1

Introduction

1.1 Localization Using Visual and Inertial Sensors

Localization is the task of determining the pose (position and orientation) of a mobile platform, using sensor measurements. Typically, a localization system includes multiple sensors that are used to capture information about the system's motion and/or its surroundings, and a computer used to process the sensor information to produce estimates about the system's pose. In recent years, localization algorithms have been at the focus of intense research interest, due to their importance for applications such as space exploration [1, 84, 112], self-driving cars [109, 115], unmanned aerial vehicles [14, 60, 79], and several others.

For applications where GPS signals can be reliably received, a number of proven localization solutions exist [25, 106, 126]. However, when GPS is unavailable, the localization problem is significantly more challenging. While successful

GPS-denied navigation systems have been developed for a number of applications [54, 55, 80, 100], a common characteristic of all these systems is that they rely on expensive, high-quality sensors, and often require significant processing power. Such reliance on expensive hardware makes the widespread adoption of these localization solutions difficult. Specifically, in many applications there exist strict limitations in terms of size, weight, power consumption, battery life, and cost, which preclude the use of high-end processors and high-quality sensors. Such *resource-constrained systems* include mobile phones, tablet computers, augmented reality (AR) devices, micro aerial vehicles (MAVs), and several others, which are quickly becoming ubiquitous in both industrial applications and our daily lives. Endowing these resource-constrained systems with the capability to accurately localize will create a host of new opportunities in commercial applications and also lower the barrier to entry in robotics research and development.

One key decision in designing a system for GPS-denied localization is the selection of appropriate sensors. These sensors can be categorized into two main types: *proprioceptive* and *exteroceptive.* Proprioceptive sensors provide measurements of quantities related to the motion of the system, with typical examples being wheel encoders [15, 122], which measure linear and rotational velocity, and inertial measurement units (IMUs) [8, 25], which measure rotational velocity and specific force. On the other hand, exteroceptive sensors capture information about the surroundings, with common examples being laser scanners [4], radars [51], magnetometers [99], cameras [67, 84, 120], and sonars [27].

In this work, we focus on localization using an IMU and a single camera, as these sensors offer several advantages for localizing small, inexpensive systems. First of all, both IMUs and cameras provide 3D motion information, which is a prerequisite for the application domains of interest. In addition, both IMUs and cameras can operate without requiring any external infrastructure, and thus the resulting localization system can operate in any environment. Moreover, since the IMU is a proprioceptive sensor and the camera is an exteroceptive one, they provide complementary measurement information. As a result, by fusing their sensory input, we can obtain increased accuracy and robustness of the localization system. More importantly, both sensors are suitable for resource-constrained systems, since they consume little power, and are small, light-weight, and inexpensive. Finally, both IMUs and cameras are already found in several commercial resource-constrained devices (e.g, mobile phones and AR devices), and thus using them for providing location information does not require installation of additional hardware.

Methods that use cameras and IMUs for localization are often termed *vision-aided inertial navigation* methods. In these, the cameras are used to detect and track distinctive visual features, which may include points [66], lines [58], planes [33], or objects [95]. Subsequently, the measurements of these features are fused with the IMU measurements for ego-motion estimation. Depending on the type of feature information available, and the system-design objectives, the vision-aided inertial navigation approaches can be grouped into three board categories: map-based localization, simultaneous localization and mapping (SLAM), and visual-inertial odometry (VIO). In the following, we briefly discuss each type of methods.

- Map-based localization: These methods assume the existence of an *a priori* feature map of the area where the localization system operates [9, 112, 121]. Using the observations of features with known coordinates, we can obtain estimates of the system pose with bounded uncertainty. This increases the system's robustness to failure and simplifies estimator design. However, accurate prior feature maps are not known in advance in many situations, and therefore the applicability of this type of approaches is limited.

- Simultaneous localization and mapping (SLAM): In SLAM methods, the sensor measurements are used to localize a system in an unknown environment, while concurrently building a map for the environment [43,50,91,105]. However, constructing and maintaining a detailed feature map is computationally costly, and thus SLAM algorithms should be only used in situations where the feature map is necessary and useful. If the goal of a vision-aided inertial navigation algorithm is to track a moving platform only, building and maintaining a map should be avoided.

- Visual-inertial odometry (VIO): The goal of VIO algorithms is to estimate the pose of a moving platform as it moves in an unknown environment, without the overhead of building a feature map [57, 66, 80, 119]. By focusing only on pose estimation, VIO algorithms attain low computational cost. This makes them suitable for systems with limited resources, which is our primary interest. These algorithms can be used either as standalone localization methods [66, 80, 120], or as parts of larger localization systems [87, 96, 97].

In this work, since our goal is to track the motion of resource-constrained systems in any environment, we focus on the problem of visual-inertial odometry.

1.2 Key Contributions

The majority of VIO algorithms developed to date have been tailored for specialized robotic systems, which are equipped with high-end sensors and processors. In these systems, the availability of high-quality sensor data and of processing resources simplifies the algorithm design process, to some extent. By contrast, in the resource-constrained systems we are interested in, the design of a localization algorithm is significantly more challenging. First, the limited CPU capabilities of the systems require VIO algorithms to *optimally* utilize the available computational resources, to attain the highest possible accuracy. Second, the lower quality of inexpensive sensors introduces additional technical challenges that need to be addressed. Specifically, the camera-to-IMU spatial and temporal configuration is typically *not* accurately known at the hardware level, which introduces extra sensor-calibration problems. Furthermore, low-cost sensors are typically affected by additional systematic errors such as IMU axis misalignment, scale factors, and g-sensitivity, and the image distortions caused by rolling-shutter cameras. These factors, which are typically not present in high-end sensor systems, need to be modelled and treated properly to achieve high-precision pose estimation. In this work, we propose algorithms for addressing each of the challenges mentioned above, as described in more detail in what follows.

1.2.1 Algorithm design for high-precision, consistent VIO

The first contribution of this work is the design of an EKF-based, high-precision, consistent VIO algorithm. We note that, to date, numerous algorithms have been proposed for VIO, with the majority relying on either the extended Kalman filter [43, 50, 119], or nonlinear iterative least-squares [24, 71, 100]. In this work, the main goal is to design VIO algorithms suitable for resource-constrained systems, where computational capabilities are limited. To this end, we focus on EKF-based VIO, since iterative least-squares algorithms require multiple re-linearizations of the nonlinear measurement models and thus lead to higher computational cost. In our work, we first perform a detailed study of existing EKF-based VIO algorithms, by comparing both their theoretical properties and empirical performance. We show that an EKF formulation where the state vector comprises a sliding window of poses (the MSCKF algorithm [80]) attains better accuracy, consistency, and computational efficiency than the SLAM formulation of the EKF, in which the state vector contains the current pose and the features seen by the camera. Moreover, we prove that both types of EKF approaches are *inconsistent*, due to the way in which Jacobians are computed. Specifically, we show that the observability properties of the EKF's linearized system models *do not* match those of the underlying system, which causes the filters to underestimate the uncertainty in the state estimates. Based on our analysis, we propose a novel, real-time EKF-based

VIO algorithm, termed MSCKF 2.0, which reports consistent motion estimates by ensuring the correct observability properties of its linearized system model. We show that the proposed MSCKF 2.0 algorithm is able to obtain substantially better estimation accuracy and consistency, compared to both existing EKF-based VIO approaches *and* an iterative non-linear least-squares algorithm.

1.2.2 Optimal utilization of the computational resources

The second key contribution of this work is an EKF-based VIO algorithm that processes the available feature measurements optimally, in terms of utilization of the computational resources, and thus enables real-time operation on the low-end processors of resource-constrained systems. Specifically, we propose a MSCKF/SLAM hybrid filter that combines the MSCKF 2.0 algorithm and EKF-based SLAM in a way that leverages their complementary computational properties, while maintaining high estimation accuracy. A key characteristic of the proposed hybrid filter is that it learns, at runtime, the feature tracks' statistical properties that affect its computational cost, and employs this information in order to minimize this cost. We show that this optimization can be formulated as a one-variable minimization problem, which can be solved efficiently at runtime. This process allows the estimator to optimally decide whether each of the available features should be processed by the MSCKF 2.0 or the EKF-SLAM approach, in order to minimize the computation. Compared to each individual algorithm, the computational cost of the hybrid filter is significantly reduced. Additionally, since the optimization process is performed by learning the feature information, the estimator is able to adapt to the changes of the environment. We show that the proposed hybrid filter can attain high-precision state estimates, while operating in real-time on low-end processors, similar to those found on a mobile phone.

1.2.3 Online camera-to-IMU calibration

The third key contribution of this work is an estimation approach that calibrates the relative spatial and temporal relationship between the visual and inertial sensors. The relative spatial relationship refers to the relative orientation and translation between the sensors' coordinate frames. On the other hand, the temporal relationship refers to the relative time offset between the sensors' measurement time stamps. For performing accurate VIO, both the spatial and temporal calibration parameters must be precisely estimated. To this end, we include these parameters into the state vector of the proposed MSCKF/SLAM hybrid filter, and estimate them along with all other variables in the state (IMU pose, velocity, biases, and feature positions). Moreover, we present a detailed identifiability analysis of these parameters. Specifically, since the identifiability properties of the spatial calibration parameters have been analyzed in prior work [43, 50], we here focus on the properties of the time offset. We show that the time offset is locally identifiable, except in a small number of degenerate motion cases, which we characterize in detail. These degenerate cases are either (i) cases known to cause loss of observability even when no time offset exists, or (ii) cases that are unlikely to occur in practice. Our estimation approach as well as the identifiability analysis are validated by both simulation and experimental results, which demonstrate that the proposed approach yields high-precision, consistent estimates, with both constant and time-varying offsets.

1.2.4 Localization using a low-cost inertial sensor and a rolling-shutter camera

The final key contribution of this work is a high-precision pose estimation algorithm which addresses the special characteristics of low-cost MEMS inertial sensors and rolling-shutter cameras. On the one hand, low-cost inertial sensors exhibit significant systematic errors, which must be estimated and compensated for to allow accurate pose estimation. On the other hand, the rolling-shutter sensors which are found on most low-cost cameras create significant distortions (the rolling shutter effect) which, if left unmodeled, can lead to poor performance. To address these issues, we propose an algorithm that employs high-fidelity sensor models for both the IMU and the rolling-shutter camera, and performs online estimation of *all* the model parameters. Moreover, we propose a novel method for processing the measurements of rolling-shutter cameras, which employs an approximate representation of the estimation *errors*, instead of the state itself. We show that the proposed error representation results in *lower* modeling errors compared to existing approaches, and thus leads to better performance. The high-fidelity sensor modeling, online calibration, and novel rolling-shutter formulation are employed in conjunction with the MSCKF/SLAM hybrid filter. The resulting algorithm is shown to result in high-precision localization using the limited computational resources and low-cost sensors of a mobile phone.

1.3 Manuscript Organization

The remainder of the dissertation is organized as follows. Chapter 2 analyzes existing EKF-based VIO approaches and presents an accurate and consistent VIO algorithm, termed MSCKF 2.0. In Chapter 3, we present a hybrid EKF which further optimizes the computational cost of the MSCKF 2.0. Chapter 4 addresses the spatial and temporal calibration problem for fusing the visual and inertial measurements. In Chapter 5, we propose our approach for performing VIO using the measurements from low-cost sensors. Finally, Chapter 6 concludes the dissertation.

Chapter 2

Consistent EKF-based Visual-Inertial Odometry

2.1 Introduction

To date, most VIO algorithms are either extended Kalman filter (EKF)-based methods [43, 50, 80], or methods utilizing iterative minimization over a window of states [21, 55, 56]. The latter are generally considered to be more accurate, as they employ re-linearization at each iteration to better deal with the nonlinearity of the measurement models. However, the need for multiple iterations also incurs a high computational cost, compared to EKF-based methods. Ideally, we would like to have an algorithm that obtains accuracy similar to, or better than, that of iterative-minimization algorithms, but at the computational cost of an EKF algorithm. In this chapter, we show that this can be achieved. Specifically, we present a novel EKF-based VIO algorithm, which attains estimation accuracy better than both existing EKF alternatives *and* an iterative-minimization-based VIO method.

We begin by comparing the performance of two families of EKF-based VIO estimators: EKF-SLAM and sliding-window algorithms. In the former class of methods, the filter state vector contains the current IMU pose as well as the visual features [43, 54, 91]. Since we are interested in performing VIO, not mapping, only the *currently visible* features are maintained in the state vector, to keep the computational cost bounded [85]. On the other hand, in the latter class of methods, the state vector contains only a sliding window of poses, and the feature measurements are used to apply probabilistic constraints between them [20, 80]. Out of this second class of methods, we focus on the multi-state-constraint Kalman filter (MSCKF) algorithm [80], which has been shown to be the maximum-a-posteriori estimator up to linearization [67]. We here show that the MSCKF algorithm outperforms EKF-SLAM in terms of both accuracy and consistency in experimental data, and provide a theoretical explanation for this result.

Having shown the advantages of the MSCKF over EKF-SLAM methods, we then focus on analyzing and further improving its performance. Specifically, our approach relies on improving the *consistency* of the MSCKF, which, in turn, also improves the accuracy of the estimates. As defined in [7, Section 5.4], a recursive estimator is consistent when the estimation errors are zero-mean and have covariance matrix equal to that reported by the estimator. In this chapter, we identify and address the root cause of inconsistency in the MSCKF, which is related to a fundamental shortcoming of the EKF: we prove that, due to the way the EKF Jacobians are computed, even though the IMU's rotation about gravity (the yaw) is *not* observable in VIO (see, e.g., [43, 50, 75]), it *appears to be* observable in the linearized system model used by the MSCKF – and the same occurs in EKF-SLAM. Thus, the estimator erroneously believes it has more information than it actually does, and reports a covariance matrix for the state that underestimates the actual one.

To address the inconsistency problem, we present an approach that ensures the appropriate observability properties for the filter's linearized system model. To achieve this, we derive a closed-form expression for the IMU's error-state transition matrix, and employ it in conjunction with a modified way to compute the filter Jacobians. We term the resulting algorithm MSCKF 2.0. Our results show that the MSCKF 2.0 obtains higher consistency and accuracy compared to the original algorithm [80], but also compared to a sliding-window iterative-minimization approach, which has much higher computational cost.

2.2 Related Work

The simplest (and most computationally efficient) approaches to VIO are *loosely-coupled* ones, i.e, methods that process the IMU and image measurements separately. For instance, some methods first process the images for computing relative-motion estimates between consecutive poses, and subsequently fuse these with the inertial measurements [20, 73, 94, 107, 120]. Alternatively, IMU measurements can be processed separately for extracting rotation estimates, and fused in an image-based estimation algorithm [14, 56, 88]. Separately processing the two sources of information leads to a reduction in computational cost. This however, comes at the expense of information loss: for instance, using feature measurements for egomotion estimation between pairs of images ignores the correlations between consecutive timesteps [82], and processing IMU measurements separately does not allow for optimal estimation of sensor biases.

In this work, we are therefore interested in *tightly-coupled* methods, which directly fuse the visual and inertial data, thus achieving higher precision. As previously mentioned, these are either based on iterative minimization over a sliding window of states, or are EKF formulations[1]. In the former methods (e.g., [21, 55, 56, 72, 88]), which essentially implement bundle-adjustment in a sliding window of states [23] with the addition of IMU measurements, older poses and/or features are removed from the state vector to maintain the computational cost bounded. The need for multiple iterations during minimization results in increased computational cost, however. In this chapter, we show that a properly designed EKF estimator can attain performance *higher* than that of iterative minimization, at only a fraction of the computation.

To fuse the visual and inertial measurements, the most commonly used tightly-coupled EKF estimator is EKF-based SLAM, in which the current camera pose and feature positions are jointly estimated [43, 50, 54, 91, 92]. To keep the computational cost bounded in EKF-SLAM algorithms, features that move out of the

[1]Note that hybrid approaches have also appeared (e.g., [81]), which use the estimates of the EKF as initial guesses for iterative minimization. Moreover, hybrid schemes that maintain both an EKF and a minimization-based estimator for robustness and/or efficiency have been proposed [14, 119].

camera's field of view must be removed from the state vector [85]. On the other hand, the MSCKF algorithm is an alternative EKF-based VIO method [80, 98], which maintains a sliding window of poses in the state vector and uses the feature measurements to impose constraints on these poses. In this chapter, we compare the MSCKF and EKF-SLAM algorithms, and show that the MSCKF can lead to estimates with better accuracy and consistency.

A key contribution of this chapter is a systematic approach to improve the consistency of EKF-based vision-aided inertial navigation. Specifically, our approach is motivated by recent work examining the relationship between the observability properties of the EKF's linearized system model and the filter's consistency, in the context of 2D SLAM [40, 41]. These works showed that, due to the way in which Jacobians are computed in the EKF, the robot's orientation appears to be observable in the linearized system model, while it is not in the actual, nonlinear system. As a result of this mismatch, the filter produces too small estimates for the uncertainty of the orientation estimates, and becomes inconsistent. In this chapter, we show that the same problem affects EKF-based VIO in 3D, and present approaches to address it and improve the estimation consistency and accuracy.

2.3 EKF-based Visual-Inertial Odometry

Consider a mobile platform, equipped with an IMU and a camera, moving with respect to a global coordinate frame, $\{G\}$. Our goal is to perform VIO, i.e., to track the position and orientation of the platform using inertial measurements and observations of naturally occurring point features, whose positions are not known *a priori*. To this end, we affix a coordinate frame $\{I\}$ to the IMU, and track the motion of this frame with respect to the global frame. We note that, to define the frame $\{I\}$, we choose its origin to be the point where the three accelerometer axes intersect, and select the direction of its axes along that of the accelerometer axes. In this chapter, it is assumed that the IMU axes are perfectly orthogonal to each other. The case where the axes are mis-aligned is treated in Chapter 5. In what follows, we first describe the parameterization we employ for the IMU state, and then discuss the two alternative tightly-coupled EKF formulations for VIO, and compare their performance.

2.3.1 IMU state parameterization

Following standard practice, the IMU state at time step k is defined as the 16×1 vector

$$\mathbf{x}_{E_k} = \begin{bmatrix} {}^{I_k}_{G}\bar{\mathbf{q}}^T & {}^{G}\mathbf{p}_{I_k}^T & {}^{G}\mathbf{v}_{I_k}^T & \mathbf{b}_{\mathbf{g}k}^T & \mathbf{b}_{\mathbf{a}k}^T \end{bmatrix}^T \tag{2.1}$$

where ${}^{I_k}_{G}\bar{\mathbf{q}}$ is the unit quaternion [113] representing the rotation from the global frame to the IMU frame at time step k, ${}^{G}\mathbf{p}_{I_k}$ and ${}^{G}\mathbf{v}_{I_k}$ are the IMU position and velocity in the global frame, and $\mathbf{b}_{\mathbf{g}_k}$ and $\mathbf{b}_{\mathbf{a}_k}$ are the gyroscope and accelerometer biases. The bias vectors $\mathbf{b}_{\mathbf{g}}$ and $\mathbf{b}_{\mathbf{a}}$ are modeled as random walk processes, driven by zero-mean white Gaussian noises $\mathbf{n}_{\mathbf{wg}}$ and $\mathbf{n}_{\mathbf{wa}}$ respectively. To define the error state corresponding to $\mathbf{x}_E$, we use the standard additive error definition for the position, velocity, and biases $\left(\text{e.g., } {}^{G}\tilde{\mathbf{p}}_I = {}^{G}\mathbf{p}_I - {}^{G}\hat{\mathbf{p}}_I\right)$. For the orientation error, out of the several possible options that exist, the preferable ones are those that (i) are local, so that singularities are avoided, and (ii) use a minimal, 3-dimensional representation of the orientation error. To obtain a local parameterization, we define the orientation error based on the quaternion $\delta\bar{\mathbf{q}}$ that describes the difference between the true and estimated orientation. Specifically, we define:

$$ {}^{I}_{G}\bar{\mathbf{q}} = {}^{I}_{G}\hat{\bar{\mathbf{q}}} \otimes \delta\bar{\mathbf{q}} \Rightarrow \delta\bar{\mathbf{q}} = {}^{I}_{G}\hat{\bar{\mathbf{q}}}^{-1} \otimes {}^{I}_{G}\bar{\mathbf{q}} \tag{2.2} $$

where $\otimes$ denotes quaternion multiplication. Intuitively, $\delta\bar{\mathbf{q}}$ is the (small) rotation that should be applied so that the estimated global frame matches the true one. To obtain a minimal representation for this rotation, we note that $\delta\bar{\mathbf{q}}$ can be written as:

$$ \delta\bar{\mathbf{q}} = \begin{bmatrix} \frac{1}{2}{}^{G}\tilde{\boldsymbol{\theta}}_I \\ \sqrt{1 - \frac{1}{4}{}^{G}\tilde{\boldsymbol{\theta}}_I^{T}\,{}^{G}\tilde{\boldsymbol{\theta}}_I} \end{bmatrix} \simeq \begin{bmatrix} \frac{1}{2}{}^{G}\tilde{\boldsymbol{\theta}}_I \\ 1 \end{bmatrix} \tag{2.3} $$

where ${}^{G}\tilde{\boldsymbol{\theta}}_I$ is a 3×1 vector describing the orientation errors about the three axes. With the above error definition, the evolving IMU error-state at time-step k is defined as the 15×1 vector:

$$ \tilde{\mathbf{x}}_{E_k} = \begin{bmatrix} {}^{G}\tilde{\boldsymbol{\theta}}_{I_k}^{T} & {}^{G}\tilde{\mathbf{p}}_{I_k}^{T} & {}^{G}\tilde{\mathbf{v}}_{I_k}^{T} & \mathbf{b}_{\mathbf{g}_k}^{T} & \mathbf{b}_{\mathbf{a}_k}^{T} \end{bmatrix}^{T} \tag{2.4} $$

It is worth pointing out that our choice of the orientation-error parameterization is guided by the analysis of [63]. That analysis showed that defining the orientation error based on the difference between the true and estimated *global* frame, as in (2.2)-(2.3), is preferable to defining it as the difference between the true and estimated *IMU* frame. Specifically, the latter choice (used, for example, in [80]) causes undesirable terms to appear in the observability matrix of the EKF's linearized system model.

2.3.2 EKF-based SLAM

In EKF-SLAM algorithms, the filter state vector contains the current evolving IMU state, $\mathbf{x}_{E_k}$, and a representation of the feature positions. Thus, the filter state vector at time-step k is defined as:

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_{E_k}^T & \mathbf{f}_1^T & \cdots & \mathbf{f}_{s_k}^T \end{bmatrix}^T \tag{2.5}$$

where $\mathbf{f}_i$, $i = 1, \ldots, s_k$, are the features included in the state vector at time step k. The features can be parameterized in different ways, which will lead to different estimation performance. We discuss the details of the feature parameterization in Section 2.3.4.

The EKF-SLAM equations are well-known, and we therefore only briefly describe them here, to introduce the necessary notation. In standard practice, the IMU measurements are used to propagate the evolving state. To describe the way in which the errors in the propagated state estimate depend on the estimation errors at the previous time step, the EKF employs a linearized equation of the form:

$$\tilde{\mathbf{x}}_{E_{k+1|k}} \simeq \mathbf{\Phi}_{I_k} \tilde{\mathbf{x}}_{E_{k|k}} + \mathbf{w}_{d_k} \tag{2.6}$$

where $\mathbf{\Phi}_{I_k}$ is the IMU error-state transition matrix at time-step k, and $\mathbf{w}_{d_k}$ is a noise vector, with covariance matrix $\mathbf{Q}_{d_k}$. The filter's covariance matrix is propagated according to

$$\mathbf{P}_{k+1|k} = \begin{bmatrix} \mathbf{\Phi}_{I_k} \mathbf{P}_{EE_{k|k}} \mathbf{\Phi}_{I_k}^T + \mathbf{Q}_{d_k} & \mathbf{\Phi}_{I_k} \mathbf{P}_{EF_{k|k}} \\ \mathbf{P}_{EF_{k|k}}^T \mathbf{\Phi}_{I_k}^T & \mathbf{P}_{FF_{k|k}} \end{bmatrix}$$

where $\mathbf{P}_{EE_{k|k}}$ is the covariance matrix of the evolving IMU state, $\mathbf{P}_{FF_{k|k}}$ is the covariance matrix of the features, and $\mathbf{P}_{EF_{k|k}}$ the cross-covariance between them.

On the other hand, the camera measurements are used to perform EKF updates. Specifically, the observation of feature i at time step k is described by the equation:

$$\mathbf{z}_{ik} = \mathbf{h}(\mathbf{x}_{E_k}, \mathbf{f}_i) + \mathbf{n}_{ik} \tag{2.7}$$

where $\mathbf{n}_{ik}$ is the measurement noise vector, modelled as zero-mean Gaussian with covariance matrix $\sigma^2\mathbf{I}_2$, and the function $\mathbf{h}$ describes the camera measurement model, given by:

$$\mathbf{h}(\mathbf{x}_{E_k}, \mathbf{f}_i) = \begin{bmatrix} \frac{{}^{C_k}x_{f_i}}{{}^{C_k}z_{f_i}} \\ \frac{{}^{C_k}y_{f_i}}{{}^{C_k}z_{f_i}} \end{bmatrix} \tag{2.8}$$

In the above equation, the vector ${}^{C_k}\mathbf{p}_{f_i} = [{}^{C_k}x_{f_i} \quad {}^{C_k}y_{f_i} \quad {}^{C_k}z_{f_i}]^T$ is the position of the feature with respect to the camera at time step k:

$${}^{C_k}\mathbf{p}_{f_i} = {}^{C}_{I}\mathbf{R}\, {}^{I_k}_{G}\mathbf{R}\big({}^{G}\mathbf{p}_{f_i} - {}^{G}\mathbf{p}_{I_k}\big) + {}^{C}\mathbf{p}_I \tag{2.9}$$

with $\{{}^{C}_{I}\mathbf{R},\ {}^{C}\mathbf{p}_I\}$ being the rotation and translation between the camera and the IMU. In this chapter, since we focus on algorithms for motion estimation, we assume that $\{{}^{C}_{I}\mathbf{R},\ {}^{C}\mathbf{p}_I\}$ are accurately known (i.e., from hardware-specific knowledge or offline calibration), and thus treated as constants in (2.9). Additionally, we also assume that the measurements of the IMU and the camera have been precisely synchronized. The more general case when the camera-to-IMU configuration is *unknown* is discussed in Chapter 4. It is also important to note that, in (2.8), we assume that the camera is pre-calibrated, and therefore the normalized image coordinates are used, instead of the "raw" image pixel coordinates. The case of imperfectly known camera calibration parameters is treated in Chapter 5.

In EKF-SLAM, the feature observations are used directly for updating the state estimates. To this end, we employ the residual between the actual and expected feature measurement, and its linearized approximation:

$$\begin{aligned}\tilde{\mathbf{z}}_{ik} &= \mathbf{z}_{ik} - \mathbf{h}(\hat{\mathbf{x}}_{E_{k|k-1}}, \hat{\mathbf{f}}_{i_{k|k-1}}) \\ &\simeq \mathbf{H}_{ik}\big(\hat{\mathbf{x}}_{k|k-1}\big)\ \tilde{\mathbf{x}}_{k|k-1} + \mathbf{n}_{ik} \qquad (2.10)\end{aligned}$$

where $\mathbf{H}_{ik}\big(\hat{\mathbf{x}}_{k|k-1}\big)$ is the Jacobian matrix of $\mathbf{h}$ with respect to the filter state, evaluated at the state estimate $\hat{\mathbf{x}}_{k|k-1}$. This is a sparse matrix, containing nonzero blocks only at the locations corresponding to the IMU state (position and orientation) and the i-th feature:

$$\mathbf{H}_{ik}\big(\hat{\mathbf{x}}_{k|k-1}\big) = \Big[\mathbf{H}_{I_{ik}}\big(\hat{\mathbf{x}}_{k|k-1}\big)\ \ \mathbf{0}\ \ \cdots\ \ \mathbf{H}_{\mathbf{f}_{ik}}\big(\hat{\mathbf{x}}_{k|k-1}\big)\ \ \cdots\ \ \mathbf{0}\Big] \qquad (2.11)$$

Once $\tilde{\mathbf{z}}_{ik}$ and $\mathbf{H}_{ik}$ have been computed, a Mahalanobis gating test is performed, and if successful the standard EKF update equations are employed [76]. Depending on the particular feature parameterization used, the exact form of the above Jacobians, as well as the "bookkeeping" required in the filter, will be slightly different.

In VIO, we must ensure that the computational cost of the algorithm remains bounded. To achieve this, features are removed from the state vector immediately once they leave the field of view of the camera. This of course means that the filter cannot process feature re-observations that occur when an area is re-visited. However, such "loop-closure" events do not need to be handled by a VIO algorithm: if desired, they can be handled by a separate algorithm running in parallel, as done for example in [81]. In the "prototypical" VIO scenario, where the camera keeps moving in new areas all the time, the EKF-SLAM algorithm described above will use all the available feature information.

2.3.3 Multi-state-constraint Kalman filter (MSCKF)

In contrast to EKF-SLAM, the MSCKF is an EKF algorithm that maintains in its state vector a sliding window of poses, and uses feature observations to impose probabilistic constraints between these poses [80]. The state vector of the MSCKF at time-step k is defined as:

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_{E_k}^T & \mathbf{x}_{I_{k-1}}^T & \mathbf{x}_{I_{k-2}}^T & \cdots & \mathbf{x}_{I_{k-m}}^T \end{bmatrix}^T \tag{2.12}$$

where $\mathbf{x}_{I_i}$, for $i = k-m, \ldots, k-1$, are the IMU poses at the times the last m images are recorded, consisting of the IMU's orientation and position:

$$\mathbf{x}_{I_i} = \begin{bmatrix} {}^{I_i}_{G}\bar{\mathbf{q}}^T & {}^{G}\mathbf{p}_{I_i}^T \end{bmatrix}^T \tag{2.13}$$

During MSCKF propagation, the IMU measurements are used to propagate the IMU state estimate and the filter covariance matrix, similarly to EKF-SLAM. The difference lies in the way in which the feature measurements are used. Specifically, every time a new image is recorded by the camera, the MSCKF state and covariance are augmented with a copy of the current IMU pose, and the image is processed to extract and match features. Each feature is tracked until all its measurements become available (e.g., until it goes out of the field of view), at which time an update is carried out using *all* the measurements simultaneously.

To present the update equations, we consider the case where the feature f_i, observed from the m poses in the MSCKF state vector, is used for an update at time step k. The first step of the process is to obtain an estimate of the feature position, ${}^G\hat{\mathbf{p}}_{f_i}$. To this end, we use all the feature's measurements to estimate its position via Gauss-Newton minimization [80]. Subsequently, we compute the residuals (for $j = k-m, \ldots, k-1$):

$$\tilde{\mathbf{z}}_{ij} = \mathbf{z}_{ij} - \mathbf{h}(\hat{\mathbf{x}}_{I_{j|k-1}}, {}^G\hat{\mathbf{p}}_{f_i}) \tag{2.14}$$

$$\simeq \mathbf{H}_{\mathbf{x}_{ij}}(\hat{\mathbf{x}}_{I_{j|k-1}}, {}^G\hat{\mathbf{p}}_{f_i})\,\tilde{\mathbf{x}}_{I_{j|k-1}} + \mathbf{H}_{\mathbf{f}_{ij}}(\hat{\mathbf{x}}_{I_{j|k-1}}, {}^G\hat{\mathbf{p}}_{f_i})\,{}^G\tilde{\mathbf{p}}_{f_i} + \mathbf{n}_{ij} \tag{2.15}$$

where $\tilde{\mathbf{x}}_{I_{j|k-1}}$ and ${}^G\tilde{\mathbf{p}}_{f_i}$ are the error of the current estimate for the j-th pose and the error in the feature position respectively, and the matrices $\mathbf{H}_{\mathbf{x}_{ij}}$ and $\mathbf{H}_{\mathbf{f}_{ij}}$ are the corresponding Jacobians, evaluated using $\hat{\mathbf{x}}_{I_{j|k-1}}$, and ${}^G\hat{\mathbf{p}}_{f_i}$. At this point we note that, in the EKF algorithm, to be able to employ a measurement residual, $\tilde{\mathbf{z}}$, for a filter update, we must be able to write this residual in the form $\tilde{\mathbf{z}} \simeq \mathbf{H}\tilde{\mathbf{x}} + \mathbf{n}$, where $\tilde{\mathbf{x}}$ is the error in the state estimate, and $\mathbf{n}$ is a noise vector that is *independent* from $\tilde{\mathbf{x}}$. The residual in (2.15) does not have this form, as the feature position error ${}^G\tilde{\mathbf{p}}_{f_i}$ is correlated to both $\tilde{\mathbf{x}}_{I_{j|k-1}}$ and $\mathbf{n}_{ij}$ (this is because ${}^G\hat{\mathbf{p}}_{f_i}$ is computed as a function of $\hat{\mathbf{x}}_{I_{j|k-1}}$ and $\mathbf{z}_{ij}$, j $= k-m, \ldots, k-1$). Therefore, in the MSCKF we proceed to remove ${}^G\tilde{\mathbf{p}}_{f_i}$ from the residual equations. For this purpose, we first form the vector containing the m residuals from all the feature's measurements:

$$\tilde{\mathbf{z}}_i \simeq \mathbf{H}_{\mathbf{x}_i}(\hat{\mathbf{x}}_{k|k-1}, {}^G\hat{\mathbf{p}}_{f_i})\tilde{\mathbf{x}}_{k|k-1} + \mathbf{H}_{\mathbf{f}_i}(\hat{\mathbf{x}}_{k|k-1}, {}^G\hat{\mathbf{p}}_{f_i})\,{}^G\tilde{\mathbf{p}}_{f_i} + \mathbf{n}_i \tag{2.16}$$

where $\tilde{\mathbf{z}}_i$, $\mathbf{n}_i$, $\mathbf{H}_{\mathbf{f}_i}$, and $\mathbf{H}_{\mathbf{x}_i}$ are block vectors/matrices with elements $\tilde{\mathbf{z}}_{ij}$, $\mathbf{n}_{ij}$, $\mathbf{H}_{\mathbf{f}_{ij}}$ and $\mathbf{H}_{\mathbf{x}_{ij}}$, respectively, defined as:

$$\tilde{\mathbf{z}}_i = \begin{bmatrix} \tilde{\mathbf{z}}_{ik-1} \\ \tilde{\mathbf{z}}_{ik-2} \\ \vdots \\ \tilde{\mathbf{z}}_{ik-m} \end{bmatrix}, \qquad \mathbf{n}_i = \begin{bmatrix} \mathbf{n}_{ik-1} \\ \mathbf{n}_{ik-2} \\ \vdots \\ \mathbf{n}_{ik-m} \end{bmatrix} \qquad \mathbf{H}_{\mathbf{f}_i} = \begin{bmatrix} \mathbf{H}_{\mathbf{f}_{ik-1}} \\ \mathbf{H}_{\mathbf{f}_{ik-2}} \\ \vdots \\ \mathbf{H}_{\mathbf{f}_{ik-m}} \end{bmatrix} \tag{2.17}$$

and

$$\mathbf{H}_{\mathbf{x}_i} = \begin{bmatrix} \mathbf{0}_{2\times15} & \mathbf{H}_{\mathbf{x}_{ik-1}} & \mathbf{0}_{2\times6} & \dots & \mathbf{0}_{2\times6} \\ \mathbf{0}_{2\times15} & \mathbf{0}_{2\times6} & \mathbf{H}_{\mathbf{x}_{ik-2}} & \dots & \mathbf{0}_{2\times6} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_{2\times15} & \mathbf{0}_{2\times6} & \mathbf{0}_{2\times6} & \dots & \mathbf{H}_{\mathbf{x}_{ik-m}} \end{bmatrix} \tag{2.18}$$

Subsequently, we define the residual vector $\tilde{\mathbf{z}}_i^o = \mathbf{V}_i^T \tilde{\mathbf{z}}_i$, where $\mathbf{V}_i$ is a matrix whose columns form a basis for the left nullspace of $\mathbf{H}_{\mathbf{f}_i}$. From (2.16), we obtain:

$$\tilde{\mathbf{z}}_i^o = \mathbf{V}_i^T \tilde{\mathbf{z}}_i \simeq \mathbf{H}_i^o(\hat{\mathbf{x}}_{k|k-1}, {}^G\hat{\mathbf{p}}_{f_i})\, \tilde{\mathbf{x}}_{k|k-1} + \mathbf{n}_i^o \tag{2.19}$$

where $\mathbf{H}_i^o = \mathbf{V}_i^T \mathbf{H}_{\mathbf{x}_i}$ and $\mathbf{n}_i^o = \mathbf{V}_i^T \mathbf{n}_i$. Note that the residual vector $\tilde{\mathbf{z}}_i^o$ is now *independent* of the errors in the feature position, and thus can be used for an EKF update. It should also be mentioned that, for efficiency, $\tilde{\mathbf{z}}_i^o$ and $\mathbf{H}_i^o$ are computed without explicitly forming $\mathbf{V}_i$ [80].

Once $\tilde{\mathbf{z}}_i^o$ and $\mathbf{H}_i^o$ are computed, we proceed to carry out a Mahalanobis gating test for the residual $\tilde{\mathbf{z}}_i^o$. Specifically, we compute:

$$\gamma_i = (\tilde{\mathbf{z}}_i^o)^T \left(\mathbf{H}_i^o \mathbf{P}_{k|k-1} (\mathbf{H}_i^o)^T + \sigma^2 \mathbf{I}\right)^{-1} \tilde{\mathbf{z}}_i^o \tag{2.20}$$

and compare it against a threshold given by the 95-th percentile of the χ^2 distribution with $2m-3$ degrees of freedom ($2m-3$ is the number of elements in the residual vector $\tilde{\mathbf{z}}_i^o$). By stacking together the residuals of all features that pass this gating test, we obtain:

$$\tilde{\mathbf{z}}^o \simeq \mathbf{H}^o \tilde{\mathbf{x}}_{k|k-1} + \mathbf{n}^o \tag{2.21}$$

where $\tilde{\mathbf{z}}^o$, $\mathbf{H}^o$, and $\mathbf{n}^o$ are block vectors/matrices, with block rows $\tilde{\mathbf{z}}_i^o$, $\mathbf{H}_i^o$, and $\mathbf{n}_i^o$, respectively. We can now use the above residual expression, which only involves the IMU poses' errors, for an EKF update. However, if a large number of features are processed at each time instant (the common situation), further computational savings can be achieved. Specifically, in [80] it is shown that instead of using the above residual, we can equivalently compute the thin QR factorization of $\mathbf{H}^o$, written as $\mathbf{H}^o = \mathbf{Q}\mathbf{H}^r$, and then employ for the updates the residual $\tilde{\mathbf{z}}^r$, defined as:

$$\tilde{\mathbf{z}}^r = \mathbf{Q}^T \tilde{\mathbf{z}}^o \simeq \mathbf{H}^r \tilde{\mathbf{x}} + \mathbf{n}^r \tag{2.22}$$

where $\mathbf{n}^r$ is the $r \times 1$ noise vector, with covariance matrix $\sigma^2 \mathbf{I}_r$. Once the residual $\tilde{\mathbf{z}}^r$ and the matrix $\mathbf{H}^r$ have been computed, the standard EKF update equations can be thus implemented to compute the posterior state estimates and the covariance matrix. After the EKF update takes place, we remove from the sliding window those poses whose observed features have all been used for updates. An overview of the MSCKF is given in Algorithm 1.

Algorithm 1 Multi-State-Constraint Kalman Filter (MSCKF)

```
Propagation: Propagate state vector and covariance matrix using IMU readings.

Update: when a new image is recorded,

    • State augmentation: augment the state vector as well as the associated
      covariance matrix with the current IMU position and orientation.

    • Image processing: extract corner features and perform feature matching.

    • Update: for each feature whose track is complete, compute $\tilde{\mathbf{z}}_i^o$ and $\mathbf{H}_i^o$,
      and perform the Mahalanobis test. Use all features that pass the test to
      formulate $\tilde{\mathbf{z}}^r$ and $\mathbf{H}^r$, and perform an EKF update.

    • State management: remove from the state vector those IMU states for
      which all associated features have been processed.
```

2.3.4 Comparison of the MSCKF and EKF-SLAM approaches

We now compare the two VIO methods discussed in the preceding subsections. We start by noting that the MSCKF and EKF-SLAM make use of the *same* measurement information. Specifically, in [67], it is shown that if the system model was linear-Gaussian, the MSCKF estimate for the current IMU pose would be the optimal, maximum a posteriori (MAP) estimate. In the linear-Gaussian case, EKF-SLAM also yields the MAP estimate, since the Kalman filter is a MAP estimator [48]. Thus, if the system models were linear-Gaussian, the EKF-SLAM and the MSCKF algorithms would yield the *same, optimal* estimates for the IMU pose. The differences in their performance arise due to fact that the actual measurement models are not linear, as discussed next.

To test the performance of the methods with the actual nonlinear system models, we performed extensive Monte-Carlo simulation tests. To obtain realistic simulation environments, we first collected a 13-minute, 5.5-km-long real-world dataset, and subsequently generated simulation environments based on the collected dataset, as described in Appendix A.6. In the simulations, the algorithms compared are (i) the MSCKF, (ii) EKF-SLAM with inverse depth parametrization (IDP) [18], (iii) EKF-SLAM with the anchored homogeneous feature parametrization (AHP) [103], and (iv) EKF-SLAM with the "traditional" XYZ feature parametrization (XYZ). Our goal in these simulations is to examine both the accuracy and the consistency of the algorithms. Therefore, we collect statistics for the average normalized estimation error squared (NEES) for the IMU pose (position and orientation), as well as the root mean squared error (RMSE) for the IMU position and orientation. We note that for a consistent estimator the average pose NEES should be 6 (equal to the dimension of the pose error), while a larger NEES value indicates inconsistency.

In all the SLAM algorithms, we wait until κ observations of a feature are available, prior to initializing it in the state vector. For this purpose, we maintain a sliding window of κ poses in the state vector, and when a feature has been observed κ times, all the measurements of the feature are used concurrently to initialize the feature parameterization and its covariance. In our tests, we used the values $\kappa = 2$, $\kappa = 4$, and $\kappa = 6$. Even though for the IDP and AHP approaches it is not necessary to use multiple observations for initialization, our results show that this results in dramatically improved performance. We note that if a feature's track ends after fewer than κ observations, its measurements are processed with the MSCKF measurement model instead. In this way no measurements are discarded, and all the algorithms compared use the same feature observations for fairness.

Fig. 2.1 shows the average NEES for the IMU pose, as well as the RMSE for the IMU position and orientation, averaged over 50 Monte-Carlo trials. This plot corresponds to the case $\kappa = 4$ for the SLAM methods. Moreover, Table 2.1 provides the numerical values for the NEES and RMSE for all the algorithms, and with different values of κ for the SLAM methods. Several interesting conclusions can be drawn from these results. The most important one is that the MSCKF

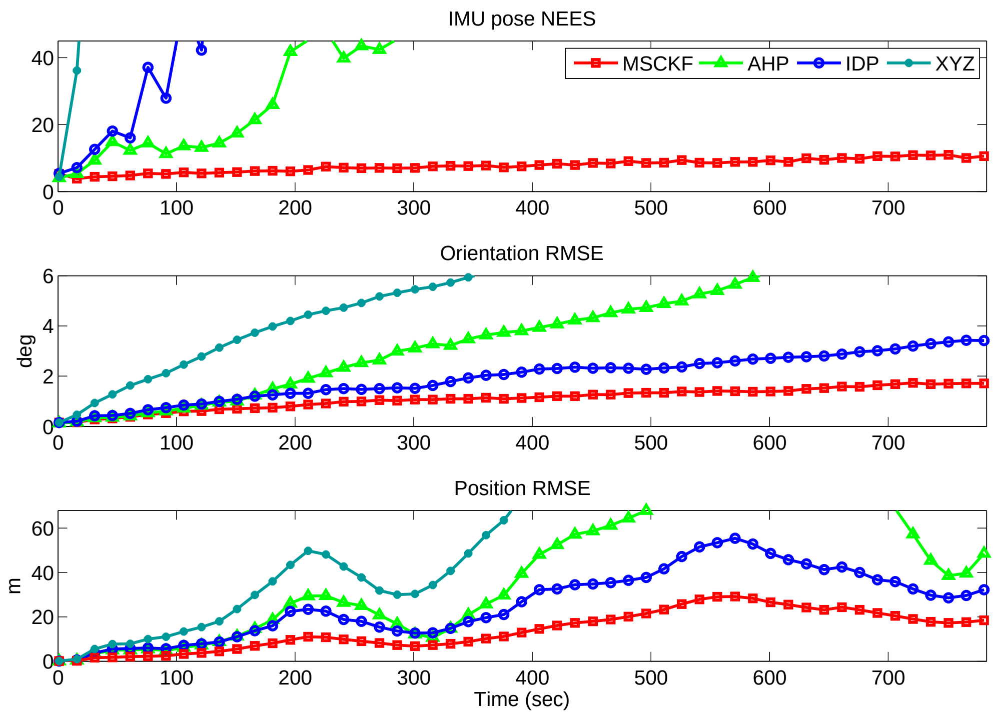


Figure 2.1: Comparison between the MSCKF and EKF-SLAM: The average NEES of the IMU pose and RMSE of the IMU position and orientation. The different lines correspond to the MSCKF (line with squares – red), the AHP (line with triangles – light green), the IDP (line with circles – blue), and the XYZ (dotted line – dark green). Note that due to their large values, the curves for some of the EKF-SLAM methods are not fully visible. Table 2.1 presents the statistics for all the curves.

Table 2.1: Simulation results: Performance metrics (IMU pose NEES, orientation RMSE ($\delta\boldsymbol{\theta}_R$), and position RMSE ($\delta\mathbf{p}_R$)) for the MSCKF and the three EKF-SLAM algorithms with varying number of observations (κ) used for initialization. The values shown are averages over all Monte-Carlo trials and all time steps.

	$\kappa = 2$			$\kappa = 4$			$\kappa = 6$		
Algorithm	$\delta\mathbf{p}_R$ (m)	$\delta\boldsymbol{\theta}_R$ (°)	NEES	$\delta\mathbf{p}_R$ (m)	$\delta\boldsymbol{\theta}_R$ (°)	NEES	$\delta\mathbf{p}_R$ (m)	$\delta\boldsymbol{\theta}_R$ (°)	NEES
XYZ	N/A	N/A	N/A	78.447	5.609	$4.9{\cdot}10^3$	53.469	3.974	$1.3{\cdot}10^3$
IDP	69.502	3.731	$2.2{\cdot}10^3$	26.193	1.916	$2.7{\cdot}10^2$	22.878	1.803	167.261
AHP	67.061	4.795	$2.7{\cdot}10^2$	52.355	4.531	$1.3{\cdot}10^2$	36.858	3.129	48.236

		$\delta\mathbf{p}_R$ (m)	$\delta\boldsymbol{\theta}_R$ (°)	NEES	
MSCKF		14.401	1.102	7.741	

algorithm outperforms all three EKF-SLAM VIO formulations, both in terms of accuracy (smaller RMSE) and in terms of consistency (NEES closer to six). This is a result that we have consistently observed in all our tests, and that we attribute to two main reasons:

- First, all EKF-SLAM algorithms assume that the errors of the IMU state *and* feature positions are jointly Gaussian at each time step. However, due to the nonlinearity of the camera measurement model, this is not a good approximation, particularly for the XYZ parameterization [18]. By intelligently choosing the feature parameterization, as AHP and IDP do, the accuracy and consistency of EKF-SLAM can be improved, as shown in these results. However, these algorithms still perform significantly worse than the MSCKF. Since in the MSCKF the features are never included in the state vector, no assumptions on the feature errors' pdf are needed, thus avoiding a major source of inaccuracy.

- In EKF-SLAM, feature measurements are linearized and processed at *each* time step. By contrast, the MSCKF employs a "delayed linearization" approach: it processes each feature only when *all* its measurements become available. This means that more accurate feature estimates are used in computing Jacobians, leading to more precise calculation of the Kalman gain and state corrections, and ultimately better accuracy.

Examining the different EKF-SLAM methods, we see that in agreement with previous results [18, 103], we find that the performance of the AHP and IDP parameterizations is significantly better than that of the XYZ parameterization. It should be mentioned that, for the XYZ parameterization, initializing features after only two observations is extremely unreliable: the estimator always fails, which does not allow us to obtain reliable statistics. This failure is characterized by a sequence of timesteps in which the filter corrections are very large (and erroneous), after which all residuals fail the Mahalanobis test, no filter updates occur, and the estimation errors increase rapidly. In fact, even if more observations are used for feature initialization, the XYZ parameterization still remains unreliable: for instance, when $\kappa = 4$, the EKF-SLAM with XYZ parametrization fails in 4% of the Monte-Carlo simulations. In the statistics reported in Table 2.1 for the XYZ parametrization, only successful trials are used, to provide more meaningful statistics. Note that no failures were observed in the IDP SLAM, AHP SLAM, or MSCKF algorithms. Moreover, we can observe that the use of more measurements for feature initialization (larger κ) leads to better performance, for all EKF-SLAM algorithms. The improvement as κ increases occurs because, with more measurements, a better initial estimate for the feature is obtained, and thus the filter Jacobians become more accurate and the feature pdf closer to a Gaussian.

In addition to the algorithms' estimation performance, it is also important to examine the computational efficiency of the different methods. For the tests performed above, the MSCKF's average runtime was 0.93 msec per update, while for the EKF-SLAM methods the average runtime was 1.54 msec for XYZ, 3.28 msec for IDP, and 4.45 msec for AHP, when $\kappa = 2$ (measured on a Core i7 processor at 2.66 GHz, with a single-threaded C++ implementation). These observed runtimes agree with the theoretical computational complexity of the algorithms: the MSCKF's computational cost per time step is linear in the number of features, as opposed to cubic for EKF-SLAM. Thus, we can conclude that due to the higher accuracy, consistency, robustness, and computational efficiency, the MSCKF is preferable to EKF-SLAM algorithms for VIO applications.

2.4 EKF Consistency and Relation to Observability

In Table 2.1 we can see that the average IMU-pose NEES for the MSCKF in the simulation tests is 7.741, i.e., above the theoretically expected value of six for a consistent estimator. Moreover, Fig. 2.1 shows that the NEES is gradually increasing over time, reaching an average of 10.6 in the last 100 sec. These results show that the MSCKF becomes *inconsistent*, albeit much less so than EKF-SLAM methods. This inconsistency can become significant in long trajectories, as demonstrated in the results of Section 2.8.2. In the remainder of this chapter, we focus on explaining the cause of the inconsistency, and proposing a solution to it.

To illustrate the main idea of our approach, consider a physical system described by general nonlinear models:

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}) + \mathbf{w} \tag{2.23}$$

$$\mathbf{z} = h(\mathbf{x}) + \mathbf{n} \tag{2.24}$$

where $\mathbf{x}$ is the system state, $\mathbf{u}$ is the control input, $\mathbf{z}$ are the measurements, and finally $\mathbf{w}$ and $\mathbf{n}$ are noise processes. To track the state vector $\mathbf{x}$ on a digital computer, discretization of the continuous-time system model is necessary. Furthermore, when an EKF is used for estimation, the filter equations (e.g., covariance propagation and update, gain computation) rely on a linearized version of the discrete-time model, which has the general form:

$$\tilde{\mathbf{x}}_{k+1} \simeq \mathbf{\Phi}_k \tilde{\mathbf{x}}_k + \mathbf{w}_k \tag{2.25}$$

$$\tilde{\mathbf{z}}_k \simeq \mathbf{H}_k \tilde{\mathbf{x}}_k + \mathbf{n}_k \tag{2.26}$$

where $\mathbf{\Phi}_k$ and $\mathbf{H}_k$ denote the error-state transition matrix and the measurement Jacobian matrix, respectively. Since the EKF equations are derived based on the linearized system model in (2.25)-(2.26), the observability properties of this model play a crucial role in determining the performance of the estimator. Ideally, one would like these properties to match those of the actual, nonlinear system in (2.23)-(2.24): if a certain quantity is unobservable in the actual system, its error should also be unobservable in the linearized model. If this is not the case, “fictitious” information about this quantity will be accumulated by the EKF, which will lead to inconsistency.

The observability properties of the nonlinear system for visual-inertial navigation have recently been studied in [43, 50, 75]. Based on the analysis of these papers, it is now known that when a camera/IMU system moves in a general trajectory, in an environment with a known gravitational acceleration but *no* known features, four degrees of freedom are unobservable: (i) the three corresponding to the global position, and (ii) one corresponding to the rotation about the gravity vector (i.e., the yaw). In Section 2.6, we analyze the observability properties of the linearized system model employed in EKF-based VIO, by studying the nullspace of the observability matrix associated with (2.25)-(2.26):

$$\mathcal{O} \triangleq \begin{bmatrix} \mathbf{H}_b \\ \mathbf{H}_{b+1}\boldsymbol{\Phi}_b \\ \vdots \\ \mathbf{H}_{b+m}\boldsymbol{\Phi}_{b+m-1}\cdots\boldsymbol{\Phi}_b \end{bmatrix} \tag{2.27}$$

The nullspace $\mathcal{O}$ describes the directions of the state space for which no information is provided by the measurements in the time interval $[b, b+m]$, i.e., the unobservable directions. For the linearized system to have observability properties analogous to the actual, nonlinear system, this nullspace should be of dimension four, and should contain the vectors corresponding to global translation and to rotation about gravity. However, our key result, proven in Section 2.6, is that this is not the case when the MSCKF (or an EKF-SLAM method) is employed for VIO. Specifically, due to the way in which the Jacobians are computed in the EKF, the global orientation *appears* to be observable in the linearized model, while it is not in the actual system. As a result of this mismatch, the filter produces too small values for the state covariance matrix (i.e., the filter becomes *inconsistent*), and this in turn degrades accuracy.

Note that, to study the nullspace of the matrix $\mathcal{O}$ in (2.27) for the VIO system, we must have an expression for the error-state transition matrix $\mathbf{\Phi}_k$, for $k = b, \ldots, b+m-1$. In turn, this requires an expression for the IMU's error-state transition matrix, defined in (2.6). Therefore, before proceeding with the observability analysis, we must derive an expression for this matrix. This is the focus of the next section.

2.5 Computing the IMU Error-State Transition Matrix

Most existing methods for computing $\mathbf{\Phi}_I$ stem from the integration of the differential equation $\dot{\mathbf{\Phi}}_I(t, t_k) = \mathbf{F}(t)\mathbf{\Phi}_I(t, t_k)$, where $\mathbf{F}(t)$ is the Jacobian of the continuous-time system model of the IMU motion [113]. For instance [80, 107] employ Runge-Kutta numerical integration, [119, 120] use a closed-form, approximate solution to the differential equation, while several algorithms (especially in the GPS-aided inertial navigation community) employ the simple approximation $\mathbf{\Phi}_I \simeq \mathbf{I} + \mathbf{F}\Delta t$ that is equivalent to using one-step Euler integration (e.g., [25, 29, 72, 116, 123]). All these methods for computing $\mathbf{\Phi}_I$ have the disadvantage that, being numerical in nature, they are not amenable to theoretical analysis. More importantly, however, when $\mathbf{\Phi}_I$ is computed numerically and/or approximately, we have no guarantees about its properties. As a result, if $\mathbf{\Phi}_I$ is computed in this fashion, we *cannot* guarantee that the observability matrix of the linearized VIO system (2.27) will have the desirable nullspace properties, a prerequisite for consistent estimation.

In what follows, we describe how the IMU error-state transition matrix can be computed in closed form, as a function of the state estimates. To this end, we first examine what motion information we can infer from the IMU data, and how this information can be used for state propagation. This will enable us to derive an expression for $\mathbf{\Phi}_I$ that holds independently of the particular method used to integrate the IMU signals. We note that the expression derived here can be employed in any estimation problem that uses IMU measurements for state propagation (e.g., GPS-aided inertial navigation, vision-aided inertial navigation, etc).

2.5.1 What information do the IMU measurements provide?

The IMU's gyroscopes and accelerometers give sampled measurements of the following continuous-time signals:

$$\boldsymbol{\omega}_m(t) = {}^I\boldsymbol{\omega}(t) + \mathbf{b_g}(t) + \mathbf{n_r}(t) \tag{2.28}$$

$$\mathbf{a}_m(t) = {}^I_G\mathbf{R}(t)\left({}^G\mathbf{a}_I(t) - {}^G\mathbf{g}\right) + \mathbf{b_a}(t) + \mathbf{n_a}(t) \tag{2.29}$$

where ${}^{I}\boldsymbol{\omega}(t)$ and ${}^{G}\mathbf{a}_I(t)$ denote the IMU angular rate and linear acceleration, respectively, $\mathbf{n}_\mathbf{r}(t)$ and $\mathbf{n}_\mathbf{a}(t)$ are white Gaussian noise processes, and ${}^{G}\mathbf{g}$ is the gravity vector expressed in the global frame.

Equation (2.28) shows that the IMU gyroscopes provide measurements of the rotational velocity, expressed in the IMU frame. Using these measurements, we can only infer the *relative* rotation of the IMU between two time instants. Moreover, (2.29) shows that the IMU accelerometers measure specific force, which includes both the body and gravitational acceleration, expressed in the local frame. These signals provide us with information about the velocity change expressed in the *local* IMU frame, and must be "gravity-compensated" before use for state propagation. In what follows, we momentarily assume that we have access to the *continuous-time* signals $\boldsymbol{\omega}_m(t)$ and $\mathbf{a}_m(t)$ in the time interval $[t_k, t_{k+1}]$ (corresponding to the transition from time-step k to $k+1$), and show how these signals can be used for state propagation. In each case, we will show how $\boldsymbol{\omega}_m(t)$ and $\mathbf{a}_m(t)$ can be used to extract the local motion information, and how the local quantities can be used to propagate the global orientation, position, and velocity estimates.

2.5.1.1 Gyroscope measurements

The orientation of the IMU frame at time t_{k+1} with respect to the IMU frame at t_k (i.e., the relative rotation) can be computed by integrating a differential equation, whose form depends on the selected representation of orientation. In the unit-quaternion representation [113], the relative rotation of the IMU between t_k and t_{k+1} is described by a 4×1 unit quaternion ${}^{I_{k+1}}_{I_k}\bar{\mathbf{q}}$. To compute an estimate of ${}^{I_{k+1}}_{I_k}\bar{\mathbf{q}}$ using $\boldsymbol{\omega}_m(t)$, we first obtain the estimated rotational velocity in $[t_k, t_{k+1}]$ as

$$\hat{\boldsymbol{\omega}}(t) = \boldsymbol{\omega}_m(t) - \hat{\mathbf{b}}_g(t) \tag{2.30}$$

and then integrate the differential equation:

$${}^{I_t}_{I_k}\dot{\hat{\bar{\mathbf{q}}}} = \frac{1}{2}\begin{bmatrix} -\lfloor \hat{\boldsymbol{\omega}}(t)\times\rfloor & \hat{\boldsymbol{\omega}}(t) \\ -\hat{\boldsymbol{\omega}}(t)^T & 0 \end{bmatrix} {}^{I_t}_{I_k}\hat{\bar{\mathbf{q}}}, \qquad t \in [t_k, t_{k+1}] \tag{2.31}$$

with initial condition ${}^{I_k}_{I_k}\hat{\bar{\mathbf{q}}} = [0 \;\; 0 \;\; 0 \;\; 1]^T$. The relative orientation estimate ${}^{I_{k+1}}_{I_k}\hat{\bar{\mathbf{q}}}$, computed from the above differential equation, can be employed for propagating the IMU *global* orientation estimate as follows:

$${}^{I_{k+1}}_{G}\hat{\bar{\mathbf{q}}} = {}^{I_{k+1}}_{I_k}\hat{\bar{\mathbf{q}}} \otimes {}^{I_k}_{G}\hat{\bar{\mathbf{q}}} \tag{2.32}$$

2.5.1.2 Accelerometer measurements

Using $\mathbf{a}_m(t)$ and an estimate of the accelerometer bias, we can obtain an estimate of the IMU's acceleration in the global frame as (see (2.29)):

$${}^{G}\hat{\mathbf{a}}_I(t) = {}^{G}_{I_t}\hat{\mathbf{R}} \left(\mathbf{a}_m(t) - \hat{\mathbf{b}}_{\mathbf{a}}(t)\right) + {}^{G}\mathbf{g} \tag{2.33}$$

Integrating this signal twice in the time interval $[t_k, t_{k+1}]$ gives the velocity and position propagation equations:

$$\begin{aligned} {}^{G}\hat{\mathbf{v}}_{I_{k+1}} &= {}^{G}\hat{\mathbf{v}}_{I_k} + \int_{t_k}^{t_{k+1}} {}^{G}\hat{\mathbf{a}}_I(\tau) d\tau \\ &= {}^{G}\hat{\mathbf{v}}_{I_k} + \int_{t_k}^{t_{k+1}} {}^{G}_{I_\tau}\hat{\mathbf{R}}\left(\mathbf{a}_m(\tau) - \hat{\mathbf{b}}_{\mathbf{a}}(\tau)\right) d\tau + {}^{G}\mathbf{g}\Delta t && (2.34) \\ &= {}^{G}\hat{\mathbf{v}}_{I_k} + {}^{G}_{I_k}\hat{\mathbf{R}}\,\hat{\mathbf{s}}_k + {}^{G}\mathbf{g}\Delta t && (2.35) \end{aligned}$$

and

$$
\begin{aligned}
{}^{G}\hat{\mathbf{p}}_{I_{k+1}} &= {}^{G}\hat{\mathbf{p}}_{I_k} + \int_{t_k}^{t_{k+1}} {}^{G}\hat{\mathbf{v}}_{I_\tau} d\tau \\
&= {}^{G}\hat{\mathbf{p}}_{I_k} + {}^{G}\hat{\mathbf{v}}_{I_k}\Delta t + {}^{G}_{I_k}\hat{\mathbf{R}}\,\hat{\mathbf{y}}_k + \frac{1}{2}{}^{G}\mathbf{g}\Delta t^2
\end{aligned} \tag{2.36}
$$

where $\Delta t = t_{k+1} - t_k$, and

$$
\hat{\mathbf{s}}_k = \int_{t_k}^{t_{k+1}} {}^{I_k}_{I_\tau}\hat{\mathbf{R}}\Big(\mathbf{a}_m(\tau) - \hat{\mathbf{b}}_{\mathbf{a}}(\tau)\Big)d\tau \tag{2.37}
$$

$$
\hat{\mathbf{y}}_k = \int_{t_k}^{t_{k+1}} \int_{t_k}^{s} {}^{I_k}_{I_\tau}\hat{\mathbf{R}}\Big(\mathbf{a}_m(\tau) - \hat{\mathbf{b}}_{\mathbf{a}}(\tau)\Big)d\tau ds \tag{2.38}
$$

Note that the terms $\hat{\mathbf{s}}_k$ and $\hat{\mathbf{y}}_k$ depend *only* on the values of $\mathbf{a}_m(t)$ and $\boldsymbol{\omega}_m(t)$ in the time interval $[t_k, t_{k+1}]$, as well as on the IMU biases. These terms express the information provided by the IMU about the change in the IMU velocity and position in $[t_k, t_{k+1}]$. As shown in (2.35) and (2.36), to use $\hat{\mathbf{s}}_k$ and $\hat{\mathbf{y}}_k$ to propagate the global velocity and position estimates, we must express them in the global frame (via the rotation matrix ${}^{G}_{I_k}\hat{\mathbf{R}}$), and account for the gravitational acceleration.

We note that in [72] it is shown how the IMU measurements can be "pre-integrated", so that they can be used even without an initial guess for the state. While [72] follows a reasoning similar to the one presented here, we go one step further, and use this analysis to obtain a closed-form expression for the error-state transition matrix.

2.5.2 Discrete-time IMU propagation

To derive equations (2.31)-(2.32) and (2.35)-(2.38), it was assumed that the signals $\boldsymbol{\omega}_m(t)$ and $\mathbf{a}_m(t)$ were available in the entire interval $[t_k, t_{k+1}]$. In practice, however, the IMU provides samples of $\mathbf{a}_m(t)$ and $\boldsymbol{\omega}_m(t)$ at the discrete times t_k and t_{k+1}. To use these measurements for state propagation, it is necessary to employ additional assumptions about the time evolution of $\mathbf{a}_m(t)$ and $\boldsymbol{\omega}_m(t)$ between the two times for which samples are available. For instance, we can assume that these signals remain constant for the entire period (equal to their values at either t_k or t_{k+1}), or that they change linearly between the sampled values. These assumptions will introduce approximations, which will be small if the sample rate is sufficiently high. We stress however, that some approximation is unavoidable, since turning a continuous-time signal to a sampled one leads to loss of information[2].

In what follows, we describe the integration approach followed in our implementation. In our work, since the IMU biases are modelled as random-walk processes, their estimates remain constant during propagation: $\hat{\mathbf{b}}_{\mathbf{g}_{k+1}} = \hat{\mathbf{b}}_{\mathbf{g}_k}$ and $\hat{\mathbf{b}}_{\mathbf{a}_{k+1}} = \hat{\mathbf{b}}_{\mathbf{a}_k}$. To propagate the IMU pose in time, at time t_{k+1} we use the IMU samples recorded at t_k and t_{k+1} and assume that the signals $\boldsymbol{\omega}_m(t)$ and $\mathbf{a}_m(t)$ change linearly between these two time instants. With this assumption, we numerically integrate (2.31) using fourth-order Runge-Kutta to obtain ${}^{I_{k+1}}_{I_k}\hat{\bar{\mathbf{q}}}$, and propagate the IMU orientation using (2.32). For the position and velocity, we employ equations (2.35) and (2.36), where the quantities $\hat{\mathbf{s}}_k$ and $\hat{\mathbf{y}}_k$ are computed using Simpson integration of (2.37) and (2.38).

[2]Note that, if the signals are known to be band-limited, more advanced signal-reconstruction methods can be employed. However this requires additional assumptions about the motion characteristics and/or the sensor, which are not always appropriate.

2.5.3 Computing $\mathbf{\Phi}_{I_k}$

We now turn our attention to computing the IMU error-state transition matrix shown in (2.6), which can be done by direct linearization of the state-propagation equations (2.32), (2.35), and (2.36). For clarity, we here show the derivation of $\mathbf{\Phi}_{I_k}$ omitting the IMU biases, while the full result for the case where the biases are included in the state vector is shown in Appendix A.1. Starting with the orientation error, we note that the orientation-error definition in (2.2)-(2.3) satisfies:

$$ {}^{I}_{G}\mathbf{R} \simeq {}^{I}_{G}\hat{\mathbf{R}} \left(\mathbf{I}_3 - \lfloor {}^{G}\tilde{\boldsymbol{\theta}}_I \times \rfloor \right) \tag{2.39} $$

Moreover, the estimated rotation in the time interval $[t_k, t_{k+1}]$ is corrupted by an error due to the inaccuracy of the gyroscope measurements as well as the assumptions employed during integration. We define this error based on the expression ${}^{I_{k+1}}_{I_k}\bar{\mathbf{q}} = {}^{I_{k+1}}_{I_k}\hat{\bar{\mathbf{q}}} \otimes \delta\bar{\mathbf{q}}_{\Delta t}$, from which we obtain

$$ {}^{I_{k+1}}_{I_k}\mathbf{R} \simeq {}^{I_{k+1}}_{I_k}\hat{\mathbf{R}} \left(\mathbf{I} - \lfloor \tilde{\boldsymbol{\theta}}_{\Delta t} \times \rfloor \right) \tag{2.40} $$

where $\tilde{\boldsymbol{\theta}}_{\Delta k}$ is a 3×1 error vector. Substituting (2.40) and (2.39) into the expression relating the true rotation matrices, ${}^{I_{k+1}}_{G}\mathbf{R} = {}^{I_{k+1}}_{I_k}\mathbf{R}\, {}^{I_k}_{G}\mathbf{R}$, and removing second-order terms, we obtain the following linearized expression for the orientation-error propagation:

$$ {}^{G}\tilde{\boldsymbol{\theta}}_{I_{k+1}} \simeq {}^{G}\tilde{\boldsymbol{\theta}}_{I_k} + \hat{\mathbf{R}}_k^T \tilde{\boldsymbol{\theta}}_{\Delta t} \tag{2.41} $$

where we used the shorthand notation ${}^{I_k}_{G}\hat{\mathbf{R}} = \hat{\mathbf{R}}_k$. For the velocity error, we linearize (2.35) using (2.39), to obtain the linearized error-propagation equation:

$$ {}^{G}\tilde{\mathbf{v}}_{I_{k+1}} \simeq -\lfloor \hat{\mathbf{R}}_k^T \hat{\mathbf{s}}_k \times \rfloor {}^{G}\tilde{\boldsymbol{\theta}}_{I_k} + {}^{G}\tilde{\mathbf{v}}_{I_k} + \hat{\mathbf{R}}_k^T \tilde{\mathbf{s}}_k \tag{2.42} $$

The term $\tilde{\mathbf{s}}_k = \mathbf{s}_k - \hat{\mathbf{s}}_k$ is the error in $\hat{\mathbf{s}}_k$, which depends only on the IMU measurement noise and the assumptions employed during integration. Similarly, for the position we obtain:

$$ {}^G\tilde{\mathbf{p}}_{I_{k+1}} \simeq -\lfloor \hat{\mathbf{R}}_k^T \hat{\mathbf{y}}_k \times \rfloor {}^G\tilde{\boldsymbol{\theta}}_{I_k} + {}^G\tilde{\mathbf{v}}_{I_k}\Delta t + {}^G\tilde{\mathbf{p}}_{I_k} + \hat{\mathbf{R}}_k^T \tilde{\mathbf{y}}_k \tag{2.43} $$

where $\tilde{\mathbf{y}}_k = \mathbf{y}_k - \hat{\mathbf{y}}_k$. By combining (2.41), (2.42) and (2.43), we can now write:

$$ \underbrace{\begin{bmatrix} {}^G\tilde{\boldsymbol{\theta}}_{I_{k+1}} \\ {}^G\tilde{\mathbf{p}}_{I_{k+1}} \\ {}^G\tilde{\mathbf{v}}_{I_{k+1}} \end{bmatrix}}_{\tilde{\mathbf{x}}_{E_{k+1}}} = \underbrace{\begin{bmatrix} \mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ -\lfloor \hat{\mathbf{R}}_k^T \hat{\mathbf{y}}_k \times \rfloor & \mathbf{I}_3 & \Delta t \mathbf{I}_3 \\ -\lfloor \hat{\mathbf{R}}_k^T \hat{\mathbf{s}}_k \times \rfloor & \mathbf{0}_3 & \mathbf{I}_3 \end{bmatrix}}_{\boldsymbol{\Phi}_{I_k}} \underbrace{\begin{bmatrix} {}^G\tilde{\boldsymbol{\theta}}_{I_k} \\ {}^G\tilde{\mathbf{p}}_{I_k} \\ {}^G\tilde{\mathbf{v}}_{I_k} \end{bmatrix}}_{\tilde{\mathbf{x}}_{E_k}} + \underbrace{\begin{bmatrix} \hat{\mathbf{R}}_k^T \tilde{\boldsymbol{\theta}}_{\Delta t} \\ \hat{\mathbf{R}}_k^T \tilde{\mathbf{y}}_k \\ \hat{\mathbf{R}}_k^T \tilde{\mathbf{s}}_k \end{bmatrix}}_{\mathbf{w}_{d_k}} \tag{2.44} $$

It is important to note that the above expression for $\boldsymbol{\Phi}_{I_k}$ has an intuitive explanation. We see that: (i) The diagonal block elements are all identity matrices, which shows that the errors in the IMU state at time t_k "carry over" to the next time step, as expected. (ii) The velocity error, multiplied by Δt, affects the position error at time t_{k+1}, and (iii) The orientation error at time t_k is multiplied by the "lever-arms" $\hat{\mathbf{R}}_k^T \hat{\mathbf{y}}_k$ and $\hat{\mathbf{R}}_k^T \hat{\mathbf{s}}_k$, causing accumulation of errors in position and velocity. To write the state transition matrix as a function of the state estimates only, we solve (2.35) and (2.36) for $\hat{\mathbf{s}}_k$ and $\hat{\mathbf{y}}_k$, respectively, and substitute in (2.44) to obtain:

$$ \boldsymbol{\Phi}_{I_k}(\hat{\mathbf{x}}_{E_{k+1}}, \hat{\mathbf{x}}_{E_k}) = \begin{bmatrix} \mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \boldsymbol{\Phi}_{\mathbf{pq}}(\hat{\mathbf{x}}_{E_{k+1}}, \hat{\mathbf{x}}_{E_k}) & \mathbf{I}_3 & \Delta t \mathbf{I}_3 \\ \boldsymbol{\Phi}_{\mathbf{vq}}(\hat{\mathbf{x}}_{E_{k+1}}, \hat{\mathbf{x}}_{E_k}) & \mathbf{0}_3 & \mathbf{I}_3 \end{bmatrix} \tag{2.45} $$

$$ \boldsymbol{\Phi}_{\mathbf{pq}}(\hat{\mathbf{x}}_{E_{k+1}}, \hat{\mathbf{x}}_{E_k}) = -\lfloor \left({}^G\hat{\mathbf{p}}_{I_{k+1}} - {}^G\hat{\mathbf{p}}_{I_k} - {}^G\hat{\mathbf{v}}_{I_k}\Delta t - \frac{1}{2} {}^G\mathbf{g}\Delta t^2 \right) \times \rfloor $$

$$ \boldsymbol{\Phi}_{\mathbf{vq}}(\hat{\mathbf{x}}_{E_{k+1}}, \hat{\mathbf{x}}_{E_k}) = -\lfloor \left({}^G\hat{\mathbf{v}}_{I_{k+1}} - {}^G\hat{\mathbf{v}}_{I_k} - {}^G\mathbf{g}\Delta t \right) \times \rfloor $$

We stress that this matrix is a closed-form function of the state estimates *only*. Thus, it can be employed regardless of the way in which the integration of the equations (2.31), (2.37) and (2.38) is carried out.

Note that in different implementations of IMU propagation, the form of the equations being integrated may be different from those shown above (for example, for velocity propagation one may choose to numerically compute the integral in (2.34), instead of breaking it into two terms as in (2.35)). However, this does not change the nature of the information provided by the IMU measurements, and thus does not (in fact, *should* not) change the way in which the errors in a state estimate propagate to future estimates. This way is described by the matrix in (2.45), as discussed above.

2.6 Observability Properties of the MSCKF System Model

We now employ the closed-form expression for $\mathbf{\Phi}_{I_k}$ derived in the preceding section to analyze the observability properties of the MSCKF's system model. To simplify the presentation, we here carry out the analysis for the case where the IMU biases are not included in the estimated state vector. These biases are known to be observable [43], and thus their inclusion would not change the key result of this analysis, which is the erroneous decrease in the dimension of the nullspace of the observability matrix[3]. The fact that the analysis also holds for the case where the biases are included in the state vector is demonstrated by the results presented in Sections 2.8 and 2.9. In the implementation used for all our simulation and experimental results, the biases are included in the state vector, as described in Section 2.3.1.

A difficulty that arises is that the linearized model used by the MSCKF has a complicated form (see, e.g., (2.19)), which makes direct analysis cumbersome. However, we can circumvent this issue by employing the result of [67], which shows that given a linear (or linearized) model, the EKF-SLAM and MSCKF measurement models are equivalent. This means that we can examine the observability properties of the MSCKF's linearized model, by analyzing those of the corresponding EKF-SLAM model. Note that we cannot directly employ the SLAM Jacobians in (2.11), however, because the MSCKF and EKF-SLAM linearize the measurements using different state estimates: in the MSCKF, a single estimate for each feature is used for computing all the Jacobians involving this feature (see (2.16)), in contrast to EKF-SLAM. Therefore, to analyze the observability properties of the linearized system model used in the MSCKF, we must use the MSCKF's linearization points in the EKF-SLAM Jacobians.

We consider a state vector containing the IMU orientation, position, and velocity, as well as the positions of all the landmarks observed in the time inter-

[3]If we include in the state additional quantities that are known to be observable, this will augment the observability matrix (2.27) with additional, linearly independent, columns and will not affect the dimension of the nullspace of $\mathcal{O}$.

val $[b, b+m]$. For this state vector, the error-state transition matrix (using the MSCKF's linearization point) at time step k is given by:

$$\mathbf{\Phi}_k(\hat{\mathbf{x}}_{E_{k+1|k}}, \hat{\mathbf{x}}_{E_{k|k}}) = \begin{bmatrix} \mathbf{\Phi}_{I_k}(\hat{\mathbf{x}}_{E_{k+1|k}}, \hat{\mathbf{x}}_{E_{k|k}}) & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{3s_k \times 3s_k} \end{bmatrix} \tag{2.46}$$

where s_k is the number of landmarks. Turning to the measurement Jacobians, we note that if feature i is processed at time-step $\alpha_i + 1$, then the Jacobians of the measurement model (2.8) are evaluated with the state estimates computed using all measurements up to α_i, and the feature position estimate ${}^G\hat{\mathbf{p}}_{f_i}$ computed via triangulation. Thus, the measurement Jacobian we use in our analysis becomes (see (2.11) and (2.15)):

$$\mathbf{H}_{ik}(\hat{\mathbf{x}}_{E_{k|\alpha_i}}, {}^G\hat{\mathbf{p}}_{f_i}) = \begin{bmatrix} \mathbf{H}_{I_{ik}}(\hat{\mathbf{x}}_{E_{k|\alpha_i}}, {}^G\hat{\mathbf{p}}_{f_i}) & \mathbf{0} & \cdots & \mathbf{H}_{\mathbf{f}_{ik}}(\hat{\mathbf{x}}_{E_{k|\alpha_i}}, {}^G\hat{\mathbf{p}}_{f_i}) & \cdots & \mathbf{0} \end{bmatrix} \tag{2.47}$$

where the Jacobians of (2.8) with respect to the IMU pose and the feature position are given by

$$\mathbf{H}_{\mathbf{f}_{ik}}(\hat{\mathbf{x}}_{E_{k|\alpha_i}}, {}^G\hat{\mathbf{p}}_{f_i}) = \mathbf{J}_{ik}(\hat{\mathbf{x}}_{E_{k|\alpha_i}}, {}^G\hat{\mathbf{p}}_{f_i})\, {}^C_I\mathbf{R}\, \hat{\mathbf{R}}_{k|\alpha_i} \tag{2.48}$$

$$\mathbf{H}_{I_{ik}}(\hat{\mathbf{x}}_{E_{k|\alpha_i}}, {}^G\hat{\mathbf{p}}_{f_i}) = \mathbf{H}_{\mathbf{f}_{ik}}(\hat{\mathbf{x}}_{E_{k|\alpha_i}}, {}^G\hat{\mathbf{p}}_{f_i}) \begin{bmatrix} \lfloor({}^G\hat{\mathbf{p}}_{f_i} - {}^G\hat{\mathbf{p}}_{I_{k|\alpha_i}})\times\rfloor & -\mathbf{I}_3 & \mathbf{0}_3 \end{bmatrix}$$

$$\mathbf{J}_{ik}(\hat{\mathbf{x}}_{E_{k|\alpha_i}}, {}^G\hat{\mathbf{p}}_{f_i}) = \frac{1}{{}^{C_k}\hat{z}_{f_i}} \begin{bmatrix} 1 & 0 & \frac{-{}^{C_k}\hat{x}_{f_i}}{{}^{C_k}\hat{z}_{f_i}} \\ 0 & 1 & \frac{-{}^{C_k}\hat{y}_{f_i}}{{}^{C_k}\hat{z}_{f_i}} \end{bmatrix} \tag{2.49}$$

with $[{}^{C_k}\hat{x}_{f_i} \quad {}^{C_k}\hat{y}_{f_i} \quad {}^{C_k}\hat{z}_{f_i}]^T$ being the estimate of the feature position with respect to the camera:

$$\begin{bmatrix} {}^{C_k}\hat{x}_{f_i} \\ {}^{C_k}\hat{y}_{f_i} \\ {}^{C_k}\hat{z}_{f_i} \end{bmatrix} = {}^C_I\mathbf{R}\, \hat{\mathbf{R}}_{k|\alpha_i}({}^G\hat{\mathbf{p}}_{f_i} - {}^G\hat{\mathbf{p}}_{I_{k|\alpha_i}}) + {}^C\mathbf{p}_I \tag{2.50}$$

By substitution of (2.46) and (2.47) in (2.27), we can therefore study the observability properties of the linearized system model of the MSCKF. Before doing that however, it is interesting to first examine what the observability matrix would look like in the "ideal" case where the *true* state estimates were used in computing all Jacobians.

2.6.1 "Ideal" observability matrix

To derive the "ideal" observability matrix, we evaluate the state transition matrix as $\mathbf{\Phi}(\mathbf{x}_{E_{k+1}}, \mathbf{x}_{E_k})$ (see (2.46)), and evaluate the Jacobian matrix in (2.47) using the true states. Substituting these matrices in (2.27) yields the following result for the block row of the observability matrix corresponding to the observation of feature i at time step k:

$$\check{\mathcal{O}}_{ik} = \check{\mathbf{M}}_{ik}\Big[\check{\mathbf{\Gamma}}_{ik} \;\; -\mathbf{I}_3 \;\; -\Delta t_k \mathbf{I}_3 \;\; \mathbf{0}_3 \;\; \cdots \;\; \mathbf{I}_3 \;\; \cdots \;\; \mathbf{0}_3\Big] \tag{2.51}$$

$$\check{\mathbf{M}}_{ik} = \check{\mathbf{J}}_{ik}\, {}^C_I\mathbf{R}\,\mathbf{R}_k \tag{2.52}$$

$$\check{\mathbf{\Gamma}}_{ik} = \left\lfloor \left({}^G\mathbf{p}_{f_i} - {}^G\mathbf{p}_{I_b} - {}^G\mathbf{v}_{I_b}\Delta t_k - \frac{1}{2}{}^G\mathbf{g}\Delta t_k^2\right) \times \right\rfloor \tag{2.53}$$

In the above equations, Δt_k denotes the time interval between time steps b and k, and we have used the symbol "ˇ" to denote a matrix computed using the true state values. At this point, if we define the matrix $\mathbf{N}$ as:

$$\mathbf{N} = \begin{bmatrix} \mathbf{0}_3 & \mathbf{R}_b{}^G\mathbf{g} \\ \mathbf{I}_3 & -\lfloor {}^G\mathbf{p}_{I_b} \times \rfloor {}^G\mathbf{g} \\ \mathbf{0}_3 & -\lfloor {}^G\mathbf{v}_{I_b} \times \rfloor {}^G\mathbf{g} \\ \mathbf{I}_3 & -\lfloor {}^G\mathbf{p}_{f_1} \times \rfloor {}^G\mathbf{g} \\ \mathbf{I}_3 & -\lfloor {}^G\mathbf{p}_{f_2} \times \rfloor {}^G\mathbf{g} \\ \vdots & \vdots \\ \mathbf{I}_3 & -\lfloor {}^G\mathbf{p}_{f_N} \times \rfloor {}^G\mathbf{g} \end{bmatrix} \tag{2.54}$$

it is easy to verify that $\check{\mathcal{O}}_{ik} \cdot \mathbf{N} = \mathbf{0}_{2\times 4}$. Since this holds for any i and any k (i.e., for all block rows of the observability matrix), we conclude that $\check{\mathcal{O}} \cdot \mathbf{N} = \mathbf{0}$. In addition, the four columns of $\mathbf{N}$ are linearly independent, which implies that they form a basis for the nullspace of the observability matrix $\check{\mathcal{O}}$ (in Appendix A.2, we prove that no additional basis vectors can be found for the nullspace).

In other words, the above shows that the observability matrix, when all Jacobians are computed using the true states, has a nullspace of dimension four. It is also interesting to examine the physical interpretation of the nullspace basis found above. We see that the first three vectors correspond to global translation of the state vector, while the last column corresponds to rotations about gravity, i.e., the yaw (see Appendix A.3 for the detailed derivations). Thus, if we were able to estimate all the Jacobians using the true state estimates, the observability properties of the linearized system model would match those of the nonlinear system, as desired.

2.6.2 MSCKF observability matrix

Using (2.46) and (2.47) in (2.27), the block row of the observability matrix $\mathcal{O}$ corresponding to the observation of feature i at time-step k becomes:

$$\mathcal{O}_{ik} = \mathbf{M}_{ik}\Big[\mathbf{\Gamma}_{ik} + \Delta\mathbf{\Gamma}_{ik} \quad -\mathbf{I}_3 \quad -\Delta t_k\mathbf{I}_3 \;\; \mathbf{0}_3 \;\; \cdots \;\; \mathbf{I}_3 \;\; \cdots \;\; \mathbf{0}_3\Big] \tag{2.55}$$

where

$$\mathbf{M}_{ik} = \mathbf{J}_{ik}(\hat{\mathbf{x}}_{E_{k|\alpha_i}}, {}^G\hat{\mathbf{p}}_{f_i})\, {}^C_I\mathbf{R}\; \hat{\mathbf{R}}_{k|\alpha_i} \tag{2.56}$$

$$\mathbf{\Gamma}_{ik} = \big\lfloor\big({}^G\hat{\mathbf{p}}_{f_i} - {}^G\hat{\mathbf{p}}_{I_{b|b}} - {}^G\hat{\mathbf{v}}_{I_{b|b}}\Delta t_k - \frac{1}{2}{}^G\mathbf{g}\Delta t_k^2\big) \times \big\rfloor \tag{2.57}$$

$$\Delta\mathbf{\Gamma}_{ik} = \lfloor{}^G\hat{\mathbf{p}}_{I_{k|k-1}} - {}^G\hat{\mathbf{p}}_{I_{k|\alpha_i}} \times\rfloor + \sum_{j=b+1}^{k-1}(\mathbf{E}_{\mathbf{p}}^j + \sum_{s=b+1}^{j}\mathbf{E}_{\mathbf{v}}^s\Delta t) \tag{2.58}$$

with

$$\mathbf{E}_{\mathbf{p}}^{j} = \lfloor {}^{G}\hat{\mathbf{p}}_{I_{j|j-1}} - {}^{G}\hat{\mathbf{p}}_{I_{j|j}} \times \rfloor \tag{2.59}$$

$$\mathbf{E}_{\mathbf{v}}^{j} = \lfloor {}^{G}\hat{\mathbf{v}}_{I_{j|j-1}} - {}^{G}\hat{\mathbf{v}}_{I_{j|j}} \times \rfloor \tag{2.60}$$

By comparing (2.55)-(2.58) to (2.51)-(2.53) we see that the structure of the observability matrix in both cases is similar. The key difference is that when the Jacobians are evaluated using the state *estimates* instead of the true states, the "disturbance" term $\Delta\mathbf{\Gamma}_{ik}$ appears. While $\Delta\mathbf{\Gamma}_{ik}$ is quite complex, we can observe that it contains terms that depend on the corrections (e.g., ${}^{G}\hat{\mathbf{p}}_{I_{j|j}} - {}^{G}\hat{\mathbf{p}}_{I_{j|j-1}}$, ${}^{G}\hat{\mathbf{v}}_{I_{j|j}} - {}^{G}\hat{\mathbf{v}}_{I_{j|j-1}}$) that the filter applies at different time steps. Since these corrections are random, the term $\Delta\mathbf{\Gamma}_{ik}$ is a random one, and this "destroys" the special structure of the observability matrix. As a result, the property $\mathcal{O}_{ik} \cdot \mathbf{N} = \mathbf{0}$ does not hold.

It can be shown that the nullspace of $\mathcal{O}$ (i.e., the unobservable subspace) is now of dimension only three (see Appendix A.4 for the detailed derivations). This nullspace is spanned by the first three column vectors (the first block column) of $\mathbf{N}$ in (2.54), which means that the global yaw *erroneously appears* to be observable. As a result, the MSCKF underestimates the uncertainty of the yaw estimates. Since the yaw uncertainty affects the uncertainty of other state variables (e.g., the position), eventually the uncertainty of all states will be underestimated, and the estimator will be inconsistent. This helps to explain the results observed in the NEES plot of Fig. 2.1.

It is important to point out that the incorrect observability properties of the linearized system model do not affect only the MSCKF algorithm. In Appendix A.5

the observability matrix of the linearized model of EKF-SLAM is shown. This matrix has a nullspace of dimension three as well, similarly to the MSCKF. In fact, for the EKF-SLAM methods, the "disturbance" term appearing in the observability matrix contains additional terms due to the corrections in the feature position estimates. Such terms do not appear in the MSCKF, which uses only one estimate for each feature in all Jacobians.

2.7 MSCKF 2.0

In the preceding section, we proved that the linearized system model employed by the MSCKF has incorrect observability properties, which cause the filter to become inconsistent. In this section, we propose an extension of the basic MSCKF algorithm, termed MSCKF 2.0, which is able to address the inconsistency problem by ensuring the correct observability properties of the filter's linearized model.

To motivate the main idea of the MSCKF 2.0 algorithm, we note that, as shown in the previous section, in the original MSCKF *different estimates of the same states* are used for computing Jacobians, and this leads to an infusion of "fictitious" information about the yaw. Specifically, the use of different estimates for the IMU position and velocity result in nonzero values for the disturbance terms $\Delta\mathbf{\Gamma}_{ik}$ (see (2.58)), which change the dimension of the nullspace of the observability matrix. To remove these $\Delta\mathbf{\Gamma}_{ik}$ terms, a simple solution is to ensure that only one estimate of each IMU position and velocity is used in all Jacobians involving it.

A causal approach to achieve this is to always use the *first* available estimate for each state. Specifically, we compute the filter Jacobians as follows:

- The IMU error-state transition matrix at time-step k is computed as:

$$\mathbf{\Phi}^{\star}_{I_k}(\hat{\mathbf{x}}_{E_{k+1|k}}, \hat{\mathbf{x}}_{E_{k|k-1}}) \tag{2.61}$$

- The measurement Jacobians are computed as:

$$\mathbf{H}^{\star}_{\mathbf{f}_{ik}} = \mathbf{J}_{ik}(\hat{\mathbf{x}}_{E_{k|\alpha_i}}, {}^{G}\hat{\mathbf{p}}_{f_i})\; {}^{C}_{I}\mathbf{R}\; \hat{\mathbf{R}}_{k|\alpha_i} \tag{2.62}$$

$$\mathbf{H}^{\star}_{I_{ik}} = \mathbf{H}^{\star}_{\mathbf{f}_{ik}} \begin{bmatrix} \lfloor({}^{G}\hat{\mathbf{p}}_{f_i} - {}^{G}\hat{\mathbf{p}}_{I_{k|k-1}})\times\rfloor & -\mathbf{I}_3 & \mathbf{0}_3 \end{bmatrix} \tag{2.63}$$

As a result of the above changes, only the "propagated" estimates for the position and velocity (e.g., ${}^{G}\hat{\mathbf{p}}_{I_{k|k-1}}$ and ${}^{G}\hat{\mathbf{v}}_{I_{k|k-1}}$) are used in computing Jacobians. It is easy to show that with this change, the observability matrix regains the correct nullspace dimension, and thus the infusion of "fictitious" information for the yaw is avoided. We stress that we allow the state estimates to be updated normally; the only change we make to the MSCKF equations is that we do not use the updated estimates of the position and velocity in computing Jacobians. This change, which incurs *no* additional computational cost, substantially improves performance, as shown in the simulation and experimental results presented in Sections 2.8 and 2.9.

The idea of using the first estimates of all states to ensure the correct observability properties of the linearized system model can also be employed for EKF-SLAM VIO. In this case, in addition to the IMU position and velocity estimates, we must also use the same (first) estimate of each feature when computing all Jacobians involving it. As shown in Section 2.8, the resulting EKF-SLAM algo-

Table 2.2: Average RMSE and NEES results for all the EKF-based VIO algorithms tested in the simulations.

	$\kappa = 2$			$\kappa = 4$			$\kappa = 6$		
Algorithm	$\delta\mathbf{p}_R$ (m)	$\delta\boldsymbol{\theta}_R$ (°)	NEES	$\delta\mathbf{p}_R$ (m)	$\delta\boldsymbol{\theta}_R$ (°)	NEES	$\delta\mathbf{p}_R$ (m)	$\delta\boldsymbol{\theta}_R$ (°)	NEES
XYZ	N/A	N/A	N/A	78.447	5.609	$4.9\cdot10^3$	53.469	3.974	$1.3\cdot10^3$
IDP	69.502	3.731	$2.2\cdot10^3$	26.193	1.916	$2.7\cdot10^2$	22.878	1.803	$1.7\cdot10^2$
AHP	67.061	4.795	$2.7\cdot10^2$	52.355	4.531	$1.3\cdot10^2$	36.858	3.129	48.236
m-XYZ	60.564	3.160	$1.6\cdot10^2$	19.297	1.512	9.185	12.477	1.238	7.385
m-IDP	40.912	2.057	57.346	18.144	1.400	8.600	15.498	1.211	7.156
m-AHP	38.288	2.311	38.932	18.010	1.385	8.357	15.494	1.205	7.160
				$\delta\mathbf{p}_R$ (m)	$\delta\boldsymbol{\theta}_R$ (°)	NEES			
MSCKF				14.401	1.102	7.741			
MSCKF 2.0				12.840	1.008	5.890			
"Ideal" MSCKF				12.720	1.001	5.816			

rithms outperform the standard ones, yet cannot reach the accuracy or consistency of the MSCKF.

2.8 Simulation Results

In this section we present simulation results that illustrate the analysis of the preceding sections, and demonstrate the performance of the MSCKF 2.0 algorithm compared to alternatives. All the simulation data is generated based on real-world datasets, as explained in Appendix A.6, and all the results reported are averages over 50 Monte-Carlo trials.

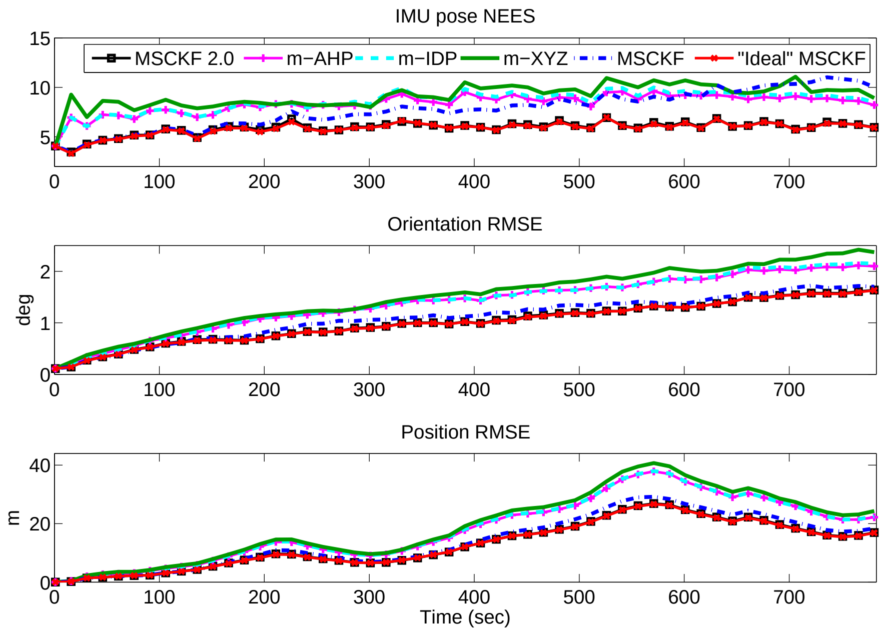


Figure 2.2: Simulation results: The average position and orientation RMSE over 50 Monte-Carlo trials. The algorithms compared are the MSCKF (blue dash-dotted line), the "ideal" MSCKF (red line with x-marks), the MSCKF 2.0 (black line with squares), the m-XYZ SLAM (green solid line), the m-IDP SLAM (cyan dashed line), and the m-AHP SLAM (magenta line with "plus"-marks).

2.8.1 Comparison to EKF-based SLAM

We first compare the performance of the MSCKF and MSCKF 2.0 algorithms to EKF-SLAM based methods for VIO. In the results presented in Section 2.3.4, it was shown that the original MSCKF algorithm outperforms the standard EKF-SLAM algorithms using either the XYZ, IDP, or AHP feature parameterizations. We thus here focus on comparing the performance of the MSCKF-based algorithms to that of the "modified" EKF-SLAM versions, where the first estimate of each state is used in computing Jacobians to ensure the correct rank of the linearized system's observability matrix. These modified algorithms are identified as m-XYZ, m-IDP, and m-AHP. Additionally, in this simulation we include an "ideal" MSCKF algorithm, in which the true IMU states and feature positions are used for computing all the filter Jacobians. This algorithm (which is only realizable in a simulation environment), can serve as a benchmark of performance for the MSCKF-based methods. For the results presented here, exactly the same simulation data as in Section 2.3.4 are used, to facilitate comparison.

Fig. 2.2 shows the average IMU pose NEES as well as the IMU position and orientation RMSE over time, for the three MSCKF based methods, as well as for the three EKF-SLAM methods (with $\kappa = 4$ for the SLAM methods). Moreover, Table 2.2 provides the numerical values for the NEES and RMSE for all the algorithms (this table includes the results of Section 2.3.4 for easier comparison).

These results show that all the "modified" algorithms, which use the first estimates of each state in Jacobian computation, outperform their counterparts that use the standard approach for Jacobian computation. Not only are these algorithms more consistent (i.e., they have smaller NEES), but also more accurate (i.e., smaller RMSE). These results show that enforcing the correct observability properties of the linearized system is crucially important for the performance of *all* EKF-based VIO methods, and validate the analysis of Section 2.6. Despite the improvement that the modified EKF-SLAM algorithms offer, however, they are all less accurate than all the MSCKF-based methods. This shows the advantages of the MSCKF approach to processing the feature measurements, which copes better with nonlinearities by not making Gaussian assumptions about the feature pdfs. Additionally, we can observe that the performance of the MSCKF 2.0 algorithm is almost indistinguishable from that of the "ideal" MSCKF, both in terms of accuracy and consistency. This indicates that, as long as the correct observability properties are ensured, using slightly less accurate linearization points in computing the Jacobians does not significantly degrade the estimation performance. Based on the simulation results (and given that the "ideal" MSCKF is not realizable), we can conclude that the MSCKF 2.0 is the preferred VIO method out of all the EKF-based approaches considered here.

2.8.2 Comparison to iterative-optimization methods

We next compare the performance of the MSCKF-based algorithms to that of an iterative-minimization based method. Specifically, we use an information-form fixed-lag smoother (FLS), based on [100] for comparison. This is a sliding-window bundle adjustment method that marginalizes older states to maintain a constant computational cost. The FLS is essentially the counterpart of the

MSCKF within the class of iterative-minimization methods, which allows for a meaningful comparison. In our implementation, the sliding window contains a number of IMU poses corresponding to the times images were recorded, as well as the features observed in these poses. The IMU measurements are used to provide the "process-model" information between the poses of the window, while the feature observations provide the "sensor-model" information (see Section 2.1 in [100]). Every time a new image is recorded, Gauss-Newton minimization is employed to update the state estimates in the sliding window, and subsequently the oldest pose, and features that are no longer observed, are marginalized out. All the methods tested (MSCKF, "ideal" MSCKF, MSCKF 2.0, and FLS) use a sliding window of the same length.

For these tests we employ a much longer dataset as our basis for generating simulated data. Specifically, we use the Cheddar Gorge dataset [101], which involves a 29-km-long trajectory, collected in 56 minutes of driving. For this dataset an Xsens IMU provided measurements at 100 Hz, and images were available at 20 Hz. In each image, 240 features were tracked on average, and the average track length was 4.1 frames (note that this is due to the fact that a very large percentage of features are tracked for short periods in this dataset, which involves a fast-moving vehicle.).

Before examining the averaged results of all the Monte-Carlo trials, it is interesting to examine the results of estimation for the rotation about gravity (the yaw) in a single trial. Fig. 2.3 shows the estimation errors in the yaw for the four algorithms, as well as the $\pm 3\sigma$ envelopes computed using the reported covariance of each method (these are the reported 99.7% confidence regions). The most important observation here is that the reported standard deviation for both the MSCKF and the FLS fluctuates about a constant value, *as if* the yaw was observable. By contrast, in the “ideal” MSCKF and the MSCKF 2.0 algorithms, the standard deviation of the yaw increases over time, as theoretically expected, given that the yaw is unobservable. This figure clearly demonstrates the importance of the observability properties of the linearized system: when these do not match the properties of the underlying nonlinear system, the estimation results (e.g., reported uncertainty) exhibit fundamentally incorrect characteristics. We note here that the FLS also suffers from the same inconsistency problem, even though it employs iterative re-linearization, as shown in [21].

The three subplots in Fig. 2.4 show the average NEES for the IMU pose, as well as the RMSE for the IMU orientation and position, averaged over 50 Monte-Carlo trials. Table 2.3 lists the average NEES and RMSE values for the four algorithms. First, we note that the performance of the MSCKF 2.0 is similar to that of the “ideal” MSCKF, and that both algorithms clearly outperform the standard MSCKF. These results once again show that by enforcing the correct observability properties, the filter's performance can be significantly improved. Additionally, in this simulation environment, we see that the performance difference between the standard MSCKF and the MSCKF 2.0 is more pronounced than before. This is due to the fact that the Cheddar Gorge dataset is significantly longer (both in trajectory length and duration). As a result, more “spurious” information about the yaw is accumulated, due to the incorrect observability properties of the filter's

Table 2.3: Average NEES and RMSE for Fig. 2.4.

	$\delta \mathbf{p}_R$ (m)	$\delta \boldsymbol{\theta}_R$ (°)	NEES
FLS	133.4	2.83	50.97
MSCKF	146.2	3.40	51.72
MSCKF 2.0	97.7	2.21	6.53
"Ideal" MSCKF	100.2	2.35	6.45

linearized model. In turn, this causes a larger degradation in the estimates of the standard MSCKF.

More importantly though, we see that the MSCKF 2.0 (as well as the "ideal" MSCKF) attains substantially better accuracy and consistency even than the iterative FLS method. This occurs even though the latter uses approximately 5 times more computation time. The performance difference between the MSCKF 2.0 and the FLS demonstrates that (at least in the case examined here) having a linearized system model with appropriate observability properties is more important than using re-linearization to better approximate the nonlinear measurement models.

2.9 Real-world Experiment

We next describe the results of a real-world experiment, during which an IMU/camera platform was mounted on top of a car and driven on the streets

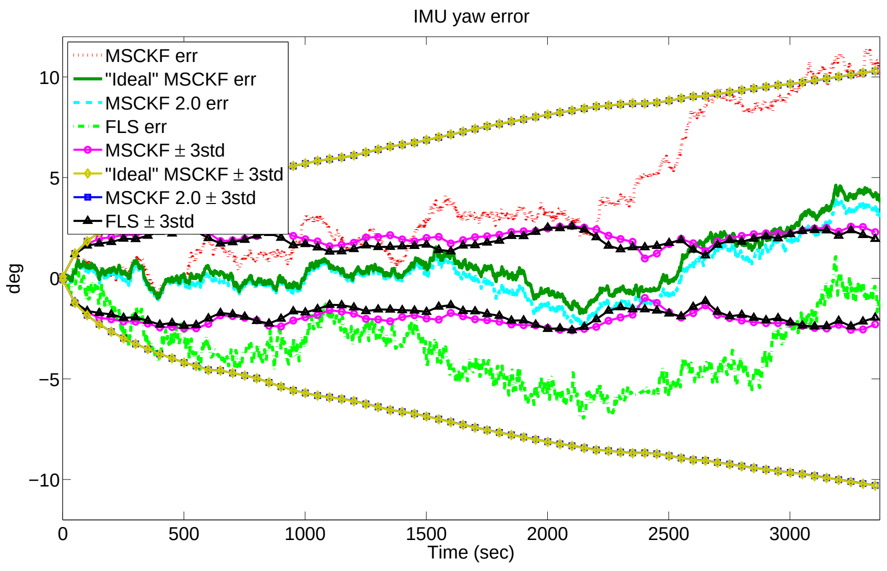


Figure 2.3: IMU yaw errors and $\pm 3\sigma$ bounds in one representative trial of the Cheddar Gorge simulation. The yaw error for the MSCKF (dotted line – red), the "ideal" MSCKF (solid line – dark green), the MSCKF 2.0 (dashed line – cyan), and the FLS (dashdotted line – light green). The $\pm 3\sigma$ bounds for the MSCKF (circles – magenta), the "ideal" MSCKF (diamonds – yellow), the MSCKF 2.0 (squares – blue), and the FLS (triangles – black).

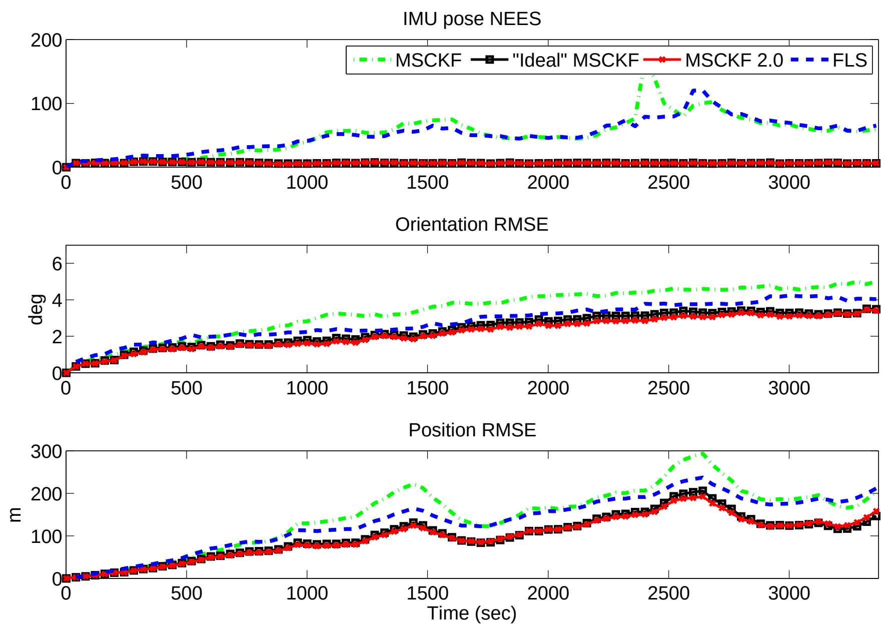


Figure 2.4: Average NEES and RMSE over 50 Monte Carlo trials of the Cheddar Gorge simulation. The dotted green line corresponds to the MSCKF, the black line with squares to the "ideal" MSCKF, the red line with x-marks to the MSCKF 2.0, and the blue dashdotted line to the FLS.

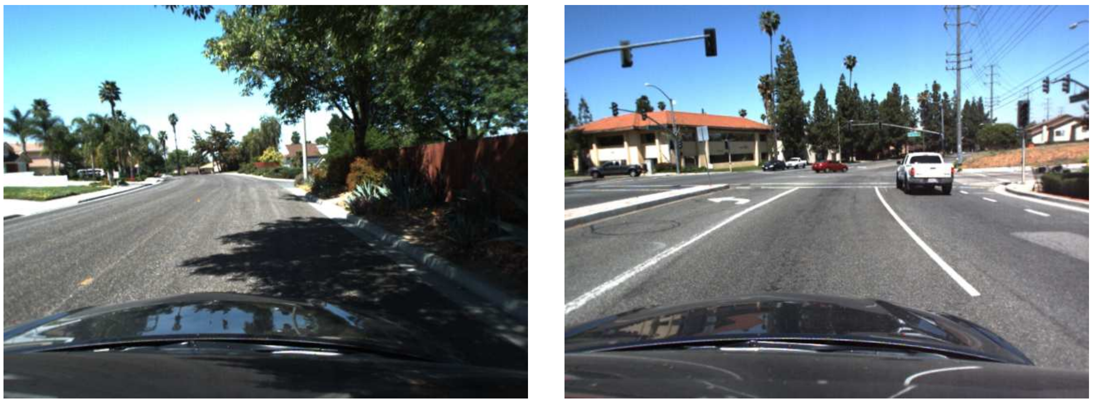


Figure 2.5: Sample images recorded during the experiment.

of Riverside, CA. The sensors consisted of an Xsens MTi-G unit, and a PointGrey Bumblebee2 stereo pair (only a single camera's images are used). The IMU provided measurements at 100 Hz, while the camera images were stored at 20 Hz. For position ground truth we used a GPS-INS estimate of the trajectory. For image processing, Harris feature points were extracted, and matching was carried out by normalized cross-correlation. On average, approximately 290 features were tracked per image. The experiment lasted about 37 minutes, during which the vehicle drove approximately 21.5 km. Some sample images from the experiment are shown in Fig. 2.5.

Fig. 2.6 shows the ground truth trajectory on a map of the area where the vehicle was driven, as well as the estimates computed by three algorithms: the MSCKF, the FLS, and the proposed MSCKF 2.0. These are the three most accurate estimators tested, and we only present their results for clarity. Fig. 2.7 plots the estimation error as well as the reported standard deviation of the yaw and the x-y position for the three algorithms. Similarly to what was observed in Fig. 2.3,

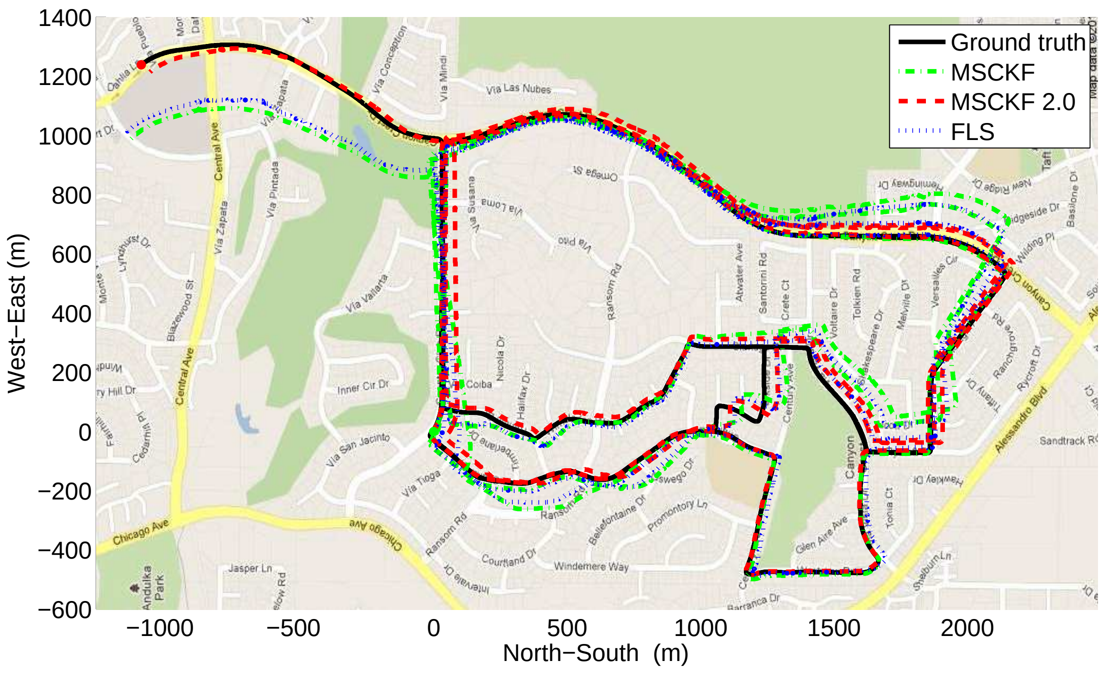


Figure 2.6: Trajectory estimates plotted on a map of the Canyon Crest area in Riverside, CA. The initial vehicle position is denoted by a green circle, and the end position by a red circle. The black solid line corresponds to the ground truth, the green dash-dotted to the MSCKF, the red dashed line to the MSCKF 2.0, and the blue dotted line to the FLS.

we see that the MSCKF 2.0 (plots on the left) offers a better characterization of the actual uncertainty. By contrast, the uncertainties of both the yaw and the IMU position are underestimated by the MSCKF and the FLS (plots on the right). The estimation errors for these algorithms are also significantly larger than those of the MSCKF 2.0. Specifically, the largest position error for the MSCKF 2.0 algorithm is approximately 65 m, which corresponds to only 0.30% of the travelled distance. By contrast, the trajectory estimates reported by the MSCKF and the

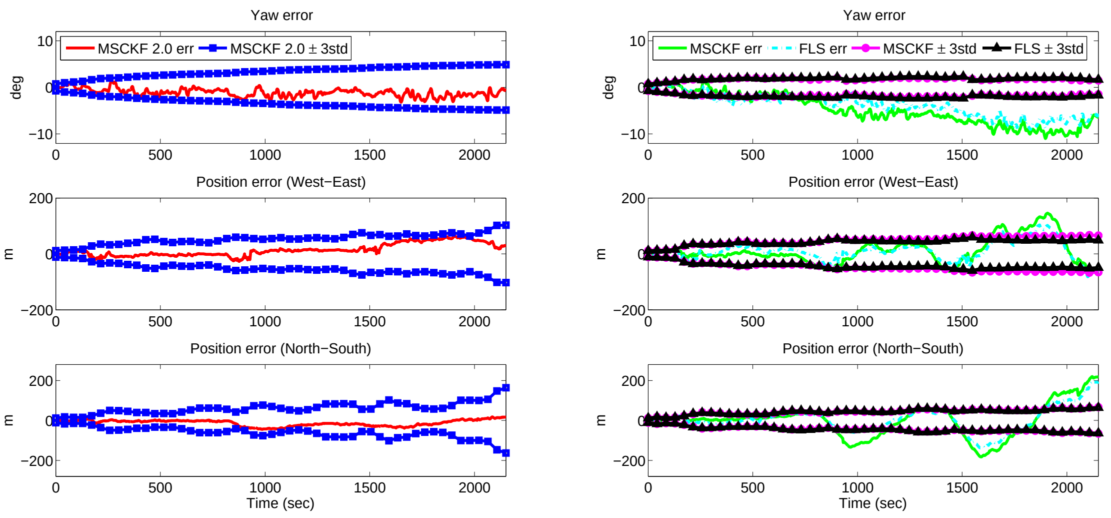


Figure 2.7: Estimation errors for the three approaches. The left plots are the results for the MSCKF 2.0, and the right plots for the MSCKF and FLS.

FLS are less accurate, with largest position errors of about 230 m and 202 m, respectively.

As a final remark, we note that in this experiment, the average processing time per update of the MSCKF 2.0 (including image processing and estimation) is 15 msec measured on a Core i7 processor at 2.66 GHz, with a single-threaded C++ implementation. Since the image period is 50 msec, the algorithm's performance is comfortably within the requirements for real-time operation.

2.10 Conclusion

In this chapter, we have presented a detailed analysis of the properties and performance of different EKF-based VIO algorithms. We show that the MSCKF algorithm attains better accuracy and consistency than EKF-based SLAM algorithms, due to its less strict probabilistic assumptions and delayed linearization. In addition, we performed a rigorous study of the consistency properties of EKF-based VIO algorithms, and showed that the filters' linearized system models have incorrect observability properties, which result in inconsistency. To address this problem, we developed the MSCKF 2.0 algorithm, which uses a novel closed-form expression for the IMU error-state transition matrix and fixed linearization states to ensure the appropriate observability properties. Extensive Monte-Carlo simulations and real-world experimental testing provide strong validation of our theoretical analysis, and demonstrate that the proposed MSCKF 2.0 algorithm is capable of performing long-term, high-precision, consistent VIO in real time. In fact, the MSCKF 2.0 algorithm is shown to outperform even an iterative-minimization based fixed-lag smoother, an algorithm with substantially higher computational requirements.

Chapter 3

Hybrid MSCKF/SLAM Visual-Inertial Odometry

3.1 Introduction

In the preceding chapter, we presented a computationally efficient VIO algorithm, the MSCKF 2.0, and showed that it is able to accurately track the motion of a moving platform. It was shown that the MSCKF 2.0 can perform real-time estimation on high-end processors, owing to the fact that its computational complexity is *linear* in the number of tracked features, which leads to efficient computation when many features are present. However, this does not guarantee that the MSCKF 2.0 will make *optimal* use of the available resources. It is possible that a different estimator may be able to process the same measurements and yield similar estimation performance, at a lower computational cost. This potential suboptimal utilization of CPU resources is clearly undesirable.

To address this problem, in this chapter, we propose a new paradigm for the design of VIO estimators, by combining the MSCKF 2.0 with an algorithm with complementary computational characteristics. This makes it possible to decide, in real time, to process different measurements (e.g., different features) by a different algorithm. Since the optimal choice for each measurement will depend on the characteristics of the sensor data, in the proposed framework we employ statistical learning to learn these characteristics. Gathering statistical information allows us to compute the expected cost of any strategy for allocating measurements to algorithms. To identify the optimal strategy we solve, in real time, an optimization problem whose objective function is the expected computation time.

Specifically, in this chapter, we design a hybrid EKF, which integrates the MSCKF 2.0 with the modified EKF-SLAM (m-SLAM) algorithm. These two estimators process the same measurement information in different ways, and have complementary computational characteristics. We demonstrate that the optimal choice of algorithm to process each individual feature depends on the distribution of the feature track lengths of all features. This distribution is not known in advance (it depends on the environment, the camera motion, as well as the feature tracker used), and therefore we learn it from the image sequence. Using this information, the optimal strategy for processing the feature measurements can be computed by solving a one-variable optimization problem, as shown in Section 3.5. Our results demonstrate that the hybrid algorithm outperforms each individual method (m-SLAM and MSCKF 2.0) by a wide margin.

3.2 Related Work

We begin by discussing existing approaches that seek to optimize the computational efficiency of localization methods. To the best of our knowledge, no prior work exists that employs learning of the feature tracks' characteristics to *explicitly* optimize the computational cost of estimation. Past work has focused primarily on EKF-SLAM, and can broadly be categorized as:

Exact reformulations of the SLAM equations Typical methods decompose the computations into smaller parts, and selectively carry out the only necessary computations at each time instant (see e.g., [32, 46, 90, 117]). Typically, in these methods the currently visible features are updated normally at every time step, while the remaining ones are updated only "on demand" or when re-visited. In the case of visual-inertial odometry, however, where the state vector contains only the actively tracked features and no loop-closure is considered, these methods are not applicable.

Approximations of the SLAM equations On the other hand, several methods exist that employ approximations to the SLAM equations (e.g., [44, 86, 108] and references therein), to reduce the required computations. In contrast to these methods, which trade-off information for computational efficiency, our proposed approach involves no information loss, and no approximations, other than the inaccuracies due to the EKF's linearization.

Feature selection methods A different family of approaches seek to reduce the computational cost of localization by processing only the most valuable, in terms of uncertainty reduction, measurements (e.g., [19, 104]). Only processing a small subset of carefully selected measurements can often result in high accuracy. However, since the remaining measurements are simply discarded, loss of information inevitably occurs. In our proposed approach, *all* the available measurements are processed, and no localization information is lost.

3.3 Computational Complexity of the MSCKF

In this section, we analyze the computational complexity of the MSCKF algorithm in detail, to demonstrate its advantages and shortcomings. This analysis guides our selection of an algorithm with complementary characteristics (the EKF-SLAM algorithm), for integration with the MSCKF in the hybrid EKF formulation.

As discussed in the previous chapter, the way in which the feature measurements are processed in the MSCKF is optimal, in the sense that no approximations are used, except for the EKF's linearization [62]. This is true, however, *only* if the sliding window of states, m (see (2.12)), contains at least as many states as the longest feature track. If it does not, then the measurements that occurred more than m timesteps in the past cannot be processed. Therefore, to use all the available feature information, the MSCKF must maintain a window of states long enough to include the longest feature tracks.

We now examine the dependence of the computational requirements of the MSCKF on the number of features and the lengths of the feature tracks. The computation time of the MSCKF is dominated by the following operations:

1) The computation of the residual and Jacobian matrix in (2.19), for each feature. If n features are processed, with feature track lengths $\ell_i, i = 1 \dots n$, this requires $\mathcal{O}(\sum_{i=1}^{n} \ell_i^2)$ operations.

2) The Mahalanobis test, requiring $\mathcal{O}(\sum_{i=1}^{n} \ell_i^3)$ operations.

3) The computation of the residual and the Jacobian matrix in (2.22), which, by exploiting the structure in $\mathbf{H}^o$, can be implemented in approximately $\mathcal{O}(\sum_{i=1}^{n} \ell_i^3)$ operations.

4) The computation of the Kalman gain and the update of the covariance matrix, which require $\mathcal{O}(r(15+6m)^2 + r^2(15+6m) + r^3)$ operations. Here $15+6m$ is the size of the state covariance matrix, and r is the number of rows in $\mathbf{H}^r$. It can be shown that, in general, r (which equals the number of independent constraints for the camera poses) is given by [64]:

$$r = 2(\ell_{(1)} + \ell_{(2)} + \ell_{(3)}) - 7 \tag{3.1}$$

where $\ell_{(1)}$, $\ell_{(2)}$, and $\ell_{(3)}$ are the three longest feature tracks.

From the above we see that, while the computational cost of the MSCKF is linear in the number of features, it is at least quadratic in the size of the sliding window, m. In fact, if the size of the sliding window is chosen to be equal to the longest feature tracks, then r is also $\mathcal{O}(m)$, and the overall complexity becomes *cubic* in m. This demonstrates a shortcoming of the MSCKF: to preserve all measurement information, the complexity of the algorithm scales cubically as a function of the *longest* feature track length.

In real-world datasets the distribution of the feature-track lengths is non-uniform, with many features tracked for short durations, and very few stabler features tracked over longer periods. For instance, Fig. 3.1 shows the distribution of the feature track lengths in two parts of a real-world dataset [101]. It can be seen that, even though the longest feature tracks reach 40 images, the vast majority of features is tracked for a small number of frames (the percentage of features tracked in five or less images is 88% and 69% in the two cases shown, respectively). To be able to process the small percentage of features with long track lengths, the MSCKF must maintain a long sliding window, which is computationally inefficient.

As discussed in the previous chapter, in addition to the MSCKF, another type of EKF-based estimator that utilizes the measurement information *optimally*, except for the EKF's linearization, is EKF-SLAM. In terms of computational complexity, the EKF-SLAM requires *cubic* computation in the number of features at each time-step, but the cost scales linearly with the features' track lengths. Therefore, if many features are tracked, each in a small number of frames, the MSCKF approach is more efficient, but if few features are tracked over long image sequences, EKF-SLAM would result in lower computational cost. Clearly, EKF-SLAM and the MSCKF algorithm are complementary, with each being superior in different circumstances. This motivates us to integrate both algorithms in a single, hybrid filter, as described next.

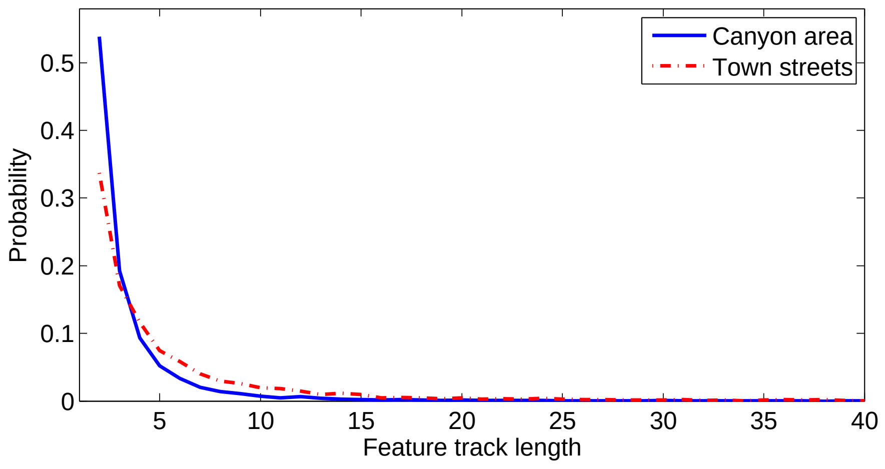


Figure 3.1: Distribution of the feature track lengths in two parts of the Cheddar Gorge dataset [101].

3.4 The hybrid MSCKF/SLAM Algorithm

3.4.1 State formulation

The state vector of the hybrid filter at time-step k is defined as:

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_{E_k}^T & \mathbf{x}_{I_{k-1}}^T & \cdots & \mathbf{x}_{I_{k-m}}^T & \mathbf{f}_1^T & \cdots & \mathbf{f}_{s_k}^T \end{bmatrix}^T \tag{3.2}$$

where $\mathbf{x}_{E_k}$ represents the evolving IMU state (see (2.1)), $\mathbf{x}_{I_i}$, $i = k-1, \ldots, k-m$, are the IMU poses (see (2.13)) corresponding to the previous m images, and $\mathbf{f}_j$, $j = 1, \ldots, s_k$, are the SLAM features, whose parametrization is described in Appendix B.1.

This formulation provides us with the ability to choose whether a feature will be processed using the MSCKF approach, or whether it will be included in the state vector and processed as in EKF-SLAM. By analyzing in detail the computational requirements of each EKF update, it can be shown that, when many features are present, there is nothing to be gained by initializing in the state vector any feature observed fewer than m times [64]. Thus, the optimal (in terms of computational requirements) strategy for using the features turns out to be a simple one: if feature i's track is lost after fewer than m frames (i.e., $\ell_i < m$), then the feature is processed using the MSCKF equations. On the other hand, if a feature is still actively being tracked after m images, it is initialized into the state vector, and used for SLAM. In the remainder of this dissertation, we term the features processed via MSCKF equations the “MSCKF features”, and the ones utilized by EKF-SLAM the “SLAM features”. In what follows we describe the detailed formulation of the EKF update for the hybrid estimator.

3.4.2 EKF update

To formulate EKF update equations, the MSCKF features will be processed via the nullspace projection (2.19), Mahalanobis testing (2.20), and QR factorization (2.22). On the other hand, for each SLAM feature that is already included in the EKF's state vector, we formulate the linearized measurement equation (2.11), and perform outlier rejection. By combining the residuals of the MSCKF and SLAM features, we obtain

$$\tilde{\mathbf{z}}_k = \begin{bmatrix} \tilde{\mathbf{z}}_k^r \\ \tilde{\mathbf{z}}_k^s \end{bmatrix} \simeq \begin{bmatrix} \mathbf{H}_k^r & \mathbf{0} \\ \mathbf{H}_k^{\mathbf{x}} & \mathbf{H}_k^{\mathbf{f}} \end{bmatrix} \tilde{\mathbf{x}}_{k|k-1} + \mathbf{n}_k \tag{3.3}$$

$$= \mathbf{H}_k \tilde{\mathbf{x}}_{k|k-1} + \mathbf{n}_k \tag{3.4}$$

where $\tilde{\mathbf{z}}_k^r$ and $\mathbf{H}_k^r$ are the residual vector and Jacobian matrix of the MSCKF features, computed via the QR factorization (2.22), $\tilde{\mathbf{z}}_k^s$ is the residual vector corresponding to the SLAM features, $\mathbf{H}_k^{\mathbf{x}}$ is the Jacobian matrix of the SLAM features with respect to the evolving state and IMU poses, $\mathbf{H}_k^{\mathbf{f}}$ is the Jacobian with respect to the SLAM features, and $\mathbf{n}_k$ is a zero-mean Gaussian noise vector, whose covariance matrix is $\sigma^2\mathbf{I}$. We note that in computing all the Jacobians described above, the first estimate of each position state is used, as described in the MSCKF 2.0 and m-SLAM algorithms.

Once the residual $\tilde{\mathbf{z}}_k$ and the Jacobian matrix $\mathbf{H}_k$ have been computed, we compute the state correction and update the covariance matrix via the standard EKF equations:

$$\boldsymbol{\Delta}\mathbf{x}_k = \mathbf{K}_k\tilde{\mathbf{z}}_k \tag{3.5}$$

$$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - \mathbf{K}_k\mathbf{S}_k\mathbf{K}_k^T \tag{3.6}$$

$$\mathbf{S}_k = \mathbf{H}_k\mathbf{P}_{k|k-1}(\mathbf{H}_k)^T + \sigma^2\mathbf{I} \tag{3.7}$$

$$\mathbf{K}_k = \mathbf{P}_{k|k-1}(\mathbf{H}_k)^T\mathbf{S}_k^{-1} \tag{3.8}$$

3.4.3 Feature initialization

As discussed in Section 3.4.1, features that are still being tracked after m images are initialized as SLAM features in the EKF state vector. As shown in [64] the initialization of all such features can be performed after the EKF update has been computed. To describe the initialization equations, let us consider a MSCKF feature, $\mathbf{f}_{new}$, that has been observed in all m frames of the sliding window. The first step in the initialization process is to use this feature as part of the normal EKF update at time-step k, via the MSCKF equations. Once the EKF update has been performed, we must calculate the feature's posterior estimate, $\hat{\mathbf{f}}_{new,k|k}$, marginal covariance matrix, $\mathbf{P}_{ff,k|k}$, and the cross-correlation between the EKF state and the new feature, $\mathbf{P}_{fx,k|k}$, in order to initialize it into the state vector.

Specifically, the posterior feature estimate is calculated by [64]:

$$\hat{\mathbf{f}}_{new,k|k} = \hat{\mathbf{f}}_{new} - \mathbf{H}_{f_{new},k}^{-1}\mathbf{H}_{x_{new},k}\Delta\mathbf{x}_k + \mathbf{H}_{f_{new},k}^{-1}\tilde{\mathbf{z}}_{new} \tag{3.9}$$

where $\hat{\mathbf{f}}_{new}$ is the feature estimate computed by least-squares minimization as part of the MSCKF update, $\tilde{\mathbf{z}}_{new}$ is the measurement residual vector computed using $\hat{\mathbf{f}}_{new}$ and $\hat{\mathbf{x}}_{k|k-1}$, $\mathbf{H}_{f_{new},k}$ and $\mathbf{H}_{x_{new},k}$ are the measurement Jacobian matrices with respect to the feature and the EKF state, and the state correction vector $\Delta\mathbf{x}_k$ is computed via (3.5). Additionally, the feature's marginal covariance and cross-correlation terms are:

$$\mathbf{P}_{fx,k|k} = -\mathbf{H}_{f_{new},k}^{-1}\mathbf{H}_{x_{new},k}\mathbf{P}_{k|k} \tag{3.10}$$

$$\mathbf{P}_{ff,k|k} = \left(\mathbf{H}_{f_{new},k}^{-1}\mathbf{H}_{x_{new},k}\right)\mathbf{P}_{k|k}\left(\mathbf{H}_{f_{new},k}^{-1}\mathbf{H}_{x_{new},k}\right)^{-1} + \mathbf{H}_{f_{new},k}^{-1}\mathbf{H}_{f_{new},k}^{-T} \tag{3.11}$$

where $\mathbf{P}_{k|k}$ is the posterior covariance matrix of the original state computed in (3.6).

Once the vector $\hat{\mathbf{f}}_{new,k|k}$ as well as the matrices $\mathbf{P}_{fx,k|k}$ and $\mathbf{P}_{ff,k|k}$ are calculated, we augment the EKF state estimate and covariance matrix as follows:

$$\hat{\mathbf{x}}_{k|k} \leftarrow \begin{bmatrix} \hat{\mathbf{x}}_{k|k} \\ \hat{\mathbf{f}}_{new,k|k} \end{bmatrix}, \quad \mathbf{P}_{k|k} \leftarrow \begin{bmatrix} \mathbf{P}_{k|k} & \mathbf{P}_{fx,k|k}^T \\ \mathbf{P}_{fx,k|k} & \mathbf{P}_{ff,k|k} \end{bmatrix} \tag{3.12}$$

If more than one feature need to be initialized, the above process is performed for each feature.

It is important to note that the EKF update and feature-initialization equations are *optimal* up to linearization, i.e., the hybrid filter would yield the MAP estimates if the system and measurement models were linear-Gaussian.

Once the EKF update and feature initialization are complete, the algorithm proceeds to remove unnecessary variables from the state vector and covariance matrix. These variables are (i) the IMU poses whose corresponding MSCKF features have been all processed, and (ii) the SLAM features that are no longer tracked. Due to the anchored inverse-depth parameterization that we employ for the SLAM features (see Appendix B.1), SLAM features have to be re-parameterized when their anchor poses are removed from the EKF state vector. The details of this process are given in Appendix B.2. The overall hybrid MSCKF/SLAM algorithm is described in Algorithm 2.

3.5 Optimizing the Performance of the Hybrid EKF

In this section we show how the size of the sliding window, m, can be selected so as to minimize the computational cost of the hybrid MSCKF/SLAM filter. As the results in Section 3.6 show, the choice of m has a profound effect on the time requirements of the algorithm. With a suitable choice, the hybrid method can significantly outperform each of its individual components.

Algorithm 2 Hybrid MSCKF/SLAM algorithm

Propagation: Propagate the state vector and covariance using IMU readings.

Update: Once camera measurements become available:

- Augment the state vector with the latest camera pose.
- For features to be processed in the MSCKF (feature tracks of length smaller than m), do the following
  - Calculate the residual and Jacobian for each feature via (2.19).
  - Perform the Mahalanobis gating test in (2.20).
  - Using all features that passed the gating test, form the residual vector and the Jacobian matrix in (2.22).
- For SLAM features, compute the residuals and measurement Jacobian matrices, and form the residual $\tilde{\mathbf{z}}_k$ and matrix $\mathbf{H}_k$ in (3.3).
- Update the state vector and covariance matrix, via (3.5)-(3.8).
- Initialize features tracked in all m images via (3.9)-(3.11).

State Management:

- Remove SLAM features that are no longer tracked, and change the anchor pose for SLAM features anchored at the oldest poses.
- Remove the oldest IMU pose from the state. If no feature is currently tracked for more than m_o poses (with $m_o < m-1$), remove the oldest $m - m_o$ poses.

3.5.1 Operation count for each update

By carefully analyzing the computations needed, we calculate the number of floating-point operations per update of the hybrid algorithm:

$$\begin{aligned} f_k = \alpha_1 \sum_{i=1}^{n} \ell_i^3 + \alpha_2 r m^2 + \alpha_3 r^2 m + \alpha_4 r m s_k \\ + \alpha_5 (r + 2s_k)^3 + \alpha_6 (r + 2s_k)^2 (6m + 15 + 3s_k) \\ + \alpha_7 (r + 2s_k)(6m + 15 + 3s_k)^2 + l.o.t \end{aligned} \tag{3.13}$$

where the α_i's are known constants, n is the number of features used in the MSCKF, r is defined in (3.1), and $l.o.t.$ stands for lower-order terms[1]. The terms shown above correspond to the computations for the MSCKF features (1st term), for the matrix $\mathbf{S}$ of the hybrid filter (2nd-4th terms), and for the EKF update (5th-7th terms). Note that (3.13) also models the probability of failure for the Mahalanobis test.

It is interesting to examine the properties of (3.13). First, we note that since r represents the number of constraints provided by the MSCKF features for the poses in the sliding window, it is bounded above by $6m - 7$: the total number of unknowns in the sliding window is $6m$, and the feature measurements cannot provide any information about the global pose or scale, which correspond to 7 degrees of freedom. If many features are available, we will have $r \approx 6m - 7$, and thus:

$$f_k \approx \alpha_1 \sum_{i=1}^{n} \ell_i^3 + \alpha_2' m^3 + \alpha_3' m^2 s_k + \alpha_4' m s_k^2 + \alpha_5' s_k^3$$

[1]The full expression for the operation count consists of tens of individual terms, whose inclusion would merely complicate the presentation, and not add much insight. Note however, that in the optimization described Section 3.5.2, the complete expression is used.

This approximate expression makes it possible to gain intuition about the behavior of the computational cost of the hybrid method: as the size of the sliding window, m, increases, more features will be processed by the MSCKF, and fewer features will be included in the state vector of the filter. Thus, as m increases, the term $\sum_{i=1}^{n} \ell_i^3$ will also increase, but s_k will decrease rapidly. These two opposing trends result in the performance curves shown in Section 3.6.1 (e.g., Fig. 3.3).

3.5.2 Minimizing the average operation count

We now turn our attention to determining the optimal value of m, in order to minimize the runtime of the hybrid EKF estimator. Equation (3.13) provides us with the operation count of one filter update, so at first glance, it may appear that one needs to minimize this equation with respect to m, at each time instant. However, that would be an ill-posed problem. To see why, consider the case where, at time step k, the sliding window has length $m = 20$, and ten features exist that have been continuously tracked in 20 images. At this time, we have the option of either increasing the size of the sliding window, or including the ten features in the state vector. Which is best depends on the *future* behavior of the feature tracks: if these features end up being tracked for a very large number of frames ($\gg 20$), then it would be preferable to include the features in the state vector. If, on the other hand, the features end up being tracked for only 21 frames, it would be preferable to increase m by one.

Clearly, it is impossible to obtain future information about any particular feature track. We can however, collect statistical information about the properties of the feature tracks, and use this information to minimize the *expected* cost of the EKF updates. This is precisely the approach we implement. Specifically, during the filter's operation, we collect statistics to learn the probability mass function (pmf) of the feature track lengths, $p(\ell_i)$, the probability of failure of the Mahalanobis gating test, as well as the pmf of the number of features tracked in the images. Using the learned pmfs, we compute the average number of operations needed for each EKF update, $\bar{f}(m)$, by direct application of the definition of the expected value of a function of random variables.

The value of m that yields the minimum $\bar{f}(m)$ can be found by exhaustive search among all possible values. However, we have found that the cost curve in practical cases is quasiconvex, which means that it has a unique minimum (see Fig. 3.2). Therefore, to reduce the time needed for the optimization, we perform the optimization by local search starting from a known good initial guess (e.g., the last computed value of m). Since the statistical properties of the feature tracks can change over time (see Fig. 3.1), we perform the learning of the pmfs as well as the selection of the optimal m in consecutive time windows spanning a few seconds (15 sec in our implementation).

It is worth noting that in modern computers the use of flop counts to model computational cost is not always suitable, as performance is affected by several factors including vectorization, cache access patterns, data locality, etc. However, we have experimentally verified that in the algorithms considered here, and for our implementation, the computed flop counts closely follow the observed runtimes. Specifically, Fig. 3.2 shows the actual runtime of the hybrid filter, as well as the

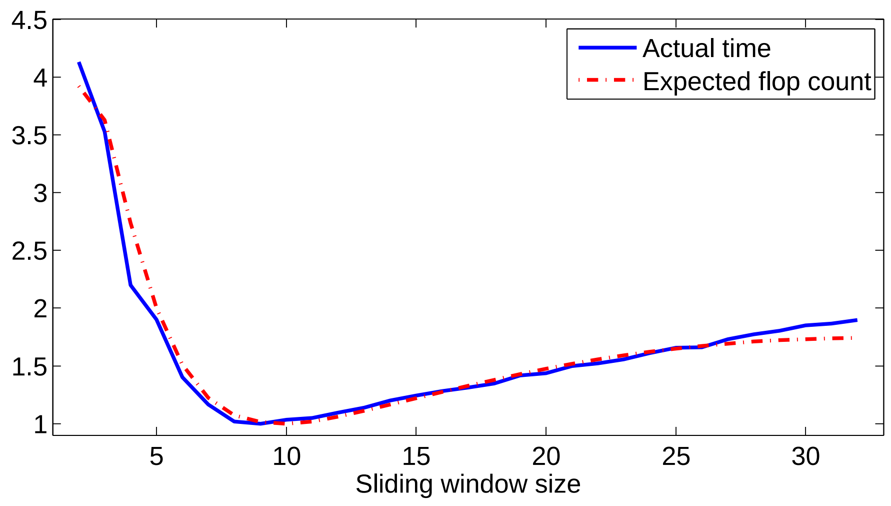


Figure 3.2: Comparison of actual runtime vs. expected flop count. Since the curves have different units, each has been normalized with respect to its minimum value.

value $\bar{f}(m)$ calculated analytically, for a specific trajectory. We can observe that the two curves are very similar, with the only significant differences observed at the two "extremes" of very small or very large m. These regions are less important, however, as they fall outside the typical operational region of the hybrid filter. Thus, using $\bar{f}(m)$ as the objective function to minimize is appropriate.

3.6 Experimental Results

3.6.1 Simulated data

To have a realistic simulation environment, we generate simulation data based on real-world datasets, as described in Appendix A.6. Similarly to Chapter 2, we generate a simulation environment based on the Cheddar Gorge dataset [101],

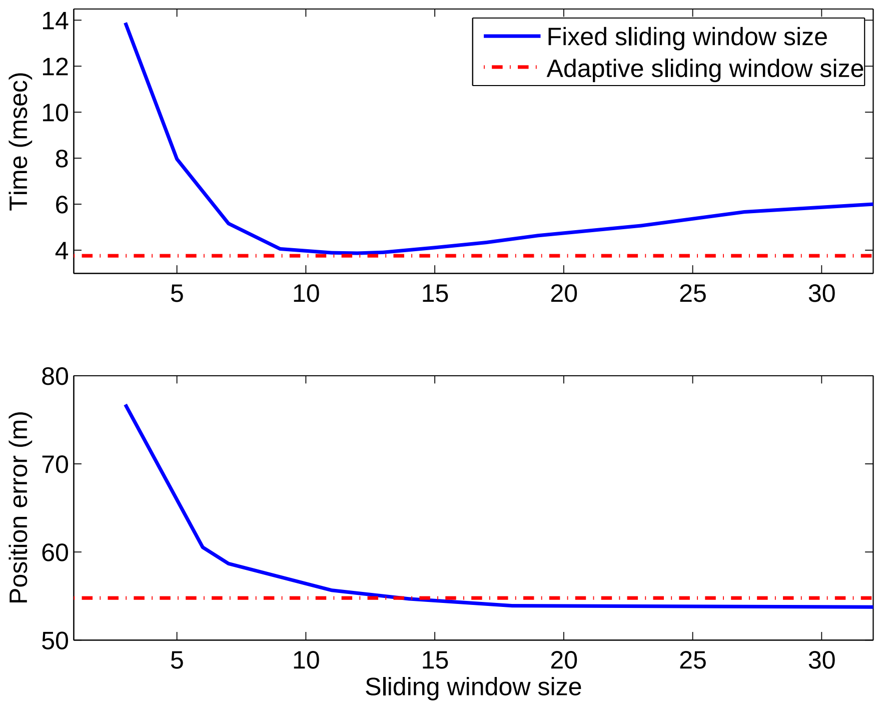


Figure 3.3: Monte-Carlo simulation results: Timing performance and RMS position errors of the hybrid filter, for changing values of m. Timing measured on a laptop computer.

which consists of a 29.6-km long trajectory through canyons, forested areas, and a town. It is important to note that, in the simulation environment, the number of features observed per image and the feature track distribution change in each part of the trajectory, as in the actual dataset (see Fig. 3.1). In the simulations, the IMU measurements are available at 100 Hz, while the camera frame rate is 20 Hz. All the results reported are averages over 50 Monte-Carlo trials.

Fig. 3.3 plots the results from the application of the hybrid filter in this setting. Specifically, the top plot shows the average runtime for each update of the hybrid algorithm. The solid blue line is the average time when m is chosen in advance, and kept constant for the entire trajectory. The red dashed line denotes the runtime achieved when applying the optimization process described in Section 3.5, which optimally chooses m in each time window, to adapt to local feature properties. This plot shows that, by optimizing the value of m in real time, we can attain computational performance higher than that of any fixed value. Note that when m is large no features are initialized, and thus the right-most part of the plot gives the performance of the plain MSCKF 2.0 (similarly, for small m we obtain pure m-SLAM). From Fig. 3.3 we see that the optimal hybrid filter has a runtime 37.17% smaller than that of the MSCKF 2.0, and 72.8% smaller than m-SLAM in this simulation environment.

In the second subplot of Fig. 3.3 we plot the RMS position error, averaged over all Monte-Carlo trials and over the duration of the trajectory. Similarly to the results shown in the previous chapter (see Table 2.2), Fig. 3.3 demonstrates that the MSCKF 2.0 outperforms m-SLAM. We attribute this to the fact that the

MSCKF 2.0 uses better linearization points due to its delayed linearization, and does not use Gaussianity assumptions for the features' positions (see Section 2.3.4). By combining the MSCKF 2.0 with EKF-SLAM some accuracy may be lost, as the errors for the features included in the state vector are now assumed to be Gaussian. However, we can see that if the size of the sliding window increases above a moderate value (e.g., 9 in this case), the change in the accuracy is almost negligible. Intuitively, when a sufficient number of observations is used to initialize features, the feature errors' pdf becomes "Gaussian enough" and the accuracy of the hybrid filter is very close to that of the MSCKF 2.0. Based on these results, in our optimization we do not allow the value of m to fall below a certain threshold (set to 7 in our implementation).

The timing results presented in Fig. 3.3 are obtained on a laptop computer with a Core i7 processor at 2.13GHz, and a single-threaded C++ implementation. Clearly, if the data were to be processed on this computer, the timing performance would easily allow for real-time implementation (the hybrid filter requires fewer than 4 msec per update with optimal m). However, our primary interest is in implementing pose estimation on resource-constrained devices. For this reason, we conducted a second set of tests, in which the data were processed on a Samsung Galaxy S2 mobile phone, equipped with a 1.2-GHz Exynos 4210 processor. For these tests, we ported our C++ implementation to Android using the Android NDK. The simulation data were produced by emulating a real-world dataset collected while driving in a residential area of Riverside, CA. The vehicle trajectory and statistics of the dataset (e.g., feature distribution, feature track length, and so on) are different, allowing us to test the proposed algorithm in a different setting.

Fig. 3.4 shows the results of the hybrid filter in this simulation environment. These results are very similar to what was observed in the first simulation, with the optimal hybrid filter outperforming each of the individual algorithms by a wide margin (runtime 45.8% smaller than the MSCKF 2.0, and 55.6% smaller than m-SLAM). More importantly, however, we observe that the hybrid filter is capable of processing the data at real-time speeds, even on the much less capable processor of the mobile phone. Specifically, the average time needed for each update of the hybrid filter with optimally chosen thresholds is 33.78 msec, corresponding to a rate of approximately 30 images per second. Since the images are recorded at 20 Hz, this means that the proposed hybrid estimator is capable of real-time processing, something that is not possible with any of the individual methods.

3.6.2 Real-world data

In addition to the synthetic datasets described above, we employed our proposed algorithm for processing the feature measurements recorded during a real-world experiment. During this experiment an IMU/camera platform was mounted on top of a car and driven on city streets. The sensors consisted of an Inertial Science ISIS IMU and a PointGrey Bumblebee2 stereo pair (only a single camera's images are used). The IMU provided measurements at 100 Hz, while the camera images were stored at 10 Hz. For obtaining feature tracks, Harris feature points

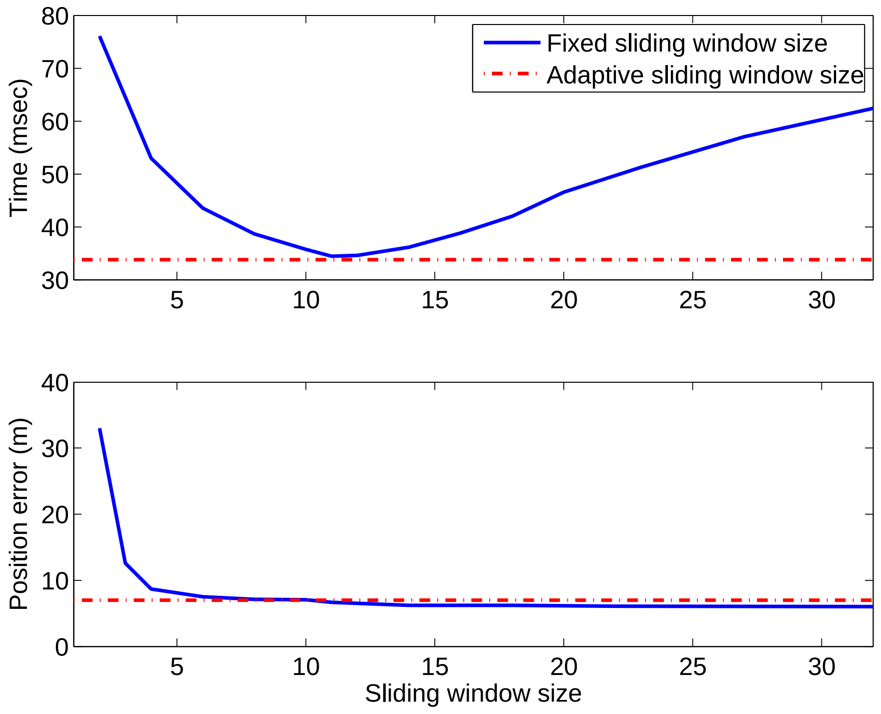


Figure 3.4: Monte-Carlo simulation results: Timing performance and RMS position errors of the hybrid filter, for changing values of m. Timing measured on a Samsung Galaxy S2 mobile phone. Note that the dataset used here is different from the one used in Fig. 3.3, hence the difference in accuracy.

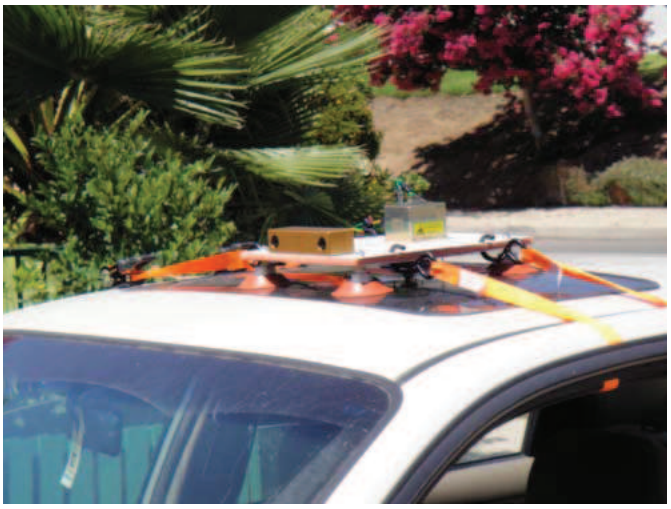


Figure 3.5: The IMU/camera platform used for the real-world experiment presented in Section 3.6.2.

are extracted and matching is carried out by normalized cross-correlation. The vehicle trajectory is approximately 5.5 km long, and a total of 7922 images are processed.

In this dataset, to compensate for the low frame rate of the images, we use lower feature-detection thresholds, which leads to a larger number of features. Specifically, 540 features are tracked in each image on average, and the average track length is 5.06 frames. This substantially increases the overall computational requirements of all algorithms. When run on the mobile phone's processor, the average time per update is 139 msec for the MSCKF 2.0, 774 msec for m-SLAM, and 77 msec for the hybrid filter with optimal m. In Fig. 3.6 the trajectory estimates computed by the three methods are plotted on a map of the area where the experiment took place. We observe that the accuracy of the MSKCF 2.0 and the hybrid filter are similar, and substantially better than that of m-SLAM.

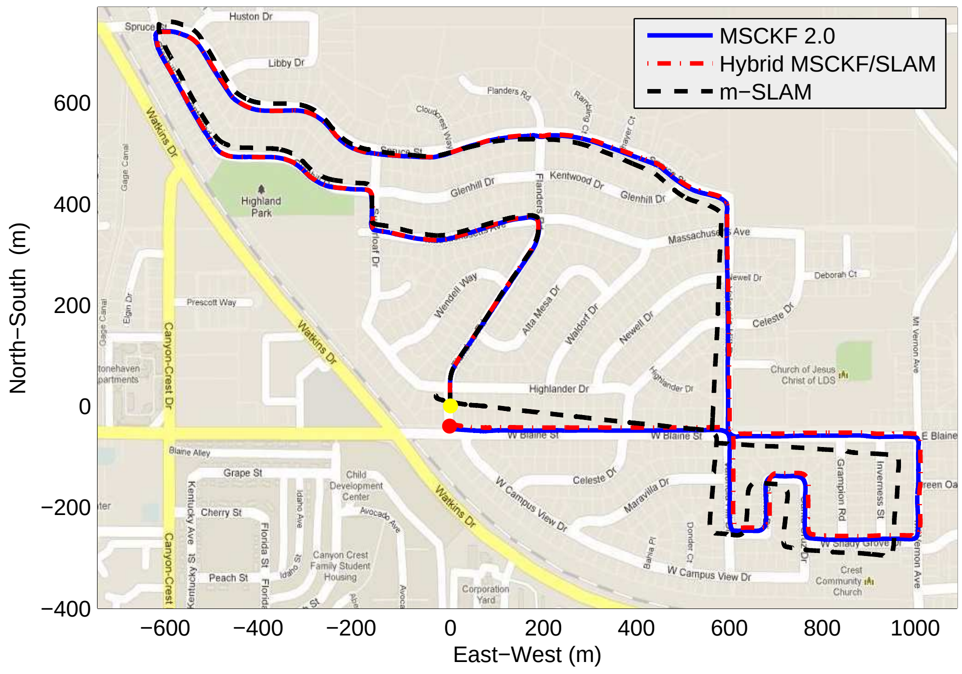


Figure 3.6: The trajectory estimates of the MSCKF 2.0, m-SLAM, and the hybrid algorithm in the real-world experiment.

In Fig. 3.7 we plot the computed values for the optimal m during the experiment. This figure shows that, due to the changing properties of the feature tracks' distribution, the optimal value varies considerably over time, justifying the need for periodic re-optimization. As a final remark, we note that the optimization process itself is computationally inexpensive. In our implementation, the optimal threshold is re-computed every 15 sec, and this process takes 0.31 msec, on average, on the mobile phone. Therefore, the optimization takes up fewer than 0.003% of the overall processing time, while resulting in substantial performance gains.

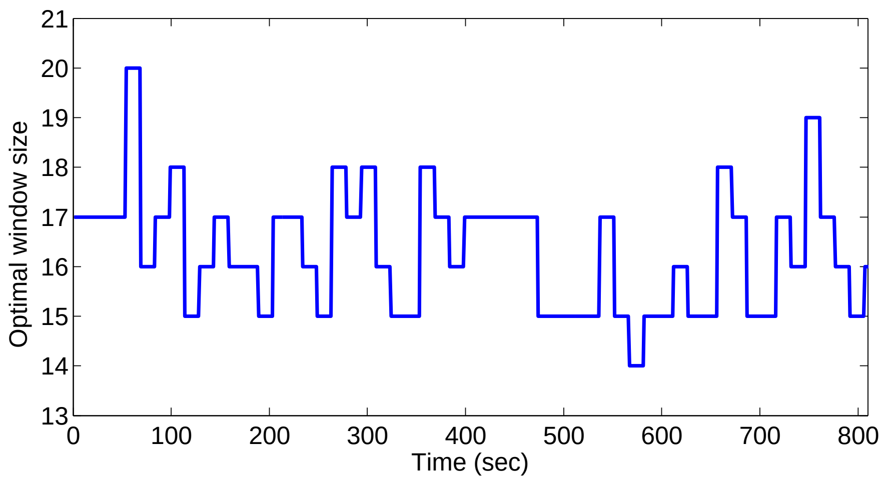


Figure 3.7: The length of the sliding window selected via the optimization process for the real-world experiment.

3.7 Conclusion

In this chapter, we presented a hybrid MSCKF/SLAM estimator for visual-inertial odometry, as well as a methodology for optimizing the estimator's computational efficiency. The simulation and experimental results demonstrate that the proposed algorithm offers significant performance gains over both the MSCKF 2.0 and m-SLAM algorithms individually. For the results presented in this chapter, we used the processor of the mobile phone to process data offline, to prove the feasibility of real-time localization using a processor of this kind. The successful results presented in this chapter served as the motivation for the implementation of online VIO on a mobile phone, presented in Chapter 5.

Chapter 4

Online Spatial and Temporal Calibration

4.1 Introduction

In the previous chapters, we proposed algorithms for performing motion estimation in unknown environments using the measurements from visual and inertial sensors. However, in the proposed estimation algorithms, i.e., the MSCKF 2.0 and MSCKF/SLAM hybrid estimators, we have assumed that the camera-to-IMU spatial and temporal configuration is perfectly known in advance. This, however, might not be always possible, especially for low-cost, resource-constrained systems. Thus, in this chapter we focus on addressing the camera-to-IMU spatial and temporal calibration problems.

The first of these problems, often termed extrinsic sensor calibration, is that of determining the spatial transformation, consisting of the relative rotation and position, between the camera and the IMU. If inaccurate spatial calibration is used for fusing the sensors' measurements, both the consistency and the accuracy of the estimator will substantially deteriorate. In the past few years, the necessity of accurate spatial calibration has been recognised, and several algorithms have been proposed for this task [43, 119].

On the other hand, temporal calibration is a less-explored topic. When processing the sensor measurements in an estimator, a timestamp is typically obtained for each camera image and IMU sample. This timestamp is taken either from the sensor itself, or from the operating system (OS) of the computer receiving the data. These timestamps, however, are typically inaccurate. Due to the time needed for data transfer, sensor latency, and OS overhead, a delay – different for each sensor – exists between the actual sampling of a measurement and its timestamp. Additionally, if different clocks are used for timestamping (e.g., on different sensors), these clocks may suffer from clock skew. As a result, an unknown *time offset*, t_d, typically exists between the timestamps of the camera and the IMU. If this time offset is not accurately known and accounted for, it will introduce unmodeled errors in the estimation process, and reduce its accuracy. By the term temporal calibration, in this chapter we refer to the process of estimating the time offset t_d.

To date, most existing visual-inertial odometry algorithms have assumed that the camera-to-IMU temporal calibration is perfectly *known* (e.g. [20, 39, 80]). This assumption is true only when the hardware characteristics are accurately known, or if an accurate offline calibration procedure is performed in advance. However, in practice, accurate timing information is *not* readily available for most user-built systems, especially the low-cost ones, and performing offline calibration requires extra effort and time. Moreover, even if offline calibration is performed, some of the calibration parameters may slowly drift over time (e.g. t_d may drift due to clock skew). In such cases, offline calibration is ineffective. To this end, in this chapter, we propose an approach for performing concurrent motion estimation and online spatial *and* temporal calibration in vision-aided inertial navigation, and analyze its properties.

Specifically, the first contribution presented in this chapter is a formulation for the online estimation of the spatial and temporal calibration parameters, by using the proposed MSCKF/SLAM hybrid estimator. Specifically, we include the time offset and camera-to-IMU frame transformation in the state vector of the hybrid estimator, and estimate them concurrently with all other variables, i.e., the evolving IMU state, IMU poses, and feature states. Compared to the MSCKF/SLAM hybrid algorithm presented in Chapter 3, which requires the calibration parameters to be perfectly known in advance, the proposed approach only incurs a minimal computational overhead, as it only requires seven additional variables to be included in the filter error-state vector.

The second main contribution of this work is the analysis of the identifiability of the calibration parameters. We note that the identifiability of the spatial calibration parameters has been studied in recent years [43, 50], and it has been shown that these are generally identifiable, when the temporal calibration is perfectly known. We here extend these results to the case of joint spatial and temporal calibration. Specifically, we prove that the time offset t_d is *locally identifiable* in general trajectories. Moreover, we characterize the critical trajectories that cause loss of identifiability, and show that these are either (i) cases that are known to be degenerate even when t_d is perfectly known (e.g., constant-velocity motion), or (ii) cases that are unlikely to occur in practice. Thus, including the time offset in the estimated state vector does not introduce new, practically significant critical trajectories for the overall system's observability.

These theoretical results have direct practical implications. They prove that, when calibration parameters are not perfectly known in advance, they *can* be estimated online by the proposed EKF-based algorithm, together with all the other quantities of interest. The identifiability properties guarantee that this estimation will be successful, even if the quantities (e.g. the time offset) are slowly drifting over time. Our experimental and simulation results confirm that the proposed methods yield high-precision, consistent state estimates, for both the calibration parameters and the motion of the platform, in cases of either constant or time-varying t_d. Importantly, we show that the accuracy obtained when the calibration parameters are estimated online is almost indistinguishable from the precision we would obtain if these were perfectly known in advance. These results, together with the fact online calibration causes only a minimal increase in the estimator's computational cost, demonstrate the practical advantages of online calibration for camera-IMU systems.

4.2 Related Work

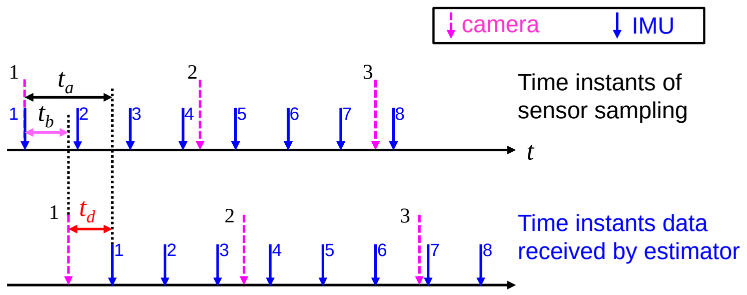
In recent years, it has been shown that the camera-to-IMU spatial calibration can be successfully performed concurrently with visual-inertial SLAM or VIO [43,50,118]. However, all such approaches assume perfectly known temporal calibration. The latter problem is a much less studied one, and it is the main focus of our work. We therefore here focus on prior work related to estimation using temporally misaligned measurements.

Sensor latency is a common problem, and therefore the topic of state estimation with time-delayed and time-offset measurements has been studied in several contexts. The vast majority of existing approaches focus on the problem of using delayed sensor measurements for estimation, when the delay is *perfectly* known in advance (see, e.g., [5,6,125], and references therein). The vision-aided inertial navigation method of [118] belongs to this category. Moreover, a number of methods have been proposed for the case where the time offset is only *approximately* known [16,45]. However, all these algorithms use the time offset between the sensors as an *input*: they do not attempt to estimate it, or to improve a prior estimate using additional data, and are thus not applicable to the problem we address.

To the best of our knowledge, the first work addressing the problem of time-offset estimation in camera-IMU systems in a principled manner is that of [49]. In that work, rotation estimates from each individual sensor are first computed, and then temporally aligned via batch ICP-like registration in the space of rotations. This technique can be applied to sensors beyond just cameras and IMUs [114]. While this approach addresses the presence of a time offset and its estimation in a rigorous manner, it has two limitations. First, being offline in nature, it cannot operate in the presence of time-varying time offsets. Moreover, since only the rotation measurements are used, this method does not utilize all the available measurement information.

We note that the standard way to estimate the time-shift between two signals is by determining the peak of the cross-correlation between them (see, e.g. [26, 31] and references therein). This technique could be used to estimate the time offset between the camera and IMU, by correlating the rotation estimates computed by the two sensors. Moreover, a number of approaches exist that determine the timing between sensors using low-level data, such as direct measurements of network response times (see, e.g., [35] and references therein). Conceivably, any of these methods could be used online, in parallel to the state estimator, to provide estimates for the time offset t_d. To deal with time-varying offsets, the methods could run periodically. However, this would lead to a more complex implementation than the solution proposed in this work, and would not be able to properly model the uncertainty of t_d in the estimator.

In contrast to the methods discussed above, the approach proposed in our work allows for online estimation of t_d, by treating it as an additional state to be estimated. This idea, which has recently also been proposed in the context of GPS-based navigation [102], makes it possible to use all the measurements from the camera, gyroscope, and accelerometer in a tightly-coupled formulation. Moreover, it models the uncertainty in the estimate of t_d, and its impact on the accuracy of motion estimation, in a natural way via the EKF covariance matrix.



camera
IMU
Time instants of sensor sampling
Time instants data received by estimator
t_a
t_b
t_d
t

Figure 4.1: Example of time offset arising due to latency in the sensor data. In this case the IMU data arrives with a latency t_a, while the camera data have a latency t_b, both of which are unknown. Since t_a and t_b are different, the estimator receives measurements that were recorded simultaneously (e.g., the first IMU and camera measurement) with timestamps that are offset by $t_d = t_a - t_b$.

Even though the observability properties of vision-aided inertial navigation, including the camera-to-IMU spatial calibration, have been studied in great detail in the past, all prior work assumes perfect knowledge of the time offset between sensors [43, 50, 75, 78]. Our identifiability analysis makes it possible to extend the results of the prior work to the unknown-t_d case. Specifically, in Section 4.5 we prove that even if t_d is unknown, it can be determined based on the sensor data, except in a small set of degenerate cases. Therefore, unless the trajectory is one of the degenerate ones, we can view the problem as one where t_d is known, and the results of [43, 50] apply.

4.3 Time-offset Definition

Consider a system comprising a camera and IMU, in which each sensor provides measurements at a constant frequency, known at least to a good approximation. As described in Section 4.1, an unknown time offset, t_d, generally exists between the reported timestamps of the visual and inertial measurements. Fig. 4.1 illustrates how a time offset can arise due to difference the sensors' latency, but t_d may also arise due to synchronization errors, missed data, and clock skew. Note that, depending on the system at hand, t_d may have a positive or negative value. For instance, if the offset is caused due to sensor latency, then t_d will be positive when the IMU has a longer latency than the camera, and negative in the opposite case. In our formulation, we use the "IMU time" as the time reference by convention. Therefore, $\boldsymbol{\zeta}(t)$ denotes the value of a quantity $\boldsymbol{\zeta}$ at the time instant the IMU measurement with timestamp t was recorded. On the other hand, if an image with timestamp t is received from the camera, this image was actually captured at time $t + t_d$. We point out that in our formulation it is possible (as proven in Section 4.5) to identify the time offset t_d, but not the individual latencies of the sensors. However, these latencies cannot be determined unless additional, external state information is available. By using the "IMU time" as the time reference we circumvent this difficulty, and obtain equations that only involve the time offset t_d, which we can estimate using only the camera and IMU measurements.

4.4 Estimator Formulation

Our goal is to perform motion estimation in an unknown environment, and concurrent online temporal and spatial calibration between the visual and inertial sensors. To this end, we include the temporal and spatial calibration parameters into the state vector of MSCKF/SLAM hybrid estimator described in Chapter 3. Specifically, we define the state vector of the new estimator as:

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_{E_k}^T & \boldsymbol{\pi}_{IC}^T & t_d & \mathbf{x}_{I_{k-1}}^T & \mathbf{x}_{I_{k-2}}^T & \cdots & \mathbf{x}_{I_{k-m}}^T & \mathbf{f}_1^T & \cdots & \mathbf{f}_{s_k}^T \end{bmatrix}^T \tag{4.1}$$

where $\boldsymbol{\pi}_{IC}$ is the spatial transformation (rotation and translation) between the camera and IMU, defined as:

$$\boldsymbol{\pi}_{IC} = \begin{bmatrix} {}^C_I\bar{\mathbf{q}}^T & {}^C\mathbf{p}_I^T \end{bmatrix}^T \tag{4.2}$$

In the MSCKF/SLAM hybrid EKF, the IMU measurements are used to propagate the state and covariance estimates. Since the camera measurements are not employed in the propagation process, including the calibration parameters in the state will not change the propagation equations of the other states. Specifically, the state propagation equations for the evolving state, $\mathbf{x}_{E_k}$, the IMU poses, $\mathbf{x}_{I_j}$, $j = k-1, \ldots, k-m$, and the feature states, $\mathbf{f}_j$, $j = 1, \ldots, s_k$, are identical to those employed in the original MSCKF/SLAM hybrid algorithm (see Section 2.5).

To describe the propagation equations for the calibration states, we first present their motion dynamics:

$$ {}^{C}_{I}\dot{\mathbf{q}}(t) = \mathbf{0} \tag{4.3}$$

$$ {}^{C}\dot{\mathbf{p}}_{I}(t) = \mathbf{0} \tag{4.4}$$

$$ \dot{t}_d(t) = 0 \tag{4.5}$$

The above equations model the scenario where the calibration parameters remain constant. If instead some of the parameters are known to drift over time, we can model them as random-walk processes. For instance, if the time offset is known to be time-varying, we can model it as a random-walk process by replacing (4.5) with $\dot{t}_d(t) = n_d(t)$, where $n_d(t)$ is a white Gaussian noise process, whose power spectral density expresses the variability of t_d. Based on (4.3)-(4.5), we see that during EKF propagation the state estimates and covariance of the calibration parameters remain unchanged.

4.4.1 State augmentation

As described in Section 2.3.3 and Section 3.4, every time a new image is available, the state vector is augmented with the IMU pose corresponding to the new image. Note that, if no time offset existed (or, equivalently, if it was perfectly known *a priori*), the state augmentation would present no difficulty. The complications arise from the fact that the image received by the filter at time t was in fact recorded at time $t + t_d$, where t_d is now a random variable. Therefore, when a new image is received with timestamp t, the IMU pose that needs to be included into the state is the pose at the time $t + t_d$. To this end, we perform EKF propagation using the IMU measurements up to timestep $t + \hat{t}_d$, and insert the estimated IMU pose $\hat{\mathbf{x}}_I(t + \hat{t}_d)$ in the state. The IMU pose

$$\hat{\mathbf{x}}_I(t + \hat{t}_d) = \left[{}^I_G\hat{\bar{\mathbf{q}}}^T(t + \hat{t}_d), {}^G\hat{\mathbf{p}}_I^T(t + \hat{t}_d)\right]^T \tag{4.6}$$

can be obtained by copying the state estimates ${}^I_G\hat{\bar{\mathbf{q}}}(t + \hat{t}_d)$ and ${}^G\hat{\mathbf{p}}_I(t + \hat{t}_d)$ from the evolving state $\hat{\mathbf{x}}_E(t + \hat{t}_d)$.

The filter covariance matrix is also augmented, as:

$$\mathbf{P}(t+\hat{t}_d) \leftarrow \begin{bmatrix} \mathbf{P}(t+\hat{t}_d) & \mathbf{P}(t+\hat{t}_d)\mathbf{J}_{new}^T \\ \mathbf{J}_{new}\mathbf{P}(t+\hat{t}_d) & \mathbf{J}_{new}\mathbf{P}(t+\hat{t}_d)\mathbf{J}_{new}^T \end{bmatrix} \tag{4.7}$$

where $\mathbf{J}_{new}$ is the Jacobian of $\mathbf{x}_I(t + t_d)$ with respect to the state vector. This matrix has the following structure:

$$\mathbf{J}_{new} = \begin{bmatrix} \mathbf{J}_E & \mathbf{0}_{6\times6} & \mathbf{J}_t & \mathbf{0}_{6\times6m} \end{bmatrix} \tag{4.8}$$

where $\mathbf{J}_E$ is the Jacobian with respect to the evolving state:

$$\mathbf{J}_E = \begin{bmatrix} \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times9} \\ \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{0}_{3\times9} \end{bmatrix} \tag{4.9}$$

and $\mathbf{J}_t$ is the Jacobian with respect to t_d:

$$\mathbf{J}_t = \begin{bmatrix} {}^I_G\hat{\mathbf{R}}^T(t + \hat{t}_d)^I\hat{\boldsymbol{\omega}}(t + \hat{t}_d) \\ {}^G\hat{\mathbf{v}}_I(t + \hat{t}_d) \end{bmatrix} \tag{4.10}$$

A few interesting comments can be made at this point. We start by noting that the image was recorded at time $t+t_d$, but the corresponding IMU pose is computed at $t + \hat{t}_d$. Since the estimate of the time offset will inevitably contain some errors, the "clone" of the IMU pose will inevitably be included in the state vector at a slightly incorrect time instant. However, the EKF *explicitly accounts for* this fact.

Specifically, since t_d is included in the estimated state vector, the filter keeps track of the uncertainty in $\hat{t}_d$, via the Jacobian matrix $\mathbf{J}_t$. Therefore, when performing an EKF update using the poses included in the state, the uncertainty in the time offset is explicitly modelled, and is accounted for in the computation of the state update. As a result, we are able to obtain both more accurate pose estimates and a better characterization of their uncertainty.

It is worth pointing out that, in some cases, the camera timestamps may be affected by a random-noise component ("jitter"), in addition to a systematic time offset. In the proposed formulation, it is straightforward to model these random effects in the estimator's equations. Specifically, if each image timestamp is affected by an independent, zero-mean, random error with standard deviation σ_t, then, instead of (4.7), the state augmentation equation is computed as:

$$\mathbf{P}(t+\hat{t}_d) \leftarrow \begin{bmatrix} \mathbf{P}(t+\hat{t}_d) & \mathbf{P}(t+\hat{t}_d)\mathbf{J}_{new}^T \\ \mathbf{J}_{new}\mathbf{P}(t+\hat{t}_d) & \mathbf{J}_{new}\mathbf{P}(t+\hat{t}_d)\mathbf{J}_{new}^T + \sigma_t^2\mathbf{J}_t\mathbf{J}_t^T \end{bmatrix} \tag{4.11}$$

This modification makes it possible to model the additional timestamp uncertainty, and to account for it in the EKF update equations.

4.4.2 EKF update

Compared to the hybrid filter update equations described in Chapter 3, only minimal modifications are required in the EKF update equations, to account for the inclusion of ${}^C_I\bar{\mathbf{q}}$ and ${}^C\mathbf{p}_I$ in the state. Specifically, the linearized residual equations for processing the "MSCKF" features (2.15) become

$$\begin{aligned}\tilde{\mathbf{z}}_{ij} &= \mathbf{z}_{ij} - \mathbf{h}(\hat{\mathbf{x}}_{I_{j|k-1}}, \hat{\boldsymbol{\pi}}_{IC_{k|k-1}}, {}^G\hat{\mathbf{p}}_{f_i}) \\ &\simeq \mathbf{H}_{\mathbf{x}_{ij}}\tilde{\mathbf{x}}_{I_{j|k-1}} + \mathbf{H}_{\mathbf{c}_{ij}}\tilde{\boldsymbol{\pi}}_{IC_{k|k-1}} + \mathbf{H}_{\mathbf{f}_{ij}}{}^G\tilde{\mathbf{p}}_{f_i} + \mathbf{n}_{ij} \end{aligned} \tag{4.12}$$

where $\mathbf{H}_{\mathbf{x}_{ij}}$, $\mathbf{H}_{\mathbf{f}_{ij}}$, and $\mathbf{H}_{\mathbf{c}_{ij}}$ are the Jacobian matrices with respect to the IMU pose, feature position, and the spatial calibration parameters. These Jacobian matrices can be computed similarly to (2.48)-(2.50):

$$\mathbf{H}_{\mathbf{f}_{ij}} = \mathbf{J}_{ij}\,{}^C_I\hat{\mathbf{R}}\,{}^{I_j}_G\hat{\mathbf{R}} \tag{4.13}$$

$$\mathbf{H}_{I_{ij}} = \mathbf{H}_{\mathbf{f}_{ij}}\left[\lfloor({}^G\hat{\mathbf{p}}_{f_i} - {}^G\hat{\mathbf{p}}_{I_j})\times\rfloor \quad -\mathbf{I}_3\right] \tag{4.14}$$

$$\mathbf{H}_{\mathbf{c}_{ij}} = \mathbf{J}_{ij}\left[{}^C_I\hat{\mathbf{R}}\,\lfloor{}^{I_j}_G\hat{\mathbf{R}}({}^G\hat{\mathbf{p}}_{f_i} - {}^G\hat{\mathbf{p}}_{I_j})\times\rfloor \quad \mathbf{I}\right] \tag{4.15}$$

$$\mathbf{J}_{ij} = \frac{1}{{}^{C_j}\hat{z}_{f_i}}\begin{bmatrix} 1 & 0 & \frac{-{}^{C_j}\hat{x}_{f_i}}{{}^{C_j}\hat{z}_{f_i}} \\ 0 & 1 & \frac{-{}^{C_j}\hat{y}_{f_i}}{{}^{C_j}\hat{z}_{f_i}} \end{bmatrix} \tag{4.16}$$

with $[{}^{C_j}\hat{x}_{f_i} \;\; {}^{C_j}\hat{y}_{f_i} \;\; {}^{C_j}\hat{z}_{f_i}]^T$ being the estimate of the feature position with respect to the camera:

$$\begin{bmatrix} {}^{C_j}\hat{x}_{f_i} \\ {}^{C_j}\hat{y}_{f_i} \\ {}^{C_k}\hat{z}_{f_i} \end{bmatrix} = {}^C_I\hat{\mathbf{R}}\,{}^{I_j}_G\hat{\mathbf{R}}({}^G\hat{\mathbf{p}}_{f_i} - {}^G\hat{\mathbf{p}}_{I_k}) + {}^C\hat{\mathbf{p}}_I \tag{4.17}$$

Similarly to (4.12)-(4.16), the linearized equations for processing the SLAM features can be also derived, by accounting for the calibration parameters ${}^C_I\bar{\mathbf{q}}$ and ${}^C\mathbf{p}_I$. Once the linearized equations are formulated, the remaining steps of the estimator are identical to that of the hybrid filter, as described in Section 3.4.2.

4.4.3 Implementation

The algorithm for concurrent pose estimation and online calibration described in the preceding sections can be implemented in a number of different ways. We have opted for a multi-threaded approach, as it allows for increased flexibility and extensibility. In our implementation, each sensor's readings are managed by a separate thread. Two queues are maintained (one for the IMU measurements and one for the camera images), and each new measurement is timestamped and placed in the appropriate queue as it arrives.

One thread is used for implementing the EKF equations. This thread waits until the image queue has at least one image, at which point it begins using the measurements in the IMU queue to propagate the state estimate and its covariance matrix. If the timestamp of the the first image in the image queue is t, then IMU measurements are used to propagate up to time $t + \hat{t}_d$. After propagation is completed, the EKF update is performed, as described in Sections 4.4.1-4.4.2. We note here that, in general, the time instant $t + \hat{t}_d$ will fall between two consecutive sample times of the IMU. Therefore, we employ linear interpolation to obtain an inferred IMU measurement at $t + \hat{t}_d$, used for propagation as well as in computing the estimated rotational velocity in (4.10).

Finally, we point out that if the camera images are delayed relative to the IMU (i.e., if $t_d < 0$), the EKF thread will produce state estimates with a latency, as it waits until an image is available before performing the state augmentation. If estimates of the current state are required at the IMU sample rate, these can be computed by a separate thread running in parallel. Specifically, this thread can use the latest available state estimate from the EKF, as well as all the measurements in the IMU queue, to compute the estimate for the current system state via propagation.

4.5 Identifiability Analysis

In the preceding sections we presented an online algorithm for estimating the spatial and temporal relationship between the camera and IMU. However, for the proposed algorithm to be able to obtain meaningful results, both t_d and $\boldsymbol{\pi}_{IC}$ must be *identifiable* (equivalently, observable[1]) given the available measurements. We note that, the camera-to-IMU spatial parameter, $\boldsymbol{\pi}_{IC}$, has been already studied in recent years, and it was shown that $\boldsymbol{\pi}_{IC}$ is identifiable in general trajectories [43, 50]. However, the identifiability of t_d still remains an unexplored topic in vision-aided inertial navigation literatures. Thus, in this section, we provide a detailed analysis of the identifiability properties of t_d, and prove that t_d is in general locally identifiable [10].

4.5.1 Camera measurement model

We begin by examining the type of information provided by the camera measurements. When a camera is observing (a sufficient number of) unknown features, the feature measurements can be used to compute (i) the orientation of the camera with respect the initial camera frame, and (ii) the position of the camera with

[1]Note that both $\boldsymbol{\pi}_{IC}$ and t_d can be viewed as either fixed parameters to be estimated or as states of a dynamical system, with zero dynamics. Therefore, we can use the terms identifiability and observability interchangeably.

respect to the initial camera frame, up to an unknown scale factor s [36]. That is, by processing the visual measurements in the time interval $[0, t]$, we can compute a measurement of the camera rotation in the time interval $[t_d, t+t_d]$, and a scaled measurement of the camera translation in the same time interval:

$$\mathbf{R}_c(t) = {}^{C_o}_{C}\mathbf{R}(t+t_d)e^{\lfloor \mathbf{n}_{cr}(t)\times \rfloor} \tag{4.18}$$

$$\mathbf{p}_c(t) = s\,{}^{C_o}\mathbf{p}_C(t+t_d) + \mathbf{n}_{cp}(t) \tag{4.19}$$

where $\mathbf{n}_{cr}(t)$, and $\mathbf{n}_{cp}(t)$ are noise vectors, and $\{C_o\}$ is the initial camera frame, i.e., the camera frame at the time instant t_d. Note that equivalently, we can state that the visual measurements can be used to compute (i) a measurement, $\boldsymbol{\omega}_c$, of the camera's rotational velocity, and (ii) a scaled measurement, $\mathbf{v}_c$, of the camera's translational velocity:

$$\boldsymbol{\omega}_c(t) = {}^{C}\boldsymbol{\omega}(t+t_d) + \mathbf{n}_{c\omega}(t) \tag{4.20}$$

$$\mathbf{v}_c(t) = s\,{}^{C_o}\mathbf{v}_C(t+t_d) + \mathbf{n}_{cv}(t) \tag{4.21}$$

where $\mathbf{n}_{c\omega}(t)$, and $\mathbf{n}_{cv}(t)$ are the measurement noise vectors.

We will carry out our analysis by using the measurement models described above, instead of the "raw" feature measurements. We stress that (4.18)-(4.19) or (4.20)-(4.21) contain all the information that the visual feature measurements provide for the camera motion, and therefore we can use these, instead of the feature measurements, for our analysis. The advantage of using the "abstracted" camera measurement models shown above (an approach also employed in [50]) is that their use leads to a significantly simplified analysis.

4.5.2 Overview of the approach

In previous work, the question of which states of the vision-aided inertial navigation system can be estimated and under what conditions, has been addressed by performing an observability analysis. Specifically, [43,50] examined the case where the feature positions, camera-to-IMU spatial transformation, and IMU biases are unknown, but the time offset between the camera and IMU is *known*. For this scenario it was shown that, in general, the following quantities are (locally weakly) observable:

(O1) The trajectory expressed in the initial camera frame.

(O2) The gravity vector expressed in the initial camera frame.

(O3) The gyroscope and accelerometer biases.

(O4) The camera-to-IMU spatial transformation, and

(O5) The positions of the features expressed in the camera frame.

The quantities O1-O5 are *generally* observable, that is, unless the camera follows "degenerate" trajectories such as constant-velocity motion or rotation about a single axis (see [43,50] for a complete characterization). On the other hand, four degrees of freedom are *always* unobservable: three corresponding to the system's position in the global frame, and one to the rotation about gravity (i.e., the yaw).

The approach we follow in order to examine the local identifiability of t_d can be explained, at a high level, as follows. We start by defining a vector $\boldsymbol{\xi}$, which contains the time offset t_d, as well as all the additional quantities needed in order to compute O1-O5. Specifically, we define $\boldsymbol{\xi}$ as:

$$\boldsymbol{\xi} = \begin{bmatrix} {}^{C_o}\mathbf{v}_o^T & {}^{C_o}\mathbf{g}^T & \mathbf{b}_\mathbf{g}^T & \mathbf{b}_\mathbf{a}^T & {}^{C}\mathbf{p}_I^T & {}_I^C\mathbf{q}^T & t_d & s \end{bmatrix}^T$$

where ${}^{C_o}\mathbf{v}_o$ is the IMU velocity at time instant t_d, expressed in $\{C_o\}$, and ${}^{C_o}\mathbf{g}$ is the gravity vector expressed in the same frame. We stress that the elements of $\boldsymbol{\xi}$ describe all the *potentially observable* quantities in the system, and none of the quantities that are known to be unobservable. To see why, we first note that $\boldsymbol{\xi}$ contains the time offset t_d, as well as O2, O3 and O4 explicitly. Moreover, using the elements of $\boldsymbol{\xi}$ and the measurements, we can compute the velocity of the camera at any time instant, expressed in the frame $\{C_o\}$ (as shown in (4.29)). Integrating this velocity yields the camera trajectory in $\{C_o\}$, which is quantity O1. Finally, once the trajectory of the camera is known, then the features' positions (quantity O5) can be computed by triangulation using the camera measurements.

The above discussion shows that if $\boldsymbol{\xi}$ is locally identifiable given the measurements, then t_d and O1-O5 are locally weakly observable. To determine whether $\boldsymbol{\xi}$ is locally identifiable, we employ the camera and IMU measurements, along with the system dynamics, to derive *constraint equations* that $\boldsymbol{\xi}$ must satisfy. These constraints have the general form:

$$\mathbf{c}_i\left(\boldsymbol{\xi}, \mathbf{z}_{[0,t]}, \boldsymbol{\omega}_{m[0,t]}, \mathbf{a}_{m[0,t]}\right) = \mathbf{0} \tag{4.22}$$

where $\mathbf{z}_{[0,t]}, \boldsymbol{\omega}_{m,[0,t]}$, and $\mathbf{a}_{m,[0,t]}$ are the camera, gyroscope, and accelerometer measurements in the time interval $[0, t]$. Once these constraints are formulated, we can check the local identifiability of $\boldsymbol{\xi}$ by examining the rank of the Jacobian matrices of these constraints with respect to $\boldsymbol{\xi}$.

Since the conditions for the observability of O1-O5 have been derived in previous work, we here focus on examining the identifiability of t_d. The main result, proven in Lemma 2, is that t_d is locally identifiable, except in a small number of degenerate motion cases that can be characterized in detail. This result validates the use of the proposed estimator described in the preceding sections. Moreover, it provides us with a way to examine the observability of the entire system (including t_d and O1-O5), by leveraging the results of [43,50]. Specifically, when t_d is identifiable, we can use the measurements alone to determine its value; therefore in terms of observability, the system can be treated as one with a known t_d in this case. Put differently, when t_d is identifiable, the observability properties of O1-O5 are *identical to those derived in previous work for the known-t_d case.* Thus, to certify that all the quantities of interest (t_d and O1-O5) are observable, one can use Lemma 2 to verify that t_d is locally identifiable, and then employ the results of [43, 50] to verify the observability of O1-O5.

Our analysis of the identifiability of t_d consists of two steps. First, we show that, even if *only* the rotational components of the motion (i.e., rotational constraints) are considered, the time offset is in general locally identifiable, and we obtain a concrete description of the cases that cause loss of identifiability. We then show that if all the available information is utilized (which is the case in the algorithms described in the preceding sections), the set of degenerate cases is further restricted. The details of the analysis are presented in what follows.

4.5.3 Derivation of the constraints for $\boldsymbol{\xi}$

We here ignore the noise (as standard in an identifiability/observability analysis) and derive two constraint equations: one based on the rotational velocity measurements, and one based on the accelerometer and (scaled) velocity measurements. We begin by using (4.20) and (2.28), and the identity ${}^{C}\boldsymbol{\omega}(t) = {}^{C}_{I}\mathbf{R}\,{}^{I}\boldsymbol{\omega}(t)$, to obtain:

$$\boldsymbol{\omega}_m(t) = {}^{C}_{I}\mathbf{R}^T \boldsymbol{\omega}_c(t - t_d) + \mathbf{b_g} \tag{4.23}$$

which we can write in the form of (4.22) as follows:

$$\mathbf{c}_1(\boldsymbol{\xi}, t) = {}^{C}_{I}\mathbf{R}^T \boldsymbol{\omega}_c(t - t_d) + \mathbf{b_g} - \boldsymbol{\omega}_m(t) = \mathbf{0} \tag{4.24}$$

This constraint involves the known functions of the IMU and camera rotational velocity measurements, as well as the unknown parameters t_d, ${}^{C}_{I}\mathbf{R}$ (equivalent to ${}^{C}_{I}\bar{\mathbf{q}}$), and $\mathbf{b_g}$.

To obtain the second constraint, we express $\mathbf{v}_c(t)$ as a function of $\boldsymbol{\xi}$, $\mathbf{R}_c(t)$, and $\mathbf{a}_m(t)$. We start by writing ${}^{C_o}\mathbf{v}_C(t + t_d)$ as:

$$\begin{aligned} {}^{C_o}\mathbf{v}_C(t + t_d) &= {}^{C_o}_{G}\mathbf{R}\,{}^{G}\mathbf{v}_C(t + t_d) && (4.25) \\ &= {}^{C_o}_{G}\mathbf{R} \left({}^{G}\mathbf{v}_I(t + t_d) - {}^{G}_{C}\dot{\mathbf{R}}(t + t_d)\,{}^{C}\mathbf{p}_I \right) && (4.26) \end{aligned}$$

Next, we note that the IMU velocity at time instant $t + t_d$ can be computed as:

$${}^{G}\mathbf{v}_I(t + t_d) = {}^{G}\mathbf{v}_I(t_d) + \int_0^t {}^{G}_{I}\mathbf{R}(\tau + t_d)\,{}^{I}\mathbf{a}(\tau + t_d) d\tau \tag{4.27}$$

Using (2.29), we can write the last equation as:

$${}^{G}\mathbf{v}_I(t + t_d) = {}^{G}\mathbf{v}_I(t_d) + \int_0^t {}^{G}_{I}\mathbf{R}(\tau + t_d)\mathbf{a}_m(\tau + t_d) d\tau + {}^{G}\mathbf{g}\, t - \int_0^t {}^{G}_{I}\mathbf{R}(\tau + t_d) d\tau \mathbf{b_a} \tag{4.28}$$

Substituting the last equation in (4.26), and using the identity $\mathbf{R}_c(\tau) = {}^{C_o}_{I}\mathbf{R}(\tau + t_d){}^{C}_{I}\mathbf{R}^T$ (see (4.18)), we obtain:

$$\begin{aligned} {}^{C_o}\mathbf{v}_C(t+t_d) &= {}^{C_o}\mathbf{v}_o + {}^{C_o}\mathbf{g}\,t + \int_0^t \mathbf{R}_c(\tau)\,{}^{C}_{I}\mathbf{R}\,\mathbf{a}_m(\tau+t_d)d\tau \\ &\quad - \int_0^t \mathbf{R}_c(\tau)d\tau\,{}^{C}_{I}\mathbf{R}\,\mathbf{b}_\mathbf{a} - \dot{\mathbf{R}}_c(t){}^{C}\mathbf{p}_I \end{aligned} \tag{4.29}$$

where we have used the notation ${}^{C_o}\mathbf{v}_o = {}^{C_o}_{G}\mathbf{R}{}^{G}\mathbf{v}_I(t_d)$. Finally, substitution in (4.21) yields the following constraint (ignoring the noise):

$$\begin{aligned} \mathbf{c}_2(\boldsymbol{\xi}, t) = s\bigg(&{}^{C_o}\mathbf{v}_o + {}^{C_o}\mathbf{g}\,t + \int_0^t \mathbf{R}_c(\tau)\,{}^{C}_{I}\mathbf{R}\,\mathbf{a}_m(\tau+t_d)d\tau \\ &- \int_0^t \mathbf{R}_c(\tau)d\tau\,{}^{C}_{I}\mathbf{R}\,\mathbf{b}_\mathbf{a} - \dot{\mathbf{R}}_c(t){}^{C}\mathbf{p}_I \bigg) - \mathbf{v}_c(t) = \mathbf{0} \end{aligned} \tag{4.30}$$

This equation is the second constraint we sought: it involves terms that are known via the IMU and camera measurements, as well as the elements of $\boldsymbol{\xi}$. In what follows, we show how we can employ (4.24) and (4.30) to determine the identifiability of t_d.

4.5.4 Identifiability of t_d based on the rotational-velocity constraint

We now prove the following result:

Lemma 1 *The time offset t_d is locally identifiable based on the rotational constraints* (4.24) *alone, if no vectors $\mathbf{k}_1$ and $\mathbf{k}_2$ exist, such that the rotational velocity of the IMU satisfies the following differential equation:*

$${}^{I}\dot{\boldsymbol{\omega}}(t) = \lfloor \mathbf{k}_2 \times \rfloor {}^{I}\boldsymbol{\omega}(t) + \mathbf{k}_1 \tag{4.31}$$

Proof. To examine the local identifiability of t_d based on the constraint (4.24), we compute the derivative of $\mathbf{c}_1(\boldsymbol{\xi}, t)$ with respect to the elements of $\boldsymbol{\xi}$ that appear in it:

$$\mathbf{D}_1(t) = \begin{bmatrix} \frac{\partial\, \mathbf{c}_1}{\partial\, \mathbf{b_g}} & \frac{\partial\, \mathbf{c}_1}{\partial\, \tilde{\boldsymbol{\phi}}} & \frac{\partial\, \mathbf{c}_1}{\partial\, t_d} \end{bmatrix} \tag{4.32}$$

where

$$\frac{\partial\, \mathbf{c}_1}{\partial\, \mathbf{b_g}} = \mathbf{I}_{3\times 3} \tag{4.33}$$

$$\frac{\partial\, \mathbf{c}_1}{\partial\, \tilde{\boldsymbol{\phi}}} = -\lfloor {}^C_I\mathbf{R}^T\, \boldsymbol{\omega}_c(t - t_d) \times \rfloor \tag{4.34}$$

$$\frac{\partial\, \mathbf{c}_1}{\partial\, t_d} = -{}^C_I\mathbf{R}^T\, \dot{\boldsymbol{\omega}}_c(t - t_d) \tag{4.35}$$

Note that, since (4.24) must hold for any t, we can generate an infinite number of constraints, by choosing different values of t. A sufficient condition for $\mathbf{b_g}$, ${}^C_I\bar{\mathbf{q}}$, and t_d to be locally identifiable based on these constraints is that there exists a set of time instants, $\mathcal{S} = \{t_1, t_2, \ldots t_s\}$, such that the matrix

$$\begin{bmatrix} \mathbf{D}_1(t_1) \\ \mathbf{D}_1(t_2) \\ \vdots \\ \mathbf{D}_1(t_s) \end{bmatrix} \tag{4.36}$$

has full column rank [22]. Equivalently, a sufficient condition is that there exists no nonzero vector $\mathbf{k} = [\mathbf{k}_1^T\ \ \mathbf{k}_2^T\ \ k_3]^T$ such that, for all $t > 0$, the condition $\mathbf{D}_1(t)\mathbf{k} = \mathbf{0}$ holds. Note that, since we are interested in detecting cases in which t_d is not identifiable, we can restrict our attention to the vectors $\mathbf{k}$ in which k_3 is nonzero. Thus, we can set $k_3 = 1$ (the scaling of $\mathbf{k}$ is arbitrary). Using (4.32)-(4.35), the sufficient condition for t_d to be locally identifiable is that there exist no vectors $\mathbf{k}_1$ and $\mathbf{k}_2$ such that:

$$\mathbf{k}_1 - \lfloor {}^C_I\mathbf{R}^T \boldsymbol{\omega}_c(t-t_d)\times\rfloor\mathbf{k}_2 - {}^C_I\mathbf{R}^T \dot{\boldsymbol{\omega}}_c(t-t_d) = \mathbf{0} \Rightarrow$$

$$\mathbf{k}_1 + \lfloor\mathbf{k}_2\times\rfloor {}^C_I\mathbf{R}^T \boldsymbol{\omega}_c(t-t_d) - {}^C_I\mathbf{R}^T \dot{\boldsymbol{\omega}}_c(t-t_d) = \mathbf{0}, \forall t > 0 \tag{4.37}$$

Using the identity ${}^C_I\mathbf{R}^T \boldsymbol{\omega}_c(t-t_d) = {}^I\boldsymbol{\omega}(t)$ in the above equation yields (4.31).

■

Note that Lemma 1 provides a sufficient condition for the local identifiability of t_d: if (4.31) does not hold for any $\mathbf{k}_1$ and $\mathbf{k}_2$, then t_d is locally identifiable. We can in fact show that this condition is also a necessary one, by showing that if it is not met, there exists at least one indistinguishable family of solutions for t_d and the remaining unknown parameters. To this end, we start by noting that if (4.31) holds for some $\mathbf{k}_1$ and $\mathbf{k}_2$, then ${}^I\boldsymbol{\omega}(t)$ is given by:

$${}^I\boldsymbol{\omega}(t) = e^{\lfloor\mathbf{k}_2\times\rfloor t}\mathbf{k}_o + \int_0^t e^{\lfloor\mathbf{k}_2\times\rfloor(t-\tau)}d\tau \cdot \mathbf{k}_1 \tag{4.38}$$

where $\mathbf{k}_o$ is the initial value of the rotational velocity. We can now prove, by substitution in (2.28) and (4.20), that for *any* scalar δ, the sets $\{t_d, {}^C_I\mathbf{R}, \mathbf{b_g}, {}^I\boldsymbol{\omega}(t)\}$ and $\{t_d', {}^C_I\mathbf{R}', \mathbf{b_g}', {}^I\boldsymbol{\omega}'(t)\}$, where

$$t_d' = t_d + \delta \tag{4.39}$$

$${}^C_I\mathbf{R}' = {}^C_I\mathbf{R}e^{-\lfloor\mathbf{k}_2\times\rfloor\delta} \tag{4.40}$$

$$\mathbf{b_g}' = \mathbf{b_g} + \int_0^\delta e^{\lfloor\mathbf{k}_2\times\rfloor(\delta-\tau)}d\tau\mathbf{k}_1 \tag{4.41}$$

$${}^I\boldsymbol{\omega}'(t) = {}^I\boldsymbol{\omega}(t) - \int_0^\delta e^{\lfloor\mathbf{k}_2\times\rfloor(\delta-\tau)}d\tau\mathbf{k}_1 \tag{4.42}$$

yield exactly the same measurements $\boldsymbol{\omega}_m(t)$ and $\boldsymbol{\omega}_c(t)$, for all t. This means that t_d and t'_d are indistinguishable, and thus t_d is not locally identifiable.

An important question to answer is whether the cases in which t_d becomes unidentifiable are practically relevant. Examination of (4.38) shows that this general functional form for ${}^I\boldsymbol{\omega}(t)$ encompasses the cases of no rotation, when $\mathbf{k}_0 = \mathbf{k}_1 = \mathbf{0}$, and of constant rotational velocity, e.g., when $\mathbf{k}_1 = \mathbf{k}_2 = \mathbf{0}$ (in fact, in these cases the indistinguishable families contain more than one free parameters). Besides these two cases, which can potentially arise in real-world trajectories, we believe that all other degenerate situations are unlikely to occur in practice. We should note however, that the cases of zero or constant rotational velocity result in loss of observability even if the time offset between the camera and IMU is perfectly known [43, 50]. In this sense, we see that having an unknown t_d does not appear to cause loss of observability in additional, practically significant situations.

4.5.5 Identifiability of t_d based on both constraints

In the preceding section we only considered the rotational constraints provided by the camera and IMU. We now also take into account the constraints arising from the velocity and acceleration measurements, and show that the set of degenerate motion cases that lead to an unidentifiable t_d can be further restricted. Specifically, we prove the following result:

Lemma 2 *The time offset between the camera and IMU is locally identifiable, if no $\mathbf{k}_1$, $\mathbf{k}_2$, k_3, $\mathbf{k}_4$, $\mathbf{k}_5$, and $\mathbf{k}_6$ exist, such that (i) the rotational velocity of the IMU* *satisfies the differential equation* (4.31)*, and (ii) the accelerometer measurements satisfy the differential equation:*

$$\begin{aligned}\dot{\mathbf{a}}_m(t) &= \left(\lfloor \mathbf{k}_2 \times \rfloor - k_3 \mathbf{I}_3\right) \mathbf{a}_m(t) - \mathbf{k}_4 \\ &\quad - \left(\lfloor {}^{I}\boldsymbol{\omega}(t) \times \rfloor^2 + \lfloor {}^{I}\dot{\boldsymbol{\omega}}(t) \times \rfloor\right) \mathbf{k}_5 - {}^{I_o}_{I}\mathbf{R}(t)^T \mathbf{k}_6 \qquad (4.43)\end{aligned}$$

Proof. The proof of this result follows a course analogous to that of Lemma 1, and is given in Appendix C.1. ■

The above result shows that, in general, t_d is locally identifiable, and it may become unidentifiable only when *both* the orientation and the position of the platform follow specific motion patterns. As discussed, equation (4.31) yields only few critical cases for the orientation that are likely to occur in practice. Similarly, by examining the nature of the solutions to (4.43) for different choices of $\mathbf{k}_i$, we find only few cases of practical significance. The most important one is that of constant accelerometer measurements, which can arise for instance when the platform is accelerating at a constant rate with no rotation, or when it is moving at constant velocity and rotations occur only about the direction of gravity. However, these scenarios would result in loss of observability even if t_d was perfectly known. Thus, we once again see that estimating t_d online does not appear to result in any new degenerate trajectories with practical significance.

As a final remark, we point out that Lemma 2 provides a means of certifying the local identifiability of t_d only. As discussed in Section 4.5.2, the identifiability/observability of the remaining states must be checked separately. Situations can arise where t_d is locally identifiable, but some of the states O1-O5 defined in

Section 4.5.2 are not. For example, if the rotational velocity of the platform is given by ${}^{I}\boldsymbol{\omega}(t) = [\sin(t) \;\; 0 \;\; 0]^T$, t_d is locally identifiable, as ${}^{I}\boldsymbol{\omega}(t)$ does not satisfy an equation of the form (4.31). However, since the rotation happens only about the axis $[1 \;\; 0 \;\; 0]^T$, in this case the orientation of the camera with respect to the IMU is not identifiable [43, 50, 78]. Thus, as explained in Section 4.5.2, in order to certify that the entire system is locally weakly observable, one should certify that (4.31) and (4.43) do not simultaneously hold, *and* that none of the critical trajectories identified in [43, 50] occurs.

4.6 Experiments

4.6.1 Simulation tests

In this section, we present simulation results to validate the proposed algorithm, for both constant t_d and time-varying t_d.

4.6.1.1 Constant t_d

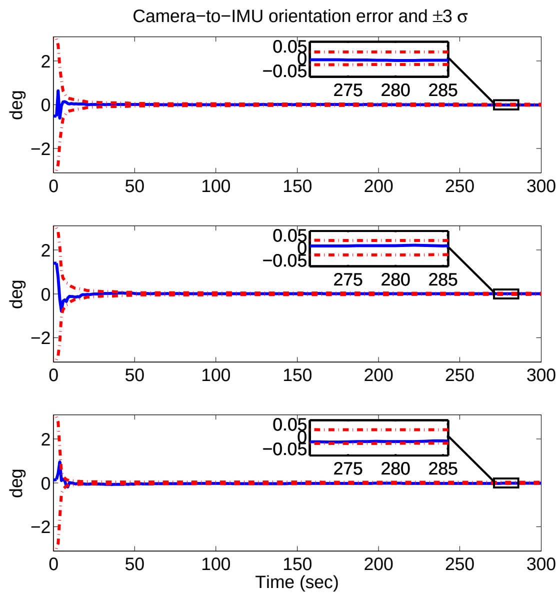
In our first simulation test, we assume that the time offset t_d is constant over time. In this test, we generated simulation environments based on a real-world dataset, which was about 5 minutes, 380 m long. The process for generating the simulation environments is presented in detail in Appendix A.6. To examine the statistical properties of our proposed algorithm, we carried out 50 Monte-Carlo trials. In each trial, the rotation and translation between the IMU and the camera were set equal to known nominal values, with the addition of random

Table 4.1: Initial uncertainties of the calibration parameters in the simulation tests.

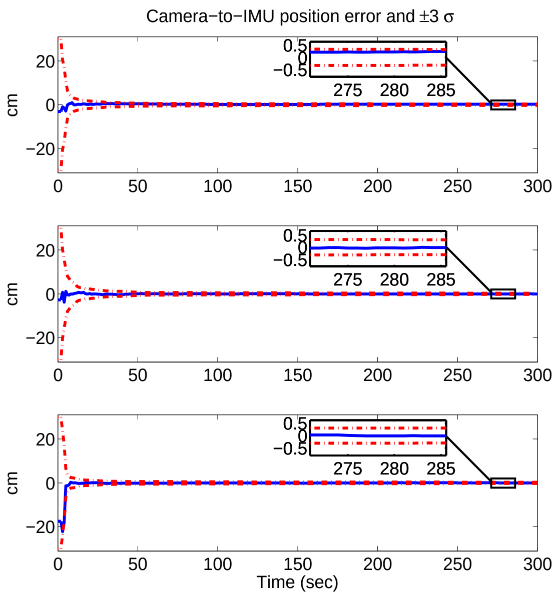
Calib. parameter	Position (σ_p)	Orientation (σ_θ)	Time offset (σ_t)
Initial uncertainty	0.1 m	1.0°	50 msec

errors $\delta\mathbf{p}$, and $\delta\boldsymbol{\phi}$. In each trial, $\delta\mathbf{p}$ and $\delta\boldsymbol{\phi}$ were randomly drawn from zero-mean Gaussian distributions with standard deviations equal to σ_p and σ_θ along each axis, respectively. In addition, t_d was randomly drawn from the Gaussian distribution $\mathcal{N}(0, \sigma_t^2)$, and kept constant for the duration of the trial. The values of σ_p, σ_θ and σ_t used in the simulations are shown in Table 4.1. Note that time offsets in the order of tens of milliseconds are typical of most systems in our experience.

In the tests presented here, we compare the performance of the proposed algorithm, in five cases: (i) t_d and $\boldsymbol{\pi}_{IC}$ estimation disabled (i.e., the "original" hybrid filter presented in Chapter 3), (ii) online $\boldsymbol{\pi}_{IC}$ estimation enabled, but t_d estimation disabled, (iii) t_d estimation enabled, but $\boldsymbol{\pi}_{IC}$ estimation disabled, (iv) t_d and $\boldsymbol{\pi}_{IC}$ estimation enabled (i.e., the proposed approach), and (v) the case where t_d and $\boldsymbol{\pi}_{IC}$ are perfectly known and not estimated. In the first four cases (termed the "imprecise" ones), the exact values of $\boldsymbol{\pi}_{IC}$ and t_d are not known (only their nominal values are known). When a particular parameter is not estimated, it is assumed to be equal to the nominal value. By comparing these four cases, we can evaluate the necessity and effectiveness of the online estimation of individ-



(a)



(b)

Figure 4.2: Motion estimation and online calibration with constant time-offset t_d: estimation errors (blue lines) and associated $\pm 3\sigma$ envelopes (red dash-dotted lines). (a) The results for the camera-to-IMU rotation, and (b) The results for the camera-to-IMU position.

ual parameters. Moreover, by comparing against case (v), where all parameters are perfectly known (the "precise" scenario), we can assess the loss of accuracy incurred due to the uncertainty in the knowledge of these parameters.

Table 4.2 shows the average RMSE and NEES for the five cases, averaged over 50 Monte-Carlo trials. For clarity, the position errors are reported in the NED (North-East-Down) frame, and IMU orientation in roll-pitch-yaw. We see that, to be able to accurately estimate the IMU's motion, both the frame transformation and the time offset between the camera and IMU must be estimated.

Table 4.2: Simulation results: The average RMSE and NEES computed by the five methods.

Scenario		imprecise				precise
${}^{C}_{I}\mathbf{T}$ estimation		off	on	off	on	N/A
t_d estimation		off	off	on	on	N/A
IMU Pose RMSE	North (m)	2.21	1.97	1.13	0.79	0.74
	East (m)	3.98	2.84	1.24	1.11	1.02
	Down (m)	1.17	0.87	0.23	0.21	0.15
	roll (°)	0.44	0.36	0.24	0.15	0.14
	pitch (°)	0.38	0.31	0.27	0.17	0.17
	yaw (°)	1.97	1.77	1.31	1.01	0.98
IMU state NEES		29.61	17.35	9.47	6.88	6.57
Calib. RMSE	${}^{C}\mathbf{p}_I$ (m)	N/A	0.057	N/A	0.003	N/A
	${}^{C}_{I}\bar{\mathbf{q}}$ (°)	N/A	0.141	N/A	0.031	N/A
	t_d (msec)	N/A	N/A	0.712	0.101	N/A
Calib. NEES	${}^{C}\mathbf{p}_I$	N/A	27.21	N/A	3.91	N/A
	${}^{C}_{I}\bar{\mathbf{q}}$	N/A	24.11	N/A	3.84	N/A
	t_d	N/A	N/A	39.07	1.77	N/A

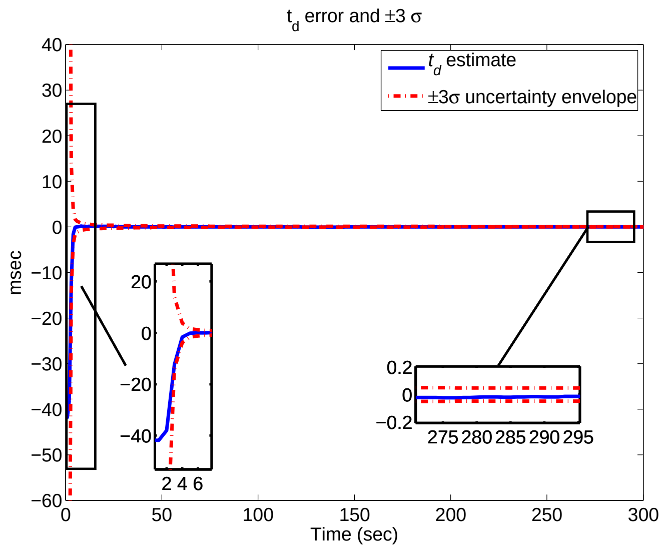


Figure 4.3: Motion estimation and online calibration with constant time-offset t_d: estimation errors for t_d (blue lines) and associated $\pm 3\sigma$ envelopes (red dash-dotted lines).

If one of these parameters or both of them are falsely assumed to be perfectly known, the estimation accuracy and consistency are considerably degraded (see the first four data columns in Table 4.2). Moreover, by comparing the fourth and fifth data columns, we can see that the accuracy obtained by our online estimation approach is very close to that obtained when both $\boldsymbol{\pi}_{IC}$ and t_d are perfectly known. This result, which is also observed in the real-world experiments (see Section 4.6.2), is significant from a practical standpoint: it shows that the proposed online approach, initialized with rough estimates, can provide pose estimates almost indistinguishable to what we would get if offline calibration was performed in advance.

Additionally, it is also interesting to examine the estimation results for the calibration parameters, when the online calibration is fully enabled (i.e., the proposed approach). In Fig. 4.2 and Fig. 4.3, we present the estimation errors as well as the corresponding $\pm 3\sigma$ uncertainty envelopes of these parameters in one representative trial, and in Table 4.2 (last six data rows of the fourth data column) we show the estimation statistics averaged over all Monte-Carlo simulation trials.

From Fig. 4.2 and Fig. 4.3, we can clearly observe that, for both the spatial and temporal calibration parameters, the estimation errors and the uncertainty curves reduce rapidly, which demonstrates that these parameters can be accurately calibrated in a short time. This result is also validated by the RMSE values in Table 4.2, where it is shown that the RMS errors of the relative position, orientation, and time offset are only 0.003 m, 0.031°, and 0.101 msec, which are substantially smaller than their initial uncertainties 0.1 m, 1°, and 50 msec respectively (see Table 4.1).

4.6.1.2 Time-varying t_d

In the previous simulation test, a constant time offset was used. We here also examine the case of a time-varying t_d. Instead of presenting Monte-Carlo simulation results (which look similar to those in Table 4.2), it is interesting to show the results of a single representative trial. In this trial, the time offset varies linearly from 20 msec at the start, to 320 msec at after 300 sec, modelling a severe clock drift of 0.3 sec in 5 minutes. Fig. 4.4 presents the estimation errors and

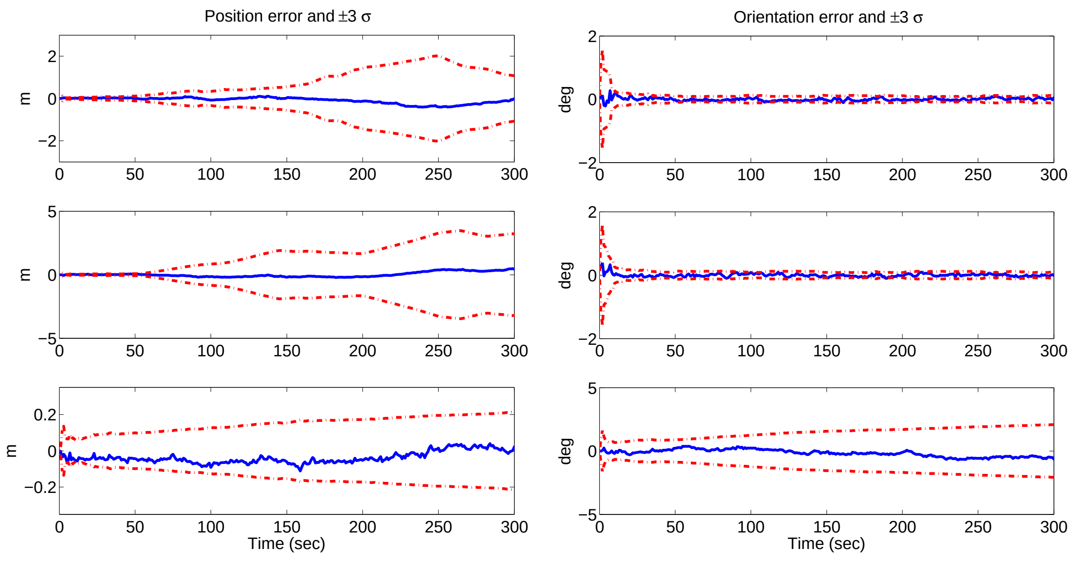


Figure 4.4: Motion estimation and online calibration with drifting time-offset: estimation errors (blue lines) and associated $\pm 3\sigma$ envelopes (red dash-dotted lines). (a) The IMU position errors and (b) The IMU orientation errors.

associated ± 3 standard deviations for the IMU position, the IMU orientation, while Fig. 4.5 presents the results for the time offset. We can see that even in this challenging situation (large and time-varying offset) the estimates remain consistent. We stress that this is due to the identifiability of t_d, which allows us to track its value closely as it drifts over time.

4.6.2 Real-world experiment

In addition to the simulation tests, we also carried out a real-world experiment, involving a car driven in the downtown area of Riverside, CA, covering approximately 5.0 km in 20 minutes. In this experiment, a sensor suite was mounted

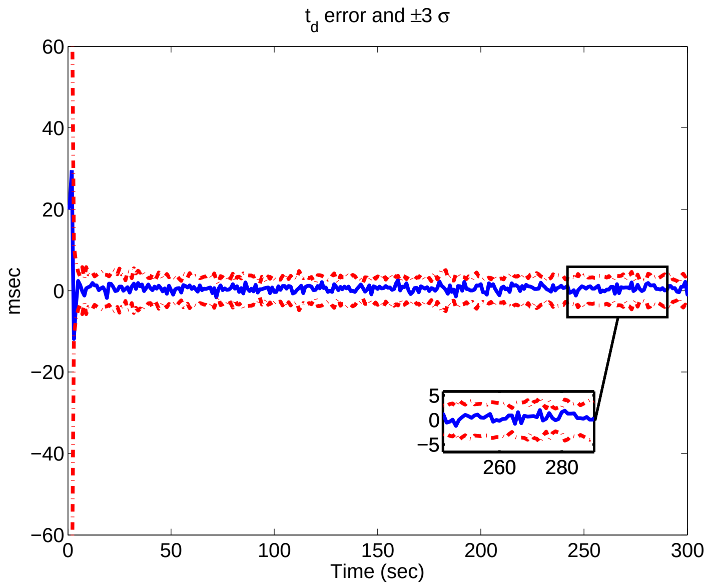


Figure 4.5: Motion estimation and online calibration with drifting time-offset t_d: estimation errors for t_d (blue lines) and associated $\pm 3\sigma$ envelopes (red dash-dotted lines).

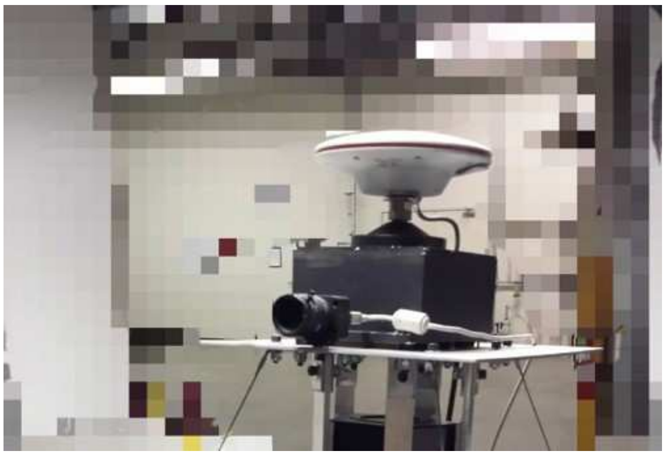


Figure 4.6: The sensor suite used for real-world experiments.

Table 4.3: Experiment results: The largest position errors of the three methods.

Largest position errors		
Online calib. t_d and $\boldsymbol{\pi}_{IC}$	No calib. t_d and $\boldsymbol{\pi}_{IC}$	Known t_d and $\boldsymbol{\pi}_{IC}$
20.83 m	37.18 m	18.39 m

on top of a car, consisting of a Point Grey MV-FMVU-03MTC camera, a NAV NV-IMU1000 IMU, and a NovAtel OEMV3 GPS receiver (see Fig. 4.6). The IMU reported inertial measurements at 200 Hz, the camera captured images at 15 Hz, and the GPS receiver provided measurements at 1 Hz. In this experiment, the measurements of the IMU and camera were used to perform concurrent motion estimation and online calibration, using the proposed VIO estimator. On the other hand, the readings of the IMU and GPS were fused together to provide an approximate "ground truth" trajectory, computed by employing a state-of-the-art GPS-aided inertial navigation algorithm [126]. To process the recorded images, we used the Shi-Tomasi algorithm to extract corner features and normalized cross-correlation to match these features.

In this experiment, we tested three VIO algorithms: (i) the proposed algorithm that performs concurrent motion estimation and online calibration, (ii) the hybrid estimator using the final estimate of t_d and $\boldsymbol{\pi}_{IC}$ as known inputs, and (iii) the original hybrid estimator that does not estimate t_d and $\boldsymbol{\pi}_{IC}$ (for $\boldsymbol{\pi}_{IC}$ manual measurements were used, and $t_d = 0$ was assumed in this case). To demonstrate the estimation results, we plot the trajectory estimates of these three algorithms

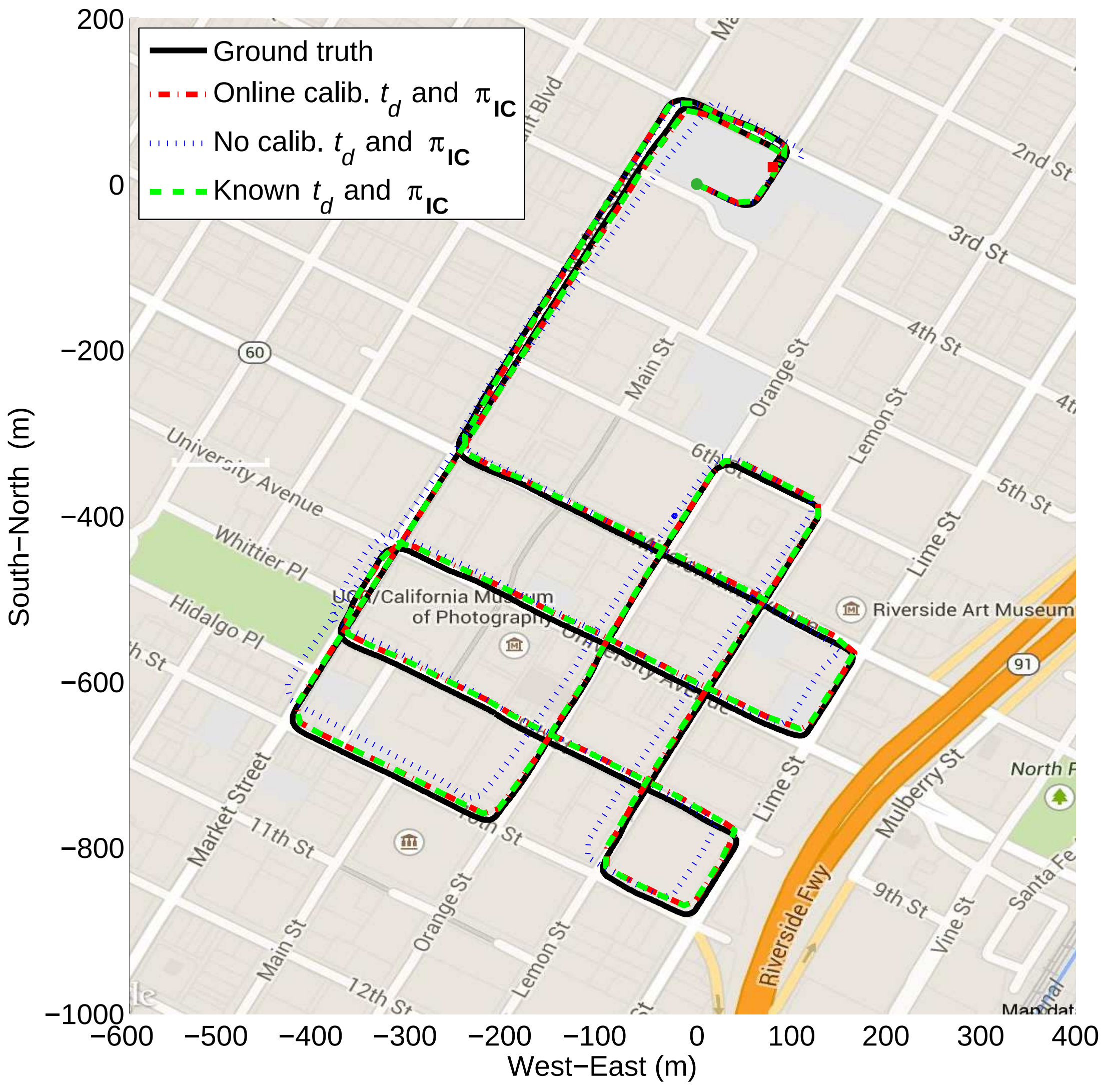


Figure 4.7: Experiment results: The trajectory estimates computed by the three methods, plotted vs. ground truth on a map of the area. The green circle represents the starting point of the trajectory, and the red square denotes the ending point.

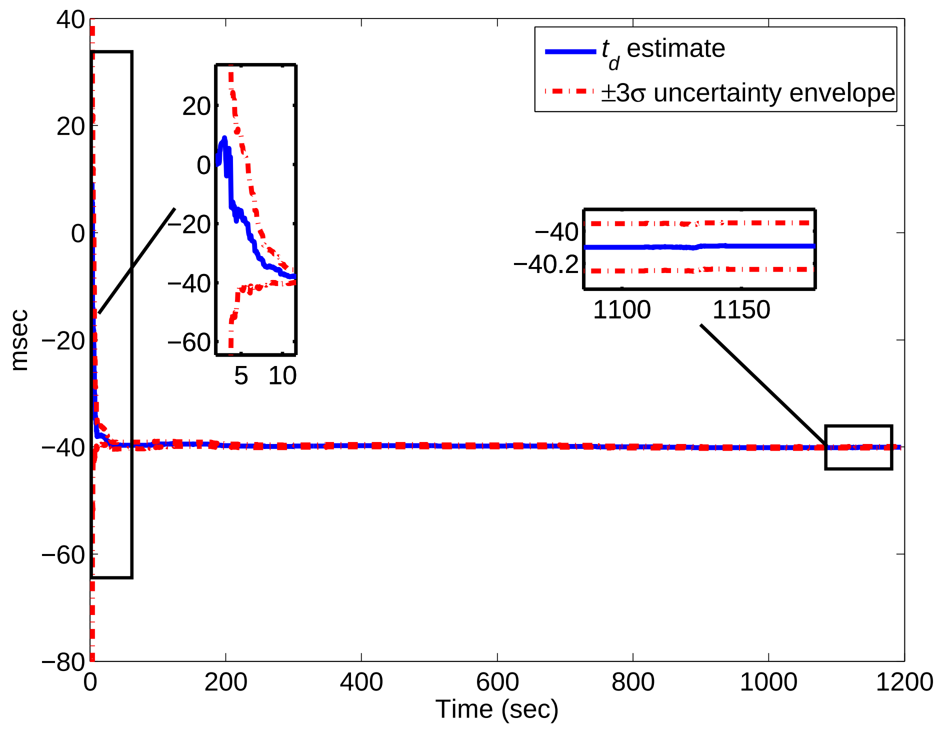


Figure 4.8: Time offset estimate and corresponding $\pm 3\sigma$ uncertainty envelope computed from the covariance matrix reported by the estimator.

as well as the approximate ground truth trajectory in Fig. 4.7. Additionally, we calculated the largest position errors for the three algorithms, over the entire trajectory, and present the results in Table 4.3.

From Fig. 4.7 and Table 4.3, we can observe that the proposed algorithm, which performs concurrent motion estimation and online calibration, outperforms the original hybrid estimator by a wide margin. These results are similar to those obtained in the simulation tests, and demonstrate that, by estimating the sensor-to-sensor parameters online, the motion estimation performance can be substantially improved. In addition, we note that the proposed method yields results that are almost identical to those we would obtain if t_d and $\boldsymbol{\pi}_{IC}$ were known in advance. We thus see that using the proposed online approach for determining sensor-to-sensor parameters, instead of an offline calibration procedure, would result in no significant loss of accuracy.

Additionally, in Fig. 4.8, we plot the estimate of the time offset t_d, as well as the uncertainty envelope defined by ± 3 times the standard deviation reported by the proposed EKF. We can see that within the first few seconds the estimate converges very close to its final value, and that the uncertainty in the estimate drops rapidly. We also point out that the standard deviation of t_d over the last one minute of the experiment is only 0.05 ms, which shows the high precision attainable by the proposed algorithm.

4.7 Conclusion

In this chapter, we have proposed an online approach for estimating the time offset, t_d, and the spatial configuration, $\boldsymbol{\pi}_{IC}$, between the camera and IMU during visual-inertial odometry. The key component of our formulation is that the variables t_d and $\boldsymbol{\pi}_{IC}$ are explicitly included in the EKF state vector, and estimated jointly with all other variables of interest. This makes it possible to track both t_d and $\boldsymbol{\pi}_{IC}$, to characterize the uncertainties in their estimates, and to model the effect of the imperfect knowledge of t_d and $\boldsymbol{\pi}_{IC}$ on the accuracy of the estimates, in a natural way. Moveover, we have shown that t_d is identifiable in general trajectories, which guarantees the effectiveness of the proposed online estimation approach. A detailed characterization of the critical motion cases that lead to loss of identifiability of t_d reveals that they are either (i) cases that are known to cause loss of observability even with a perfectly known t_d, or (ii) cases that are unlikely to occur in practice. Our simulation and experimental results indicate that the proposed approach leads to high-precision estimates for both the system motion, as well as for the temporal and spatial alignment between the camera and IMU.

Chapter 5

Real-time Visual-Inertial Odometry and Self-Calibration Using Low-Cost Sensors

5.1 Introduction

In the preceding chapters, we have presented algorithms for performing VIO with minimum computational resources. These algorithms are applicable in any system, regardless of whether it is equipped with high-end or low-cost sensors. In the latter case, however, there exist qualitative differences in the nature of the sensor data, which must be addressed in order to achieve high-precision localization. This is the focus of this chapter.

Specifically, a key characteristic of low-cost sensors is that they often exhibit significant, systematic errors, which must be modeled and compensated for. On the one hand, lost-cost MEMS IMUs have non-negligible misalignment between the sensor axes, non-unit scale factors, and g-sensitivity, which are typically not a major problem in high-end sensors. On the other hand, low-cost cameras can have significant lens distortion, and typically employ rolling-shutter (RS) sensors. In contrast to a global-shutter (GS) camera, in which all the pixels in an image are captured simultaneously, a RS camera captures the image rows sequentially over a time interval of non-zero duration (called the image readout time[1]). Therefore, when a RS camera is moving, each row of pixels is captured from a different camera pose. This can create significant image distortion (see Fig. 5.1), and must be modelled. To address these challenges, in this chapter we propose systematic approaches for modeling and utilizing the measurements from both low-cost IMUs and RS cameras.

The first main contribution in this chapter is an algorithm for performing joint motion estimation and online calibration of visual and inertial sensors. Specifically, we employ high-fidelity sensor models that are able to accurately characterize the sensor measurements from both low-cost IMUs and RS cameras, and perform online estimation of *all* the parameters in these models, by including them in the state vector of the estimator presented in Chapter 3. The sensor characteristics being calibrated online include

- the IMU biases
- the misalignment and scale factors of the IMU sensors

[1]Note that alternative terms have been used to describe this time interval (e.g., shutter roll-on time [70] and line delay [89]). We here adopt the terminology defined in [93].

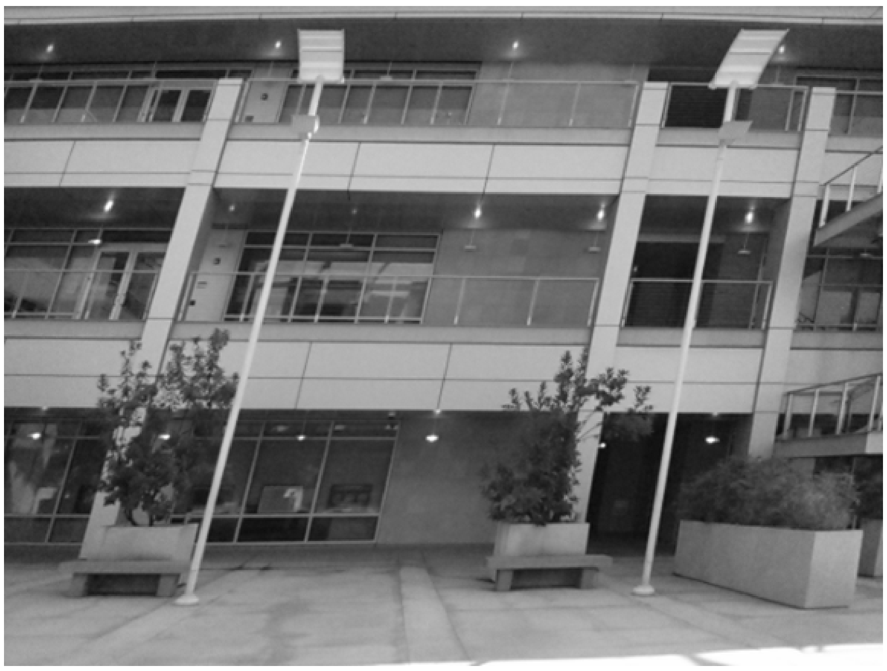


Figure 5.1: An example image with rolling-shutter distortion.

- the acceleration dependence (typically called g-sensitivity) of the gyroscope measurements
- the camera-to-IMU spatial configuration
- the camera intrinsic parameters, including lens distortion
- the image readout time of the rolling-shutter camera
- the time offset between the timestamps of the camera and the IMU

Performing online calibration of these quantities has a number of advantages. First, it removes the need for offline calibration, which is an often tedious process that should be repeated periodically as mechanical shocks and other factors can lead to changes in the parameters over time. Second, we note that, even if high-quality offline calibration is performed, the existence of some uncertainty in the calibration parameters is inevitable. If this uncertainty is not modelled, it will degrade the performance of VIO. The proposed online calibration approach addresses this problem, since it explicitly models the uncertainty in the estimates of the calibration parameters, and accounts for its effect in the computed motion estimates. As a result, we obtain both more accurate motion estimates and a better characterization of their uncertainty.

The second main contribution in this chapter is an efficient approach for processing the measurements from RS cameras in vision-aided inertial navigation. We note that, due to the fact that image rows in a RS camera are captured at slightly different time instants, they correspond to different camera poses. Since including in the estimator one state for each such pose (the "exact" approach) is computationally intractable, all existing methods employ some assumption about the nature of the camera trajectory (see, e.g., [37, 61, 89]). By contrast, our approach employs *no* assumptions on the form of the camera trajectory itself, and is thus able to model arbitrarily complex motions. Instead, it uses an approximate representation for the time-evolution of the estimation *errors* during the image readout time. Since these errors are typically small, this leads to only small modeling inaccuracies. Moreover, since the statistical properties of the errors are known in advance, we can compute upper bounds on the worst-case magnitude of these modeling errors and use the bounds to guide the selection of the appropriate error representation. We demonstrate that, in practice, a very low-dimensional representation suffices, and thus the computational cost can be kept low, almost identical to that needed for processing measurements from a GS camera.

We incorporate the high-fidelity sensor modeling and self-calibration approach, as well as our novel approach for the processing of RS measurements, into the MSCKF/SLAM hybrid estimator, and present an overall algorithm for visual-inertial odometry on resource-constrained systems. Through both Monte-Carlo simulations and real-world experiments we demonstrate that the resulting estimator is able to compute high-precision, consistent motion estimates in real time, on various resource-constrained systems.

5.2 Related Work

As mentioned in the previous section, there are two main contributions in this chapter: (i) high-fidelity modeling and self-calibration of low-cost visual and inertial sensors, and (ii) an algorithm for using the measurements from RS cameras in vision-aided inertial navigation. Thus, in this section, we discuss the related literature of these two topics separately.

5.2.1 Self-calibration of visual and inertial sensors

To the best of our knowledge, the topic of joint self-calibration of visual and inertial sensors has not been addressed in the past. In what follows, we discuss relevant approaches, divided into three categories:

a) Camera calibration: In the vast majority of cases, real-time vision-based localization algorithms assume that the camera's intrinsic parameters are known in advance, e.g., via an offline calibration (see [36] for a comprehensive review of calibration methods). Camera self-calibration is also possible. In [17] the authors present an online approach to self-calibration, which employs a Sum-of-Gaussians filter for visual SLAM. However, this is a vision-only approach, and thus IMU calibration is not addressed.

For a rolling-shutter camera, in addition to the geometry of the camera's projection model (e.g., focal length, principal point, lens distortion) one must also know the image readout time. Calibrating this readout time is a much less studied issue. The approaches presented to date are offline ones, and require knowledge about the properties of the scene (e.g., a known calibration pattern in [89], or a LED flashing at a known frequency in [77]), a special motion of the camera [47], or a combination thereof. Moreover, these approaches require the camera's projection geometry to be perfectly known. By contrast, in our work both the camera geometry and the readout time are jointly estimated online.

b) IMU calibration: Gyroscope and accelerometer measurements are typically affected by biases, which are slowly changing over time. The majority of high-precision vision-aided inertial navigation algorithms model these biases, and estimate them online along with the system motion (see., e.g., [43, 50, 83] and references therein). However, the misalignment of the IMU axes, IMU scale factors, and the g-sensitivity of the gyroscope measurements are typically not estimated online. Most often these effects are assumed to be negligible (realistic for high-end, expensive, sensors), while offline calibration methods have also been employed [13, 105]. Our approach removes the need for offline calibration, by estimating all the above systematic errors online.

We note that methods for high-precision IMU calibration using a camera are presented in [59,124]. In these works, a camera with known intrinsic parameters is used, in conjunction with a known calibration pattern. The visual measurements of known landmarks are then used in bundle adjustment [59] or in an EKF [124], to estimate the IMU characteristics as well as the transformation between the camera and IMU frames. Compared to these methods, the self-calibration approach we describe jointly estimates the IMU *and* camera parameters, *without* known scene structure.

5.2.2 Using measurements from RS cameras

Most work on RS cameras has focused on the problem of image rectification, for compensating the visual distortions caused by the rolling shutter. These methods estimate a parametric representation of the distortion from the images, which is then used to generate a rectified video stream (see, e.g., [28,68] and references therein). Gyroscope measurements have also been employed for distortion compensation, since the most visually significant distortions are caused by rotational motion [34,42,47]. In contrast to these methods, our goal is not to undistort the images (which is primarily done to create visually appealing videos), but rather to use the recorded images for motion estimation.

One possible approach to this problem is to employ estimates of the camera motion in order to "correct" the projections of the features in the images, and to subsequently treat the measurements as if they were recorded by a GS sensor. This approach is followed in [53], which describes an implementation of the well-known

PTAM algorithm [52] using a RS camera. The feature projections are corrected by assuming that the camera moves at a constant velocity, estimated from image measurements in a separate least-squares process. Similarly, image corrections are applied in the structure-from-motion method of [38], in which only the rotational motion of the camera is compensated for. In both cases, the "correction" of the image features' coordinates entails approximations, which are impossible to avoid in practice: for completely removing the RS effects, the points' 3D coordinates as well as the camera motion would have to be perfectly known.

The approximations inherent in the approaches described above are circumvented in methods that include a representation of the camera's motion in the states to be estimated. This makes it possible to explicitly model the motion in the measurement equations, and thus no separate "correction" step is required. All such methods to date employ low-dimensional parameterizations of the motion, for mathematical simplicity and to allow for efficient implementation. For example, in [2, 3, 74] a constant-velocity model is used to enable the estimation of an object's motion. In [37], a RS bundle-adjustment method is presented, which uses a constant-velocity model for the position and SLERP interpolation for the orientation. As mentioned in Section 5.1, however, these simple representations of the camera trajectory introduce modeling inaccuracies, which can be significant if the motion is not smooth.

To demonstrate this, in Fig. 5.2(left) we plot the rotational velocity, $\boldsymbol{\omega}_m$, and acceleration, $\mathbf{a}_m$, measured by the IMU on a Nexus 4 device during one of our experiments. The plots show a one-second-long window of data, recorded while

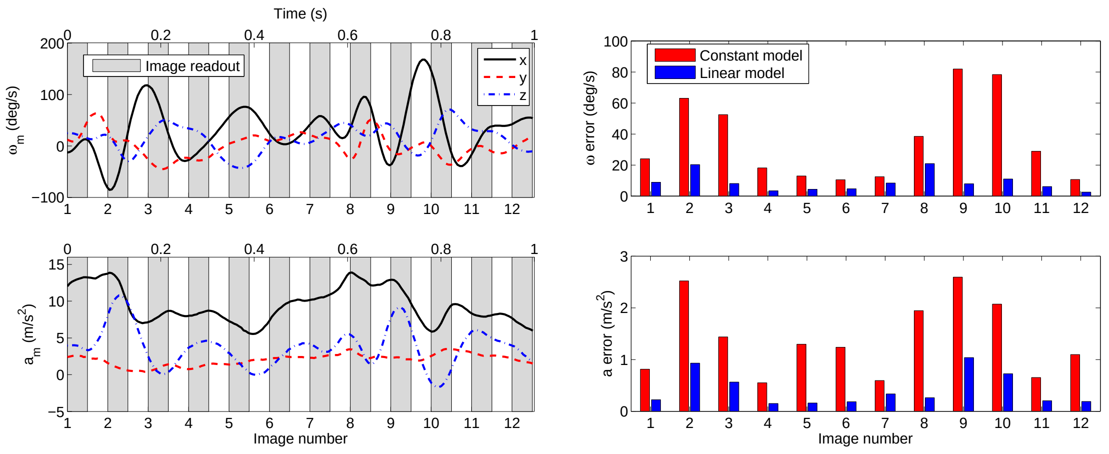


Figure 5.2: Left: The rotational velocity (top) and acceleration (bottom) measurements recorded during a one-second period in one of our experiments. Right: The largest modeling errors incurred in each readout interval when using the best-fit constant (red) and linear (blue) representations of the signals.

the device was held by a person walking at normal pace. In this time interval, 12 images were captured, each with a readout time of 43.3 ms. From the plots of $\boldsymbol{\omega}_m$ and $\mathbf{a}_m$, it becomes clear that the device's motion is changing rapidly, and thus low-dimensional motion representations will lead to significant inaccuracies. To quantify the modeling inaccuracies, in Fig. 5.2(right) we plot the largest absolute difference between the signals and their best-fit constant and linear approximations in each readout interval. Clearly, the approximations are quite poor, especially for the rotational velocity. The modeling errors of the constant-velocity model for $\boldsymbol{\omega}_m$ (employed, e.g., in [61]) reach 81.8 ^{o}/s. Even if a linear approximation of $\boldsymbol{\omega}_m$ were to be used, the modeling inaccuracies would reach 20.9 ^{o}/s. We point out that, due to their small weight, resource-constrained devices such as mobile phones or MAVs typically exhibit highly-dynamic motion profiles, such as the one seen in Fig. 5.2.

Since in this work we focus on performing real-time VIO on resource-constrained devices, the methods that use low-dimensional motion parameterizations can be of limited utility.

An elegant approach to the problem of motion parameterization in vision-based localization is offered by the continuous-time formulation originally proposed in [30]. This formulation has recently been employed for pose estimation and RS camera calibration in [89], and for visual-inertial SLAM with RS cameras in [69]. A similar approach has also been used in [11,12], to model the trajectory for 2D laser-scanner based navigation. The key idea of the continuous-time formulation is to use a weighted sum of temporal basis functions (TBF) to model the motion. This approach offers the advantage that, by increasing the number of basis functions, one can model arbitrarily complex trajectories. Thus, highly-dynamic motion profiles can be accommodated, but this comes at the cost of an increase in the number of states that need to be estimated (see Section 5.4.1 for a quantitative analysis). The increased state dimension is not a significant obstacle if offline estimation is performed (which is the case in the aforementioned approaches), but is undesirable in real-time applications. Similar limitations exist in the Gaussian-Process-based representation of the state, described in [111].

We stress that, with the exception of the continuous-time formulation of [69], all the RS motion-estimation methods discussed above are *vision-only* methods. To the best of our knowledge, the only work to date that presents large-scale, real-time localization with a RS camera and an IMU is our previous work in [61]. The limitations of that work, however, stem from its use of a constant-velocity model for the motion during the image readout period. This reduces accuracy when the system undergoes significant accelerations, but also increases computational requirements, as both the linear and rotational velocity at the time of image capture must be included in the state vector. In the experimental results presented in Section 5.6, we show that the novel formulation for using the RS measurements presented here outperforms [61], both in terms of accuracy and computational efficiency.

5.2.3 Organization

The remainder of this chapter is organized as follows. In Section 5.3, we describe the high-fidelity models for the measurements from low-cost IMUs and RS cameras. Subsequently, we present an efficient algorithm for processing the RS measurements in Section 5.4. In Section 5.5, we present an EKF algorithm to perform real-time VIO, by employing the proposed approaches in the hybrid EKF. Finally, Sections 5.6-5.7 demonstrate the simulation and the experimental results, and Section 5.8 concludes this chapter.

5.3 Sensor Modeling

5.3.1 IMU model

In Section 2.5.1, we presented measurement models for the IMU's gyroscopes and accelerometers (see (2.28)-(2.29)). While these models are valid for typical high-end sensors, they can be insufficient for low-cost IMUs, which are commonly affected by additional systematic error terms. In our work, to model these extra terms, we employ a sensor model that characterizes all the systematic errors that can be modeled linearly and have the most impact on precision. For a full description of the sources of errors and their modeling, the reader is referred to [110].

To begin with, we note that, due to the imprecision of the manufacturing process of low-cost IMUs, the three axes of the accelerometers are typically affected by sensor misalignment, i.e., the three axes are not mutually perpendicular to each other. As a result, our previous definition of the IMU coordinate frame $\{I\}$, which relies on the directions of the three accelerometer axes (see Section 2.3), cannot be used in this case, and we must re-define the IMU frame $\{I\}$. For the origin of $\{I\}$, we choose the point at which the three accelerometer axes intersect, similarly to Section 2.3. On the other hand, we select the orientation of $\{I\}$ based on the orientation of the camera frame, $\{C\}$. Specifically, we *define* the rotation matrix of $\{I\}$ with respect to $\{C\}$ to be a known, constant matrix[2] ${}^{C}_{I}\mathbf{R}$, whose value can be selected at will. For instance, it can be chosen as the identity matrix for simplicity, or it can be set equal to a prior estimate of the camera-to-IMU rotation. In our work we use the latter option.

[2]Defining the orientation of the IMU frame in this way is necessary because the precise directions of the accelerometer are not known, and must be estimated. If we chose the orientation of $\{I\}$ independently of the camera, we would need to estimate both the accelerometer' directions and the orientation of the $\{C\}$ with respect to $\{I\}$. This can be shown to introduce unidentifiable parameters in the system model. Alternative choices do exist, such as aligning one axis of $\{I\}$ with one axis of the accelerometer [59]. However, that leads to more complex measurement models.

To describe the IMU measurement models, we first consider the accelerometer. Each of the three accelerometer sensors in an IMU provides scalar measurements of specific force, modelled as:

$$a_{m_i} = s_i {}^I\mathbf{u}_i^T \, {}^I\mathbf{a}_s + \mathbf{b}_{\mathbf{a}_i} + \mathbf{n}_{\mathbf{a}_i} \qquad i = 1, 2, 3 \tag{5.1}$$

where ${}^I\mathbf{u}_i$ is a unit vector along the sensing direction, s_i is a scale factor close to unity, b_{a_i} is a bias, n_{a_i} is random measurement noise, and ${}^I\mathbf{a}_s$ is the specific-force vector:

$${}^I\mathbf{a}_s = {}^I_G\mathbf{R}({}^G\mathbf{a}_I - {}^G\mathbf{g})$$

Stacking the measurements of the three accelerometer sensors, we obtain the 3×1 vector:

$$\mathbf{a}_m = \mathbf{T}_a \, {}^I\mathbf{a}_s + \mathbf{b}_\mathbf{a} + \mathbf{n}_\mathbf{a} \tag{5.2}$$

where $\mathbf{T}_a$ is a 3×3 matrix whose i-th row is $s_i {}^I\mathbf{u}_i^T$, and $\mathbf{b}_a$ and $\mathbf{n}_a$ are vectors with elements b_{a_i} and n_{a_i}, respectively.

The gyroscope measurements are modelled as:

$$\boldsymbol{\omega}_m = \mathbf{T}_g {}^I\boldsymbol{\omega} + \mathbf{T}_s {}^I\mathbf{a}_s + \mathbf{b}_\mathbf{g} + \mathbf{n}_\mathbf{r} \tag{5.3}$$

where $\mathbf{T}_g$ and $\mathbf{T}_s$ are 3×3 matrices, $\mathbf{b}_\mathbf{g}$ is the measurement bias, and $\mathbf{n}_\mathbf{r}$ the measurement noise. Similarly to the case of the accelerometer measurements, $\mathbf{T}_g$ arises due to the scale factors in the gyroscope measurements and the misalignment of the gyroscope sensors to the principal axes of $\{I\}$. On the other hand, the matrix $\mathbf{T}_s$ represents the acceleration dependence (g-sensitivity) of the measurements, which can be significant for low-cost MEMS sensors [110].

It is important to note that the matrices $\mathbf{T}_a$ and $\mathbf{T}_g$ in (5.2)-(5.3) model the IMU sensors' scale factors, the sensors' misalignment, *and* their direction with respect to the camera frame. Specifically, the scale factors of the accelerometer and gyroscope sensors are given by the norm of the rows of $\mathbf{T}_a$ and $\mathbf{T}_g$, respectively, while the sensors' direction in $\{I\}$ is defined by the unit vector corresponding to each row (see (5.1)). Knowing these unit vectors makes it possible to estimate the misalignment of the sensors. Moreover, since the rotation between the camera frame and $\{I\}$ is by definition a known constant, these unit vectors also provide us with the direction of the IMU sensors with respect to $\{C\}$.

5.3.2 Rolling-shutter camera model

We here present the measurement model of the RS camera. The defining characteristic of a RS camera is that it captures the rows of an image over an interval of duration t_r (the readout time). If the image has N rows, then the time instants these rows are captured are given by:

$$t_n = t_o + \frac{nt_r}{N}, \qquad n \in \left[-\frac{N}{2}, \frac{N}{2}\right] \tag{5.4}$$

where t_o is the midpoint of the image readout interval.

If a feature with position ${}^G\mathbf{p}_f$ is observed at a location that is n rows from the middle, its image coordinates are described by the measurement model:

$$\mathbf{z} = \mathbf{h}({}^C\mathbf{p}_f(t_n)) + \mathbf{n} \tag{5.5}$$

where

$${}^C\mathbf{p}_f(t_n) = {}^C_I\mathbf{R}\,{}^I_G\mathbf{R}(t_n)\left({}^G\mathbf{p}_f - {}^G\mathbf{p}_I(t_n)\right) + {}^C\mathbf{p}_I \tag{5.6}$$

where ${}^{C}\mathbf{p}_f(t_n) = [{}^{C}x_f(t_n), {}^{C}y_f(t_n), {}^{C}z_f(t_n)]^T$ is the position of the feature with respect to the camera frame at time t_n, $\mathbf{n}$ is the measurement noise vector, and $\mathbf{h}(\cdot)$ is the camera's projection function. The measurement function (5.5) is similar to the corresponding function of a global shutter camera (2.7)-(2.8). However, we note that, for features that are captured at different image rows, the time instants t_n are different. As a result, the corresponding poses ${}^{I}_{G}\mathbf{R}(t_n)$ and ${}^{G}\mathbf{p}_I(t_n)$ are also different, which is not the case in a GS camera. It is also important to note that, since we are interested in calibrating the intrinsic parameters of the RS camera, we here employ the "raw" pixel measurements instead of normalized measurements. Therefore, $\mathbf{h}(\cdot)$ is defined as:

$$\mathbf{h}({}^{C}\mathbf{p}_f(t_n)) = \begin{bmatrix} x_{\text{pixel}} \\ y_{\text{pixel}} \end{bmatrix} = \mathbf{p}_c + \begin{bmatrix} a_u & 0 \\ 0 & a_v \end{bmatrix} \begin{bmatrix} u_d \\ v_d \end{bmatrix} \tag{5.7}$$

where $\mathbf{p}_c$ is the pixel location of the principal point, (a_u, a_v) represent the camera focal length measured in horizontal and vertical pixel units, and (u_d, v_d) are defined as:

$$\begin{bmatrix} u_d \\ v_d \end{bmatrix} = d \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} 2t_1uv + t_2(u^2 + v^2 + 2uv) \\ t_1(u^2 + v^2 + 2uv) + 2t_2uv \end{bmatrix} \tag{5.8}$$

$$d = 1 + k_1(u^2 + v^2) + k_2(u^2 + v^2)^2 + k_3(u^2 + v^2)^3 \tag{5.9}$$

$$\begin{bmatrix} u \\ v \end{bmatrix} = \frac{1}{{}^{C}z_f} \begin{bmatrix} {}^{C}x_f \\ {}^{C}y_f \end{bmatrix} \tag{5.10}$$

In the above, k_i, $i = 1, 2, 3$ are the radial distortion coefficients, while t_1, t_2 are the tangential distortion coefficients. The fact that the intrinsic parameters of the camera appear in $\mathbf{h}(\cdot)$ allows us to compute Jacobians of the measurements with respect to these parameters in the EKF, as needed for their estimation.

5.4 Rolling-shutter Modeling

The main difficulty in performing VIO using a RS camera is that, as explained, feature measurements in different rows of a RS image are captured at slightly different instants. To process these measurements, the direct solution would be to include in the estimator one camera pose per image row, which is computationally intractable. Instead, the goal of practical formulations for pose estimation using RS measurements is to include in the estimator only a small number of states per image, while keeping the model inaccuracies small. Following this paradigm, we here describe an efficient method for processing RS camera measurements, which can be employed in any linearization-based estimator. Subsequently, in the next section, we show how this method is integrated into our proposed hybrid EKF to perform real-time VIO.

We start by noting that the measurement function of the RS camera (5.5) can be re-written as:

$$\mathbf{z} = \begin{bmatrix} x_{\text{pixel}} \\ y_{\text{pixel}} \end{bmatrix} = \mathbf{h}\left({}^{I}_{G}\mathbf{q}(t_n), {}^{G}\mathbf{p}_I(t_n), \mathbf{x}_a\right) + \mathbf{n} \tag{5.11}$$

where $\mathbf{x}_a$ is a vector consisting of the feature position, ${}^{G}\mathbf{p}_f$, and all the calibration parameters (described in Section 5.5).

In any linearization-based estimator, the processing of the measurement in (5.11) is based on computing the associated residual, defined by:

$$\tilde{\mathbf{z}} = \mathbf{z} - \mathbf{h}\left({}^{I}_{G}\hat{\mathbf{q}}(t_n), {}^{G}\hat{\mathbf{p}}_I(t_n), \hat{\mathbf{x}}_a\right) \tag{5.12}$$

In this expression $\mathbf{z}$ is the actual camera measurement, the estimates $\hat{\mathbf{x}}_a$ are available in the estimator's state vector, and t_n can be calculated from $\mathbf{z}$, using (5.4) with $n = x_{\text{pixel}} - N/2$. Thus the only "missing" part in computing the residual $\tilde{\mathbf{z}}$ in (5.12) is the estimate of the IMU pose, $\left({}^I_G\hat{\mathbf{q}}(t_n), {}^G\hat{\mathbf{p}}_I(t_n)\right)$, at time t_n. In our approach, we compute this by utilizing the IMU measurements. Specifically, we include in the state vector the estimates of the IMU state at t_o, and compute ${}^G\hat{\mathbf{p}}_I(t_n)$ and ${}^I_G\hat{\mathbf{q}}(t_n)$, $n \in [-N/2, N/2]$, by integrating the IMU measurements in the readout time interval.

In addition to computing the residual, linearization-based estimators require a linear (linearized) expression relating the residual in (5.12) to the errors of the state estimates. To obtain such an expression, we begin by directly linearizing the camera observation model in (5.11), which yields:

$$\tilde{\mathbf{z}} \simeq \mathbf{H}_{\boldsymbol{\theta}} {}^G\tilde{\boldsymbol{\theta}}_I(t_n) + \mathbf{H}_{\mathbf{p}} {}^G\tilde{\mathbf{p}}_I(t_n) + \mathbf{H}_a \tilde{\mathbf{x}}_a + \mathbf{n} \tag{5.13}$$

where $\mathbf{H}_{\boldsymbol{\theta}}$ and $\mathbf{H}_{\mathbf{p}}$ are the Jacobians of the measurement function with respect to the IMU orientation and position at time t_n, and $\mathbf{H}_a$ is the Jacobian with respect to $\mathbf{x}_a$. Since the state at t_n is not in the estimator's state vector, we cannot directly employ the expression in (5.13) to perform an update – what is needed is an expression relating the residual to quantities at t_o, which do appear in the state. To obtain such an expression, we start with the Taylor-series expansions:

$$ {}^G\tilde{\mathbf{p}}_I(t_n) = \sum_{i=0}^{\infty} \frac{(nt_r)^i}{N^i i!} \; {}^G\tilde{\mathbf{p}}_I^{(i)}(t_o) \tag{5.14}$$

$$ {}^G\tilde{\boldsymbol{\theta}}_I(t_n) = \sum_{i=0}^{\infty} \frac{(nt_r)^i}{N^i i!} \; {}^G\tilde{\boldsymbol{\theta}}_I^{(i)}(t_o) \tag{5.15}$$

The above expressions are exact, but are not practically useful, as they contain an infinite number of terms, which cannot be included in an estimator. We therefore truncate the two series to a finite number of terms:

$$ {}^G\tilde{\mathbf{p}}_I(t_n) \simeq \sum_{i=0}^{l_p} \frac{(nt_r)^i}{N^i i!} \; {}^G\tilde{\mathbf{p}}_I^{(i)}(t_o) \tag{5.16} $$

$$ {}^G\tilde{\boldsymbol{\theta}}_I(t_n) \simeq \sum_{i=0}^{l_\theta} \frac{(nt_r)^i}{N^i i!} \; {}^G\tilde{\boldsymbol{\theta}}_I^{(i)}(t_o) \tag{5.17} $$

where l_p and l_θ are the chosen truncation orders for the position and orientation, respectively. Substitution in (5.13) yields:

$$ \mathbf{r} \simeq \sum_{i=0}^{l_\theta} \frac{(nt_r)^i \mathbf{H}_{\boldsymbol{\theta}}}{N^i i!} \; {}^G\tilde{\boldsymbol{\theta}}_I^{(i)}(t_o) + \sum_{i=0}^{l_p} \frac{(nt_r)^i \mathbf{H}_{\mathbf{p}}}{N^i i!} \; {}^G\tilde{\mathbf{p}}_I^{(i)}(t_o) + \mathbf{H}_a \tilde{\mathbf{x}}_a + \mathbf{n} \tag{5.18} $$

This equation expresses the residual as a function of the errors in the first l_θ derivatives of the orientation and the first l_p derivatives of the position. Therefore, if we include these quantities in the state vector of the estimator, we can perform an update based on the linearized expression in (5.18). However, this will only be useful if l_θ and l_p are small.

Clearly, any choice of truncation order in (5.16)-(5.17) will lead to an unmodeled error, and the lower the truncation order, the more significant the error will be in general. The key observation here is that, since we have prior knowledge about the magnitude of the estimation errors, we can *predict* the worst-case unmodeled error incurred by our choice of l_θ and l_p. To evaluate the importance of these unmodeled errors, we analyze the impact that they have on the residual. If the residual term due to the unmodeled truncation errors is small, compared to the measurement noise, this would indicate that the loss of modeling accuracy would be acceptable.

We start this analysis by re-writing (5.16)-(5.17) to illustrate the physical interpretation of the first few terms on the right-hand side of the series:

$$ {}^G\tilde{\mathbf{p}}_I(t_n) = {}^G\tilde{\mathbf{p}}_I(t_o) + \frac{nt_r}{N}\,{}^G\tilde{\mathbf{v}}_I(t_o) + \frac{(nt_r)^2}{2N^2}\,{}^G\tilde{\mathbf{a}}_I(t_o) + \dots \tag{5.19} $$

$$ {}^G\tilde{\boldsymbol{\theta}}_I(t_n) = {}^G\tilde{\boldsymbol{\theta}}_I(t_o) + \frac{nt_r}{N}\,{}^G\tilde{\boldsymbol{\omega}}(t_o) + \dots \tag{5.20} $$

Here ${}^G\tilde{\mathbf{v}}_I$ represents the error in the estimate of the IMU velocity, ${}^G\tilde{\mathbf{a}}_I$ represents the error in the IMU acceleration, and ${}^G\tilde{\boldsymbol{\omega}}$ is the error in the rotational velocity expressed in the global frame.

Let us first focus on the position errors. If we only keep the position and velocity terms in the series (i.e., $l_p = 1$), then the truncation error in (5.19) is given by

$$ \Delta\mathbf{p} = \frac{(nt_r)^2}{2N^2} \begin{bmatrix} {}^G\tilde{\mathbf{a}}_{I_x}(\tau_1) \\ {}^G\tilde{\mathbf{a}}_{I_y}(\tau_2) \\ {}^G\tilde{\mathbf{a}}_{I_z}(\tau_3) \end{bmatrix} \tag{5.21} $$

where $\tau_i \in [t_o, t_n]$, $i = 1, 2, 3$, and the unmodeled term in the residual in (5.18), due to this truncation error, is given by $\mathbf{H}_{\mathbf{p}}\Delta\mathbf{p}$. If the worst-case acceleration error in each direction is ϵ_a, the 2-norm of the truncation error is bounded above by $||\Delta\mathbf{p}||_2 \leq \sqrt{3}\frac{(nt_r)^2}{2N^2}\epsilon_a$, and the unmodeled term in the residual satisfies:

$$ \begin{aligned} \delta_p(n) &= ||\mathbf{H}_{\mathbf{p}}\Delta\mathbf{p}||_2 \\ &\leq ||\mathbf{H}_{\mathbf{p}}||_2 ||\Delta\mathbf{p}||_2 \\ &\leq \sqrt{3}\, H_{\mathbf{p}_u} \frac{(nt_r)^2}{2N^2} \end{aligned} $$

where $H_{\mathbf{p}_u}$ is an upper bound on $||\mathbf{H_p}||_2$. By choosing $n = \pm N/2$ we can compute the upper bound on the magnitude of the unmodeled residuals in the entire image (all n), as:

$$\bar{\delta}_p = \frac{\sqrt{3}}{8} H_{\mathbf{p}_u} t_r^2 \epsilon_a \tag{5.22}$$

Turning to the representation of the orientation errors, if we only maintain a *single* term in the series (i.e., $l_\theta = 0$), we similarly derive the following upper bound for the unmodeled residual term:

$$\delta_\theta(n) \leq \sqrt{3} H_{\boldsymbol{\theta}_u} \frac{|n| t_r}{N} \epsilon_\omega \tag{5.23}$$

where $H_{\boldsymbol{\theta}_u}$ is an upper bound on $||\mathbf{H}_{\boldsymbol{\theta}}||_2$, and ϵ_ω is the upper bound on the rotational velocity errors. In turn, the upper bound over all rows is given by:

$$\bar{\delta}_\theta = \frac{\sqrt{3}}{2} H_{\boldsymbol{\theta}_u} t_r \epsilon_\omega \tag{5.24}$$

We have therefore shown that if (i) we include in the state vector of the estimator the IMU position, orientation, and velocity at t_o, (ii) compute the measurement residual as shown in (5.12), and (iii) base the estimator's update equations on the linearized expression:

$$\mathbf{r} \simeq \mathbf{H}_{\boldsymbol{\theta}} {}^G\tilde{\boldsymbol{\theta}}_I(t_o) + \mathbf{H_p} {}^G\tilde{\mathbf{p}}_I(t_o) + \frac{n t_r}{N} \mathbf{H_p} {}^G\tilde{\mathbf{v}}_I(t_o) + \mathbf{H}_a \tilde{\mathbf{x}}_a + \mathbf{n} \tag{5.25}$$

we are *guaranteed* that the residual terms due to unmodeled errors will be upper bounded by $\bar{\delta}_\theta + \bar{\delta}_p$. The value of this bound will depend on the characteristics of the sensors used, but in any case it can be evaluated to determine whether this choice of truncation orders would be acceptable.

For example, for the sensors on the LG Nexus 4 smartphone used in our experiments (see Table 5.1), the standard deviation of the noise in the acceleration measurements is approximately 0.04 m/s^2. Using a conservative value of $\epsilon_a = 1$ m/s^2 (to also account for errors in the estimates of the accelerometer bias and in roll and pitch), a readout time of $t_r = 43.3$ ms, and assuming a camera with a focal length of 500 pixels and 60-degree field of view observing features at a depth of 2 m, we obtain $\bar{\delta}_p = 0.12$ pixels. Similarly, using $\epsilon_\omega = 1$ ^{o}/s, we obtain $\bar{\delta}_\theta = 0.44$ pixels (see Appendix D.1 for the derivations of these bounds). We therefore see that, for our system, the residual terms due to unmodeled errors when using (5.25) are guaranteed to be below 0.56 pixels. This (conservative) value is smaller than the standard deviation of the measurement noise, and likely in the same order as other sources of unmodeled residual terms (e.g., camera model inaccuracies and the nonlinearity of the measurement function).

The above discussion shows that, for a system with sensor characteristics similar to the ones described above, the choice of $l_p = 1$, $l_\theta = 0$ leads to approximation errors that are guaranteed to be small. If for a given system this choice is not sufficient, more terms can be kept in the two series to achieve a more precise modeling of the error. On the other hand, we can be even more aggressive, by choosing $l_p = 0$, i.e., keeping only the camera pose in the estimator state vector, and not computing Jacobians of the error with respect to the velocity. In that case, the upper bound of the residual due to unmodeled position errors becomes:

$$\delta'_p(n) \leq \sqrt{3} H_{\mathbf{p}_u} \frac{|n| t_r}{N} \epsilon_v \tag{5.26}$$

where ϵ_v is the worst-case velocity error. Using a conservative value of $\epsilon_v = 20$ cm/s (larger than what we typically observe), we obtain $\bar{\delta}'_p = 2.2$ pixels.

If these unmodeled effects were materialized, the estimator's accuracy would likely be reduced. However, we have experimentally found that the performance loss by choosing $l_p = 0$ is minimal (see Section 5.6), indicating that the computed bound is a conservative one. Moreover, as shown in the results of Section 5.6, including additional terms for the orientation error does not lead to a substantially improved performance. Therefore, in our implementations, we have favored two sets of choices: $l_p = 1, l_\theta = 0$, due to the theoretical guarantee of small unmodeled errors, and $l_p = 0, l_\theta = 0$, due to its lower computational cost, as discussed next.

5.4.1 Discussion

It is interesting to examine the computational cost of the proposed method for processing the RS measurements, as compared to the case where a GS camera is used. The key observation here is that, if the states with respect to which Jacobians are evaluated are already part of the state, then *no additional states* need to be included in the estimator's state vector, to allow processing the RS measurements. In this case, the computational overhead from the use of a RS camera, compared to a GS one, will be negligible[3].

In the previous chapters, we discussed three major types of EKF-based estimators for performing visual-inertial odometry: EKF-SLAM, the MSCKF estimator,

[3]A small increase will occur, due to the IMU propagation needed to compute the estimates ${}^G\hat{\mathbf{p}}_I(t_n)$ and ${}^I_G\hat{\bar{\mathbf{q}}}(t_n)$, $n = -N/2, \ldots, N/2$. However, this cost is negligible, compared to the cost of matrix operations in the estimator.

and the MSCKF/SLAM hybrid filter. For EKF-SLAM, since the state vector contains the IMU position, orientation, and velocity (see (2.5)), choosing $l_p \leq 1$ and $l_\theta = 0$ will require no new states to be added, and no significant overhead. For the MSCKF-based estimators, since the measurement Jacobian matrices are computed with respect to the IMU states in the sliding window, the sliding-window states (2.13) should be defined based on the selection of l_p and l_θ. If we select $l_p \geq 1$ or $l_\theta \geq 1$, additional states will have to be maintained in the sliding window, leading to increased computational requirements. Finally, since the hybrid filter is formulated by combining the MSCKF and the EKF-SLAM, its computational cost will be affected by both SLAM-type update and MSCKF-type update. As a result, for the hybrid filter, if $l_p \geq 1$ or $l_\theta \geq 1$ the computational cost will increase. However, if $l_p = l_\theta = 0$ is chosen, no additional states would have to be introduced, and the cost of processing the RS measurements would be practically identical to that of a GS camera (see also Section 5.6).

We next discuss the relationship of our approach to the TBF formulation of [11, 30, 69, 89]. First, we point out that the expressions in (5.16)-(5.17) effectively describe a representation of the errors in terms of the temporal basis functions $f_i(\tau) = \tau^i$, $i = 1, \ldots l_{p/\theta}$ in the time interval $[-t_r/2, t_r/2]$. This is similar to the TBF formulation, with the difference that in our case the *errors*, rather than the *states* are approximated by a low-dimensional parameterization. This difference has two key consequences. First, as we saw it is possible to use knowledge of the error properties to compute bounds on the effects of the unmodeled errors. Second, the errors are, to a large extent, *independent* of the actual trajectory, which makes the approach applicable in cases where the motion contains significant accelerations. By contrast, in the TBF formulation the necessary number of basis functions is crucially dependent on the nature of the trajectory. In “smooth” trajectories, one can use a relatively small number of functions, leading to low computational cost (see, e.g. [89]). However, with fast motion dynamics, the proposed error-parameterization approach requires a lower dimension of the state vector.

To demonstrate this with a concrete example, let us focus on the acceleration signal shown in Fig. 5.2(left). In [69, 89], fourth-order B-splines are used as the basis functions, due to their finite support and analytical derivatives. This, in turn, means that the acceleration is modeled by a linear function between the time instants consecutive knots are placed. If we place one knot every 43.3 ms (e.g., at the start and end of each image readout), we would be modeling the acceleration as a linear function during each image readout. The bottom plot in Fig. 5.2(right) shows the largest errors between $\mathbf{a}_m$ and the best-fit linear model during each readout time. The worst-case error is 1.03 m/s^2, which is one order of magnitude larger than the standard deviation of the accelerometer measurement noise (see Table 5.1). Therefore, placing knots every 43.3 ms would lead to unacceptably large unmodeled terms in the accelerometer residual. To reduce the error terms to the same order of magnitude as the noise, knots would have to be placed every approximately 15 ms, or approximately three poses per image. Therefore, in this example (which involves motion dynamics common in small-scale systems) the proposed error-parameterization approach would lead to a significantly faster algorithm, as it results in a much smaller state vector.

5.5 Motion Estimation with a Low-cost IMU and a Rolling-shutter Camera

In this section, we describe the main algorithm that is used for performing real-time VIO and self-calibration, using measurements from an IMU and a RS camera. Specifically, we propose a modified MSCKF/SLAM hybrid filter in this section, which is able to i) estimate the calibration parameters of both the IMU and camera online, and ii) estimate the motion of the camera-IMU platform in real time.

5.5.1 Calibration state

We first define the vector $\mathbf{x_c}$, which consists of *all* the calibration parameters:

$$\mathbf{x_c} = \begin{bmatrix} \boldsymbol{\pi}_I^T & \boldsymbol{\pi}_C^T & {}^C\mathbf{p}_I^T & t_d \end{bmatrix}^T \tag{5.27}$$

In the above equation, $\boldsymbol{\pi}_I$ is a 27×1 vector, consisting of the time-invariant parameters of the IMU, i.e., each element of the matrices $\mathbf{T}_a$, $\mathbf{T}_g$, and $\mathbf{T}_s$ (see (5.2)-(5.3)). In addition, $\boldsymbol{\pi}_C$ is the camera calibration vector, containing both the intrinsic parameters (see (5.7)-(5.9)) as well as the rolling-shutter readout time:

$$\boldsymbol{\pi}_C = \begin{bmatrix} \mathbf{p}_c^T & a_u & a_v & k_1 & k_2 & k_3 & t_1 & t_2 & t_r \end{bmatrix}^T \tag{5.28}$$

Finally, ${}^{C}\mathbf{p}_I$ is the camera-to-IMU translation vector and t_d is the time-offset between the sensor measurements, which have been discussed in the previous chapter.

5.5.2 EKF state vector

To perform online calibration, we incorporate the calibration state $\mathbf{x_c}$ into the state vector of the MSCKF/SLAM hybrid estimator. Specifically, the new state vector is defined as

$$\mathbf{x}(t) = \begin{bmatrix}\mathbf{x}_E^T(t) & \mathbf{x}_\mathbf{c}^T & \mathbf{x}_{I_1}^T & \cdots & \mathbf{x}_{I_m}^T & \mathbf{f}_1^T & \cdots & \mathbf{f}_s^T\end{bmatrix}^T \tag{5.29}$$

Compared to the state vector of the hybrid estimator defined in (4.1), which performs concurrent motion estimation and camera-to-IMU calibration, (5.29) has two differences. First, the calibration state $\mathbf{x_c}$ contains *all* the calibration parameters, instead of the camera-to-IMU configuration only. Second, as explained in Section 5.4, the formulation of each of the IMU states $\mathbf{x}_{I_i}$ depends on the choices of the truncation orders l_p and l_θ. As a result, $\mathbf{x}_{I_i}$ contains at a minimum the IMU's position and orientation, and may additionally contain some of their derivatives.

5.5.3 EKF propagation

Similarly to the original MSCKF/SLAM hybrid EKF, the IMU measurements are used for propagating the state estimates between timesteps. Specifically, given the IMU measurements in a certain time interval, as well as estimates for the IMU parameters, we compute the estimated acceleration and rotational velocity as:

$$ {}^{I}\hat{\mathbf{a}}_s = \hat{\mathbf{T}}_a^{-1}\left(\mathbf{a}_m - \hat{\mathbf{b}}_a\right) \tag{5.30} $$

$$ {}^{G}\hat{\mathbf{a}}_I = {}^{I}_{G}\hat{\mathbf{R}}^T\,{}^{I}\hat{\mathbf{a}}_s + {}^{G}\mathbf{g} \tag{5.31} $$

$$ {}^{I}\hat{\boldsymbol{\omega}} = \hat{\mathbf{T}}_g^{-1}\left(\boldsymbol{\omega}_m - \hat{\mathbf{T}}_s{}^{I}\hat{\mathbf{a}}_s - \hat{\mathbf{b}}_g\right) \tag{5.32} $$

The acceleration and rotational velocity estimates in (5.30)-(5.32) are used to propagate the estimate of the evolving state via numerical integration, as described in Section 2.5. Moreover, the covariance matrix of the EKF is propagated. For this step, we first compute the state transition matrix $\mathbf{\Phi}_{I_k}$ and the Jacobian matrix $\mathbf{\Gamma}_{I_k}$, which describe the relationship between the evolving-state errors in propagation:

$$ \tilde{\mathbf{x}}_{E_{k+1}} = \mathbf{\Phi}_{I_k}\tilde{\mathbf{x}}_{E_k} + \mathbf{\Gamma}_{I_k}\tilde{\boldsymbol{\pi}}_I + \mathbf{w}_{d_k} \tag{5.33} $$

The above equation is similar to (2.44), with the addition of the term $\mathbf{\Gamma}_{I_k}\tilde{\boldsymbol{\pi}}_I$, which models the errors due to the IMU calibration parameters. Using (5.33), the EKF's covariance propagation equation is given by:

$$ \mathbf{P}_{k+1} = \begin{bmatrix} \mathbf{\Phi}_{I_k} & \mathbf{\Gamma}_{I_k} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \mathbf{P}_k \begin{bmatrix} \mathbf{\Phi}_{I_k} & \mathbf{\Gamma}_{I_k} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix}^T + \mathbf{Diag}(\mathbf{Q}_k, \mathbf{0}) \tag{5.34} $$

5.5.4 State augmentation

When a new image is received, a new IMU state is added to the filter state vector. Let us consider the case where an image with timestamp t is received. We here note that, by our convention (see (5.4)-(5.5)), the image timestamps correspond to the midpoint of the image readout. As described in Chapter 4, a time offset, t_d, exists between the camera and IMU timestamps. Thus, if an image with timestamp t is received, the midpoint of the image readout interval was actually at time $t + t_d$. Therefore, when a new image is received, the EKF state is augmented with an estimate of the IMU state at $t + t_d$. What is included in this state depends on the choices of the truncation orders l_p and l_θ. Specifically, if $l_p = l_\theta = 0$, the IMU state only contains the position and orientation terms, similarly to the case when a GS camera is used. However, if $l_p \geq 1$ or $l_\theta \geq 1$, derivatives of the position or orientation components will be included in the IMU state that is added to the sliding window. In this chapter, we derive the equations for the case $l_p = 1$ and $l_\theta = 0$, and the other cases can be derived similarly.

When the image is received, we propagate the EKF state up to $t+\hat{t}_d$, at which point we augment the EKF state with the estimate $\hat{\mathbf{x}}_I(t+\hat{t}_d) = [{}^I_G\hat{\mathbf{q}}^T(t+\hat{t}_d) \;\; {}^G\hat{\mathbf{p}}_I^T(t+\hat{t}_d) \;\; {}^G\hat{\mathbf{v}}_I^T(t+\hat{t}_d)]^T$. Moreover, the EKF covariance matrix is augmented to include the covariance matrix of the new state, and its correlation to all other states in the system. For this computation, an expression relating the errors in the new state to the errors of the EKF state vector is needed. This is given by:

$$\tilde{\mathbf{x}}_I(t+\hat{t}_d) = \begin{bmatrix} \mathbf{I}_9 & \mathbf{0}_{9\times 46} & \mathbf{J}_t & \mathbf{0} \end{bmatrix} \tilde{\mathbf{x}}(t+\hat{t}_d) \tag{5.35}$$

where $\mathbf{J}_t$ is the Jacobian with respect to the time offset t_d, defined as

$$\mathbf{J}_t = \begin{bmatrix} {}^G_I\hat{\mathbf{R}}(t+\hat{t}_d)\,{}^I\hat{\boldsymbol{\omega}}(t+\hat{t}_d) \\ {}^G\hat{\mathbf{v}}_I(t+\hat{t}_d) \\ {}^G_I\hat{\mathbf{R}}(t+\hat{t}_d)\,{}^I\hat{\mathbf{a}}_s + {}^G\mathbf{g} \end{bmatrix} \tag{5.36}$$

5.5.5 EKF update

Once state augmentation is performed, the image is processed to extract and match features. Subsequently, these feature measurements are processed by the hybrid estimator, via either the SLAM-type update or the MSCKF-type update. The detailed feature allocation and update policy is described in Section 3.4.2. We here note that both types of updates require the computation of the measurement residual vector and the linearized measurement equation. Specifically, the residual vector corresponding to the ith IMU state and jth feature can be computed as:

$$\tilde{\mathbf{z}}_{ij} = \mathbf{z}_{ij} - \mathbf{h}(\hat{\mathbf{x}}_{I_j}, \hat{\mathbf{x}}_{\mathbf{c}}, \hat{\mathbf{f}}_i) \tag{5.37}$$

The above residual computation utilizes the state estimate $\hat{\mathbf{x}}_{I_j}$, estimated calibration parameters $\hat{\mathbf{x}}_{\mathbf{c}}$, as well as the IMU measurements in the readout time interval of the corresponding image, as described in Section 5.4. Additionally, (5.37) also requires an estimate of the feature state $\hat{\mathbf{f}}_i$, which is obtained via least-squares triangulation for a MSCKF feature or is directly available from the state vector for a SLAM feature. Linearizing the residual equation (5.37), we obtain:

$$\tilde{\mathbf{z}}_{ij} \simeq \mathbf{H}_{\boldsymbol{\theta}_{ij}} {}^G\tilde{\boldsymbol{\theta}}_{I_j} + \mathbf{H}_{\mathbf{p}_{ij}} {}^G\tilde{\mathbf{p}}_{I_j} + \frac{n_{ij} t_r \mathbf{H}_{\mathbf{p}_{ij}}}{N} {}^G\tilde{\mathbf{v}}_{I_j} + \mathbf{H}_c \tilde{\mathbf{x}}_{\mathbf{c}} + \mathbf{H}_{\mathbf{f}_{ij}} \tilde{\mathbf{f}}_i + \mathbf{n}_{ij} \tag{5.38}$$

where n_{ij} is the image row on which $\mathbf{f}_i$ is observed in image j. It is important to note that (5.38) can be directly used for an EKF update in the hybrid estimator, since the term ${}^G\tilde{\mathbf{v}}_{I_j}$ is included in the state.

5.6 Simulations

We present results from three sets of Monte-Carlo simulations demonstrating the performance the proposed algorithm. In all of these simulations, we generated simulation environments based on real-world data-sets, as described in Appendix A.6.

5.6.1 Processing RS measurements

The first simulation test is to validate the proposed method for processing the measurements from a RS camera. In this simulation, the simulation environment was generated based on a dataset collected by a Nexus 4 mobile device. The device is equipped with a RS camera capturing images at 11 Hz with a readout time of 43.3 ms, and an Invensense MPU-6050 IMU, which provides inertial measurements at 200 Hz. The noise characteristics and additional details for the sensors can be found in Table 5.1. During the data collection, the device was hand-held by a person walking at normal pace, for a duration of 4.23 min. The total distance traveled is approximately 227 m.

We here compare the performance of our proposed approach to the processing of RS measurements, with four different options for the modeling of the errors during the readout time, obtained by choosing $l_p = \{0, 1\}$ and $l_\theta = \{0, 1\}$. We additionally evaluate the performance of the constant-velocity approach of [61], and of an approach that treats the camera as if it has a global shutter. To collect statistics for these six approaches, we carried out 50 Monte-Carlo simulation trials. Moreover, in order to isolate the effects of the RS model used, in this simulation,

Table 5.1: Sensor characteristics of the mobile devices used in the experiments.

Device	LG Nexus 4	Galaxy SIII
Gyroscope rate (Hz)	200	200
Accelerometer rate (Hz)	200	100
σ_r (^{o}/s)	$2.4 \cdot 10^{-1}$	$2.6 \cdot 10^{-1}$
σ_a (m/s^2)	$4.0 \cdot 10^{-2}$	$5.6 \cdot 10^{-2}$
σ_{wg} ($^o/\sqrt{\text{s}^3}$)	$1.6 \cdot 10^{-3}$	$3.2 \cdot 10^{-3}$
σ_{wa} (m$/\sqrt{\text{s}^5}$)	$7.0 \cdot 10^{-5}$	$1.4 \cdot 10^{-4}$
t_r (ms)	43.3	15.9
Frame rate (Hz)	11	20
Resolution (pixels)	432×576	480×640
σ_{im} (pixels)	0.75	0.75

we assume that the calibration parameter vector $\mathbf{x_c}$ is *perfectly* known and *not* estimated online. Additionally, in each simulation trial all the approaches use exactly the same initial state estimate and covariance matrix, the same estimator structure (the hybrid filter presented in Section 5.5), and process exactly the same IMU and feature measurements.

Fig. 5.3 shows the RMSE (averaged over the 50 Monte-Carlo trials) in the estimates of the IMU orientation, position, and velocity computed by the six approaches. In Table 5.2 we also provide the average RMSE and the average NEES for the IMU motion state (position, orientation, and velocity), averaged over all 50 trials, and over the last 25 seconds of motion[4]. We note that, since

[4]Since no loop-closure information is available, the uncertainty of the position and the yaw gradually increases over time. We here plot the statistics for the last 25 seconds, to evaluate the estimators' final accuracy and consistency.

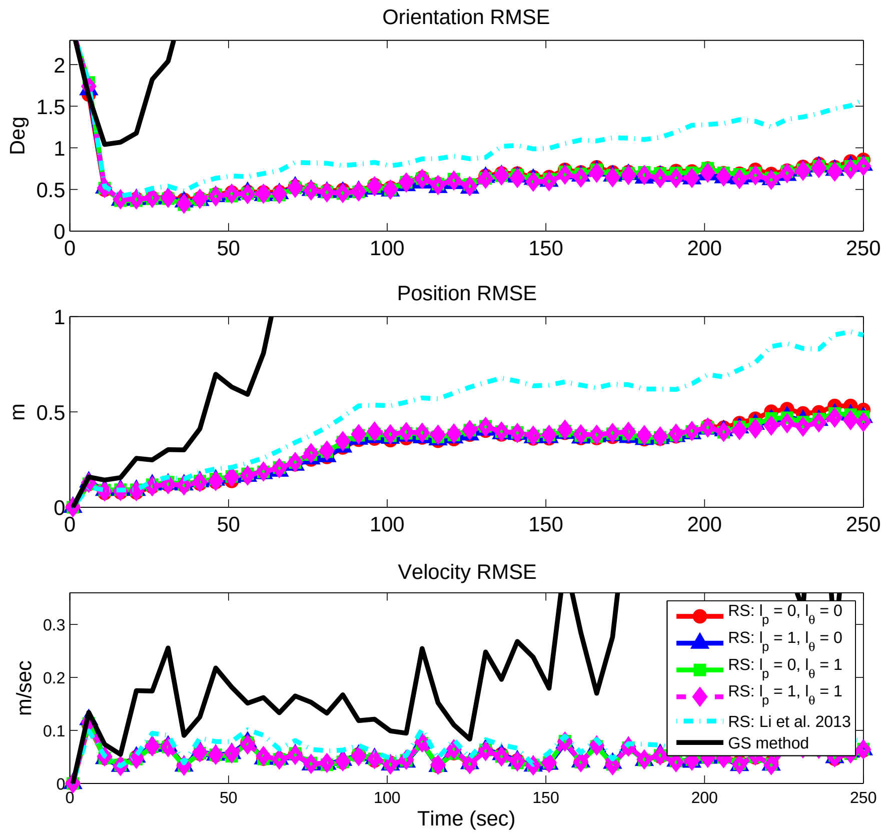


Figure 5.3: Simulation results: The RMSE of the IMU orientation, position, and velocity over 50 Monte-Carlo simulation trials for different RS models. Because the performance of the four error-state models is very similar, the lines are hard to distinguish in the plot. For clarity, the numerical values of the average RMSE are provided in Table 5.2.

Table 5.2: Simulation Results: RMSE and NEES for different RS models.

	GS	Li et al. 2013	Proposed method			
Position error model	N/A	N/A	$l_p = 0$	$l_p = 1$	$l_p = 0$	$l_p = 1$
Orientation error model	N/A	N/A	$l_\theta = 0$	$l_\theta = 0$	$l_\theta = 1$	$l_\theta = 1$
Position RMSE (m)	5.403	0.806	0.471	0.447	0.447	**0.427**
Orientation RMSE (deg)	15.12	1.34	0.735	**0.693**	0.723	0.695
Velocity RMSE (m/s)	0.302	0.077	0.053	**0.052**	**0.052**	**0.052**
IMU state NEES	571.2	19.78	11.05	10.64	10.56	**10.44**
Time per update (ms)	**0.85**	2.17	0.87	1.49	1.50	2.24

the hybrid filter is able to report consistent estimation results for GS cameras (see Section 3.6.1), by examining the average NEES values of the six approaches, we can evaluate the significance of the unmodeled errors due to the RS model.

Several observations can be made based on the results of Fig. 5.3 and Table 5.2. First, we clearly observe that the approach that assumes a GS camera model has very poor performance, with errors that are one order of magnitude larger than all other methods. By assuming a GS camera, the motion of the camera during the readout time is neither modeled nor compensated for, which inevitably causes unmodeled errors and degrades the estimation accuracy. The existence of large unmodeled errors is also reflected in the average NEES value, which is substantially higher than that of all other methods.

From these results we can also see that the constant-velocity models used in [61] result in significantly larger errors, compared to the four models that are based on the proposed approach. Specifically, the position and orientation errors are approximately double, while the velocity errors are approximately 50% larger.

As discussed in Section 5.2.2, this is due to the fact that the motion in this experiment is characterized by significant variations, especially in the rotational velocity. These variations, which are to be expected in low-mass systems such as the one used here, cause the constant-velocity assumption to be severely violated, and lead to the introduction of non-negligible unmodeled errors. These errors also lead to an increase in the average NEES.

Turning to the four different error parameterizations that are based on the approach proposed here, we see that they all perform similarly in terms of accuracy. As expected, the choice $l_p = 1, l_\theta = 1$ outperforms the lower-order models, but only by a small margin. Moreover, we see that using a higher-order model leads to a lower NEES, as the unmodeled errors become smaller. The average NEES of the four different approaches is somewhat higher than the theoretically-expected value of 9. This outcome is to be anticipated, due not only to the approximations in the RS modeling, but also to the non-linear nature of the estimation problem.

Even though the accuracy and consistency of the four models is comparable, the computational cost incurred by their use is significantly different. Increasing the order of the error model results in an increase in the dimension of the error-state vector, and therefore in a higher computational cost. The last row in Table 5.2 shows the average CPU time needed per update in each of the cases tested, measured on an Intel Core i3 2.13 GHz processor. These times show that, even though the accuracy difference between the most accurate error model ($l_p = l_\theta = 1$) and the least accurate one ($l_p = l_\theta = 0$) is less than 10%, the latter is more than 2.5 times faster. In fact, the computational cost of the method with

$l_p = l_\theta = 0$ is practically identical to the cost of using a GS model, and 2.5 times faster than the model of [61]. Since our main interest is in resource-constrained systems, where CPU capabilities are limited and preservation of battery life is critical, the preferred choice in our implementation of real-time VIO is $l_p = l_\theta = 0$. If additional precision were required, based on the results of Table 5.2 and the theoretical results of Section 5.4, we would select the model with $l_p = 1, l_\theta = 0$. Both these two sets of choices are tested in the subsequent simulation tests and real-world experiments.

5.6.2 Online parameter calibration

In this simulation test, we examine the capability of the proposed algorithm to perform online calibration of the 41×1 calibration state $\mathbf{x_c}$. For this test, we base our simulation on a dataset involving significant rotations and translations, while moving in a room-sized environment for a period of 45 sec. The data was also collected using an LG Nexus 4 device. In each trial of the simulation tests, the calibration parameters were set equal to known nominal values, with the addition of errors drawn from zero-mean Gaussian distributions. In this simulation test, we use the truncation orders $l_p = 1$ and $l_\theta = 0$.

Table 5.3 shows the RMS errors over 100 Monte-Carlo simulations for all the parameters. The second and sixth lines correspond to the initial errors, the third and seventh lines to the errors after 15 sec of motion, and the fourth and eighth lines to the final errors. We note that the initial RMS error values for the $\mathbf{T}_a$ and

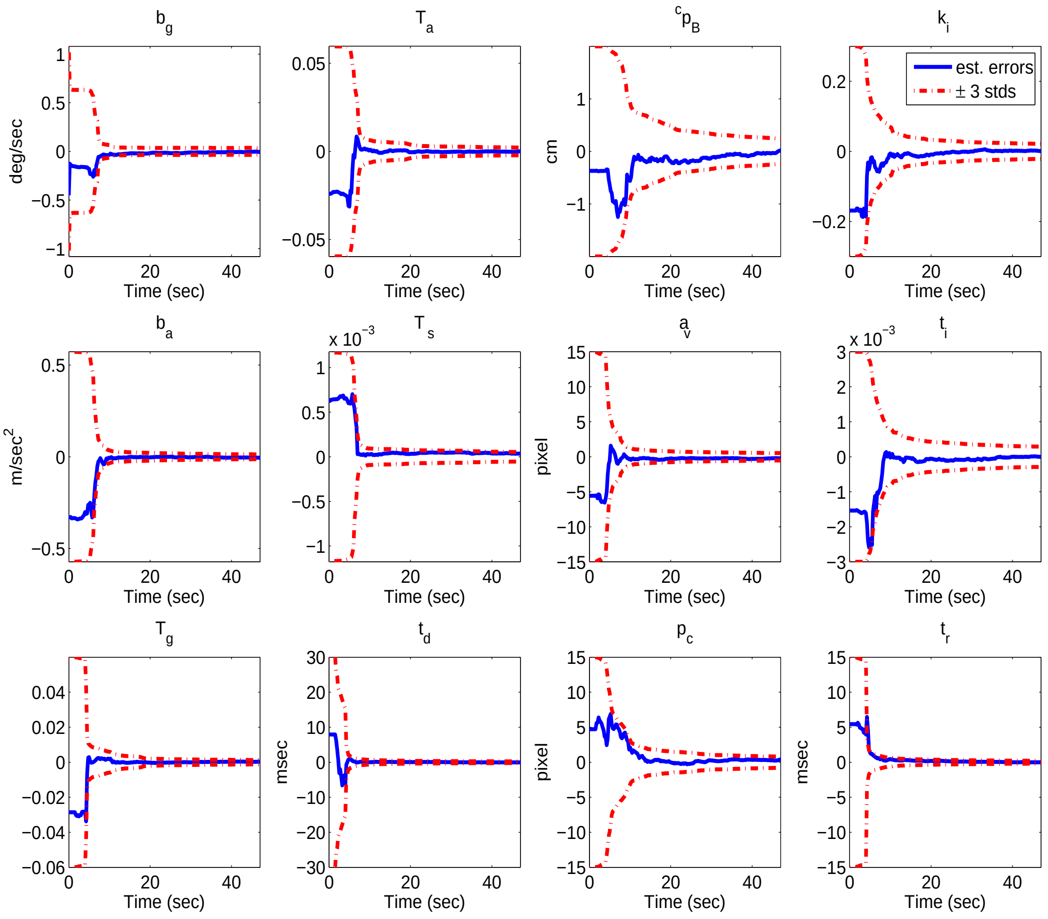


Figure 5.4: Results of a representative simulation trial: estimation errors and the reported ±3 standard deviations. (Top to bottom, left to right) (a) gyroscope biases, (b) accelerometer biases, (c) gyroscope misalignment/scale, (d) accelerometer misalignment/scale, (e) g-sensitivity, (f) camera-to-IMU time offset, (g) camera-to-IMU position (h) camera focal lengths, (i) camera principal point, (j) camera radial distortion, (k) camera tangential distortion, and (l) rolling shutter readout time. For all vectorial parameters the least accurate element is shown.

Table 5.3: Simulations: RMS errors of the calibration parameters.

Time (sec)	$\mathbf{b}_g$ (°/sec)	$\mathbf{b}_a$ (m/sec^2)	${}^C\mathbf{p}_B$ (cm)	t_d (msec)	$\mathbf{T}_g$	$\mathbf{T}_a$
0	0.334	0.187	0.610	9.450	0.0200	0.0200
15	0.014	0.009	0.160	0.076	0.0007	0.0011
45	0.013	0.007	0.110	0.062	0.0006	0.0008
Time (sec)	$\mathbf{T}_s$ (°/sec/g)	a_u, a_v (pix.)	$\mathbf{p}_c$ (pix.)	k_i	t_i	t_r (msec)
0	0.216	5.146	4.941	0.082	0.0010	4.171
15	0.020	0.475	0.470	0.015	0.0004	0.190
45	0.010	0.400	0.332	0.012	0.0003	0.108

$\mathbf{T}_g$ matrices correspond to misalignment errors with standard deviation of approximately 1.15^o and/or scale factors of 2%. These values (as well as most of the other initial errors shown in Table 5.3) are larger than what we typically find in practice. From the results of this table we can clearly observe that the proposed online calibration approach can accurately estimate all the desired quantities. Moreover, we note that the estimation errors after 15 seconds are significantly smaller than the initial ones, and almost as same as the final errors, indicating quick convergence.

It is also useful to examine the results of a single representative trial, shown in Fig. 5.4. These plots show the estimation errors and ± 3 standard deviations reported by the filter for all the calibration parameters. For the vector parameters (e.g., biases, the $\mathbf{T}_a$, $\mathbf{T}_g$, $\mathbf{T}_s$ matrices, and so on) we here only plot the least accurate element. These figures also demonstrate the rapid reduction in uncertainty, and show that the uncertainty reported by the EKF is commensurate with the actual errors, indicating consistency. It is important to note that the results of both Fig. 5.4 and Table 5.3 suggest that, with sufficiently exciting motion, the

Table 5.4: Initial standard deviation of the calibration parameters in the third set of simulations.

$\mathbf{T}_g$	$\mathbf{T}_a$	$\mathbf{T}_s$ (°/sec/g)	a_u, a_v (pix.)
0.0058	0.0058	0.169	0.350
$\mathbf{p}_c$ (pix)	k_i	t_i	t_r (msec)
0.450	0.01	0.001	0.500

calibration parameters are *observable* (identifiable), a result that we seek to prove in our future work.

5.6.3 Concurrent motion estimation and online calibration

We now present the results of a third set of simulation tests, which demonstrates the improvement in performance gained by using high-fidelity IMU models and online calibration. In this simulation, we use the truncation orders $l_p = 1$ and $l_\theta = 0$. We here compare the proposed approach that uses full self-calibration against the approach that is described in Chapter 4, in which only the IMU biases, camera-to-IMU spatial relationship, and the time offset between the camera and IMU are estimated. Since the approach in Chapter 4 only calibrates a subset of the sensor parameters, we term it "partial calibration" here. In the simulations, the IMU and camera are affected by *small* calibration errors, whose standard deviations are shown in Table 5.4. Note that the chosen values for $\mathbf{T}_a$ and $\mathbf{T}_g$ correspond to scale factors of less than 1% and/or axis misalignment of approximately 0.3 degrees (these are similar to or smaller than the values we encountered in practice

Table 5.5: Third simulation: RMSE of pose estimation.

	$\delta\boldsymbol{\theta}_R$ (°)	$\delta\mathbf{p}_R$ (m)	$\delta\mathbf{v}_R$ (m/sec)
Full calibration	0.468	0.619	0.027
Partial calibration	2.045	2.914	0.092

for MEMS sensors). Moreover, the camera parameters' errors are similar to what we obtain with offline calibration.

We conducted 100 Monte-Carlo tests, generated based on a 6-minute, 400-m long dataset. In each trial the calibration parameters are corrupted by random errors, drawn from zero-mean Gaussian pdfs with the standard deviations given in Table 5.4. Table 5.5 shows the RMS errors for the position, orientation, and velocity, in the case of full calibration (the proposed approach) vs. partial calibration. We can clearly observe a significant difference in accuracy in all state variables. Moreover, we should point out that the partial-calibration approach *failed* in 15% of the trials, due to the presence of unmodeled errors (the results of the failed trials are not included in the calculations). By properly modeling the uncertainty of the calibration, the proposed approach is more robust to the presence of errors (no simulation trials failed for the full-calibration approach). We thus see that by using high-fidelity models of the IMU and camera, and performing online estimation of the model parameters, we are able to obtain better precision as well as increased reliability of the estimator.

5.7 Real-world Experiments

We here present results of the real-world experiments. Similarly to the simulation tests, we conducted multiple sets of experiments, to evaluate different aspects of the proposed algorithm.

5.7.1 Processing RS measurements

In our first experiment, we test the performance of the proposed method for processing RS measurements. Similarly to the simulations presented in Section 5.6.1, we pre-calibrated the IMU parameters, and did not estimate them online, to isolate the other factors which might affect the performance of the algorithm.

In this experiment, a Samsung Galaxy SIII mobile phone and a Xsens MTi-G unit (for GPS ground-truth data collection) was mounted on top of a car driven on the streets of Riverside, CA. Fig. 5.5 shows the setup of the devices. The total distance driven is approximately 11 km, covered in 21 min. Sample images from the experiment are shown in Fig. 5.6. During this experiment the sensor data were saved to disk, and later processed offline, to allow the comparison of the alternative methods. Image features are extracted by an optimized Shi-Tomasi feature extractor [65], and matched using normalized cross-correlation (NCC). A 17×17 image template is used for NCC, and a minimum matching threshold of 0.8.

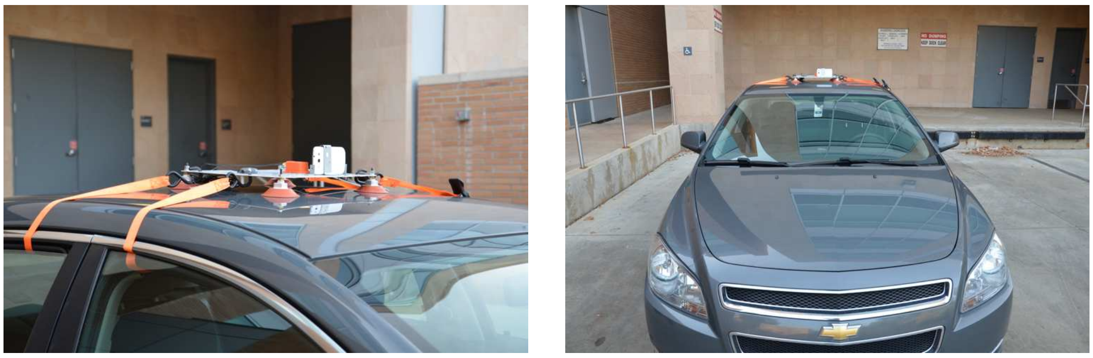


Figure 5.5: Car experiment: Device setup.

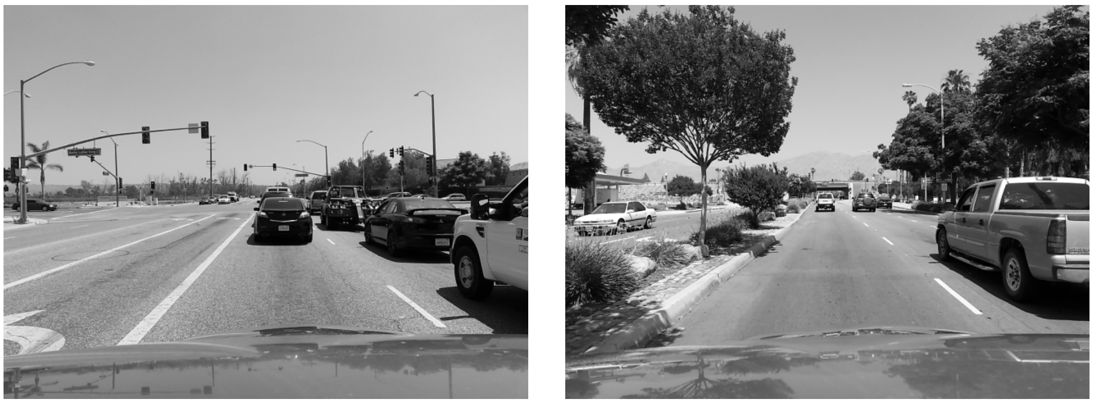


Figure 5.6: Car experiment: Sample images.

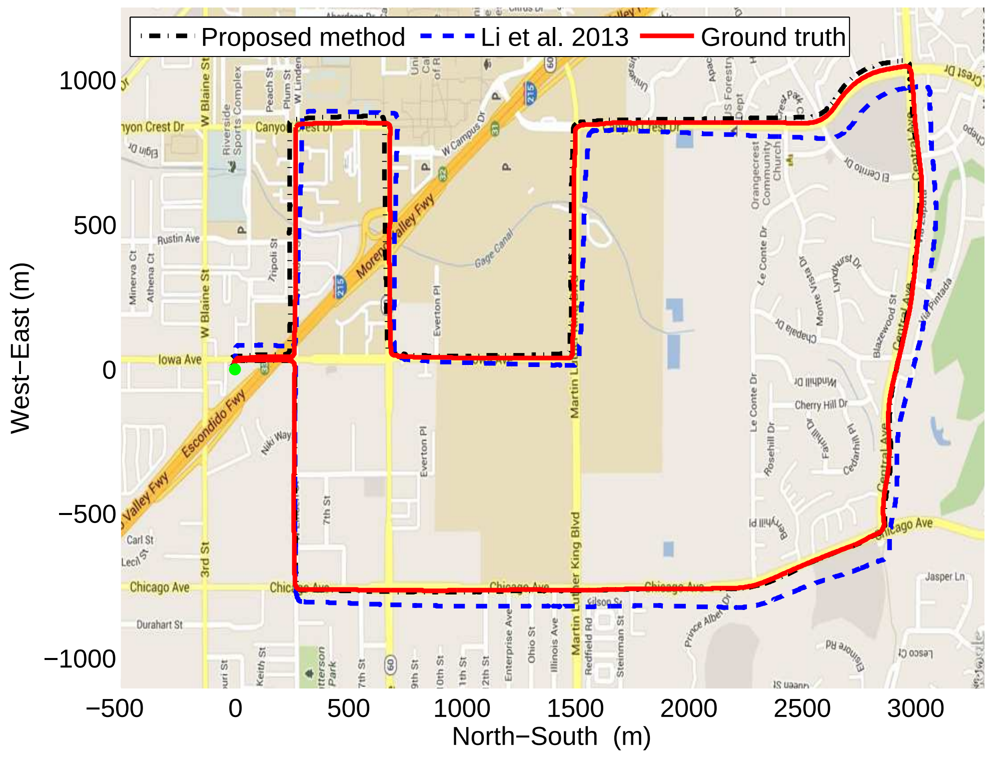


Figure 5.7: Car experiment: Trajectory estimates by the proposed approach and the approach of [61], compared to ground truth.

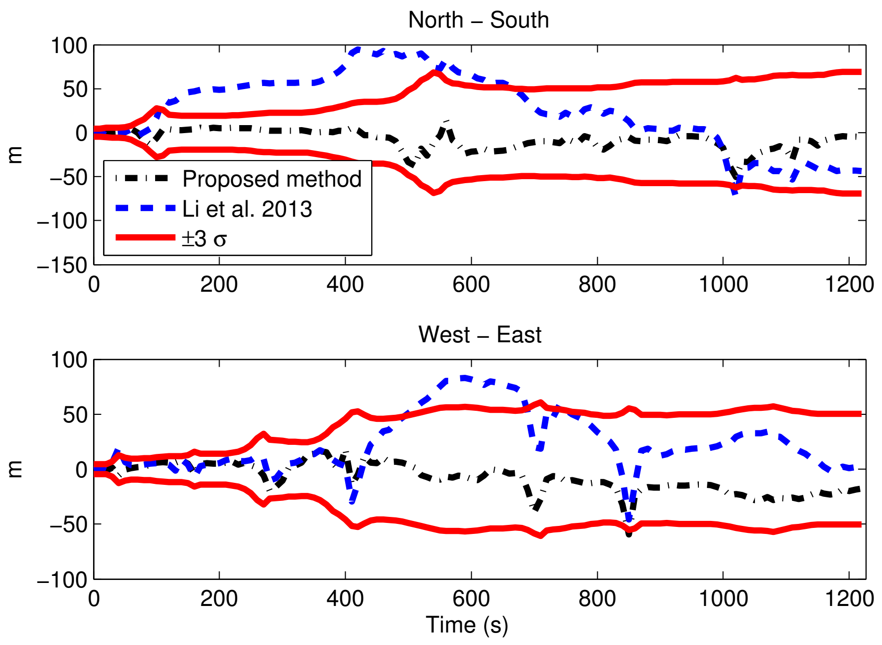


Figure 5.8: Car experiment: Estimation errors for the two approaches compared.

In this experiment, we compared three methods: (i) the proposed method using $l_p = l_\theta = 0$, (ii) the method that employs the linear velocity model during rolling shutter readout time [61], and (iii) a GPS system, which is treated as the ground truth. We note that, since the method that employs the global-shutter camera measurement model performs even worse that the method [61] (see Section 5.6.1), this method was not implemented for comparison here. Fig. 5.7 shows the trajectories obtained by these three methods. The estimated trajectories are manually aligned to the GPS ground truth, and plotted on a map of the area where the experiment took place. Moreover, Fig. 5.8 shows the position errors of the two approaches, as well as the reported uncertainty envelope for the proposed approach. This envelope corresponds to ±3 standard deviations, computed as the square roots of the corresponding diagonal elements of the EKF's state covariance

Figure 5.9: Sample images recorded during the experiment.

matrix. Here we only plot the position errors in the horizontal plane, as the accuracy of the ground truth in the vertical direction is not sufficiently high. From these results it becomes clear that the proposed method outperforms that of [61] by a wide margin. It produces more accurate estimates (the largest position error throughout the experiment is approximately 63 m, compared to 123 m for [61]), and the estimation errors agree with the estimator's reported uncertainty. This result is similar to what we obtained in simulation, which demonstrates that the proposed method provides better characterization of the estimation errors during the rolling-shutter readout time and thus leads to better performance.

5.7.2 Concurrent motion estimation and online calibration

We next present results from another real-world experiment, to demonstrate the capability of the proposed algorithm to perform concurrent motion estimation and online calibration. This experiment was conducted using a hand-held Nexus 4

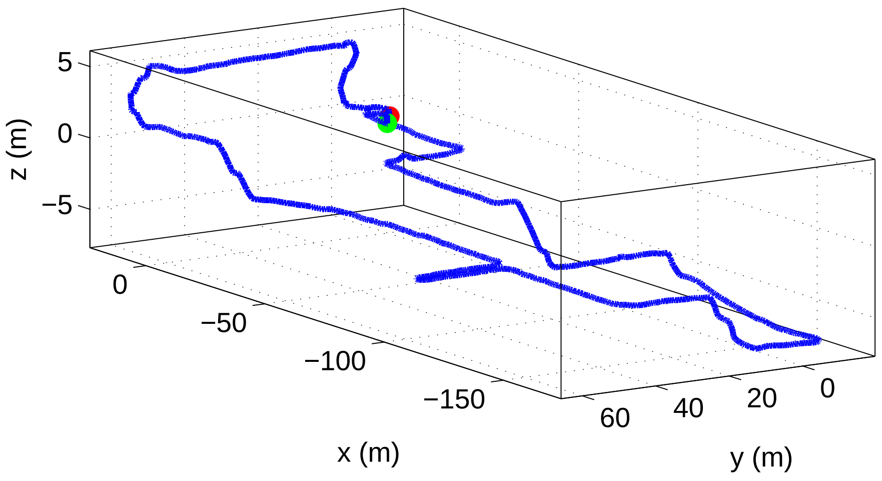


Figure 5.10: Real world experiment: Estimated trajectory. The red mark represents the initial position, while the green mark represents the end position.

device, involving motion in three floors of the UCR Engineering building (sample images are shown in Fig. 5.9). The total duration of the experiment is 9.8 min and the trajectory length is approximately 633 m. It is important to note that, in this experiment, all the sensor measurements were processed in *real time* on the phone.

For this experiment, prior estimates for the calibration parameters are obtained as follows: zero values are used for the biases and for the g-sensitivity matrix, $\mathbf{T}_s$; the priors for $\mathbf{T}_a$ and $\mathbf{T}_g$ are set to the identity matrix; the image readout time was set equal to the image period; the camera-to-IMU translation prior is set to zero; the time offset t_d prior was set to zero; and finally for the camera intrinsics the calibration parameters from a different device of the same type were used. This is done to demonstrate that even rough information about the camera parameters can be employed as an initial guess for the proposed method.

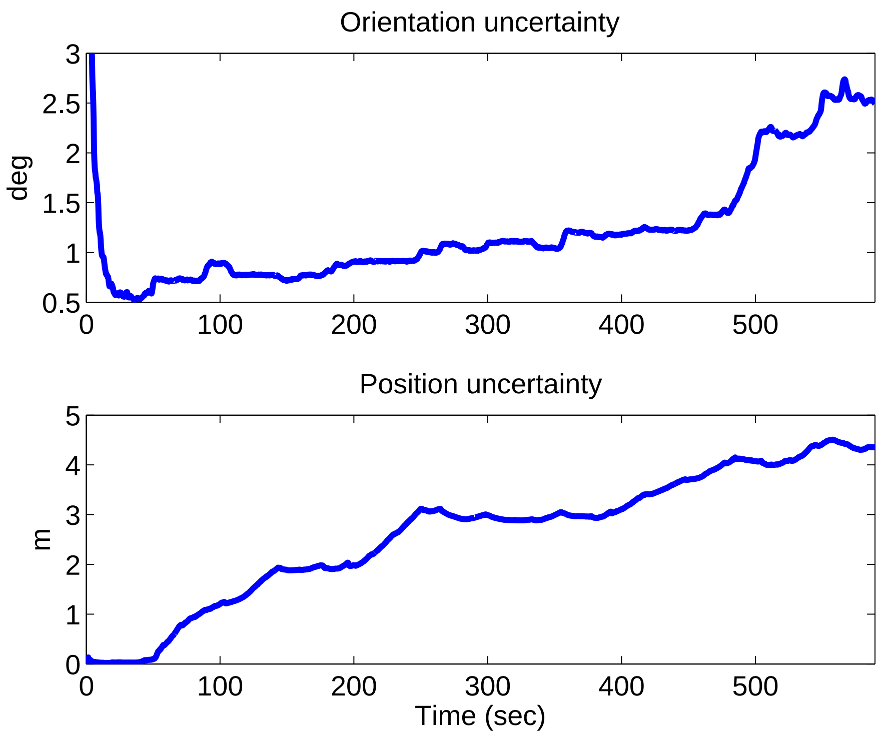


Figure 5.11: Real world experiment: Orientation and position uncertainty (3σ) reported by the EKF during the experiment. The plotted values correspond to the major axis of the uncertainty ellipses. Note the sharp drop in the initial orientation uncertainty, which occurs as the calibration parameters become more certain during the first few seconds.

Fig. 5.10 shows the trajectory estimated by the proposed approach, while the reported uncertainties of the IMU orientation and position are shown in Fig. 5.11. Although an accurate ground truth for the entire trajectory cannot be obtained, the final position error is equal to 1.09 m, corresponding to 0.17% of the travelled distance. This error is commensurate with the reported uncertainty, indicating a consistent estimator. We point out that the largest misalignment of the IMU axes was estimated as 0.85^o, the maximum scale factor deviation from unity was estimated as 1.8%, and the maximum value of the g-sensitivity corresponds to 0.33^o/sec/g. As seen in the simulation results of the preceding section, these values are large enough to cause significant degradation of performance, if not properly modeled. In addition, Fig. 5.12 plots the uncertainty curves of the calibration parameters. Similarly to Fig. 5.4, this figure shows the results for the element with the highest uncertainty, for all vectorial parameters. Fig. 5.12 shows that the uncertainty of all calibration parameters drops rapidly during the first few seconds of estimation, which indicates that these parameters are *observable* and can be calibrated in a short time.

5.8 Conclusion

In this chapter, we have presented an online approach to process the measurement from a low-cost IMU and a rolling-shutter camera for VIO. To model the low-cost sensors, our approach employs detailed models of the IMU, the camera projection geometry, as well as of the relative spatial configuration and timing

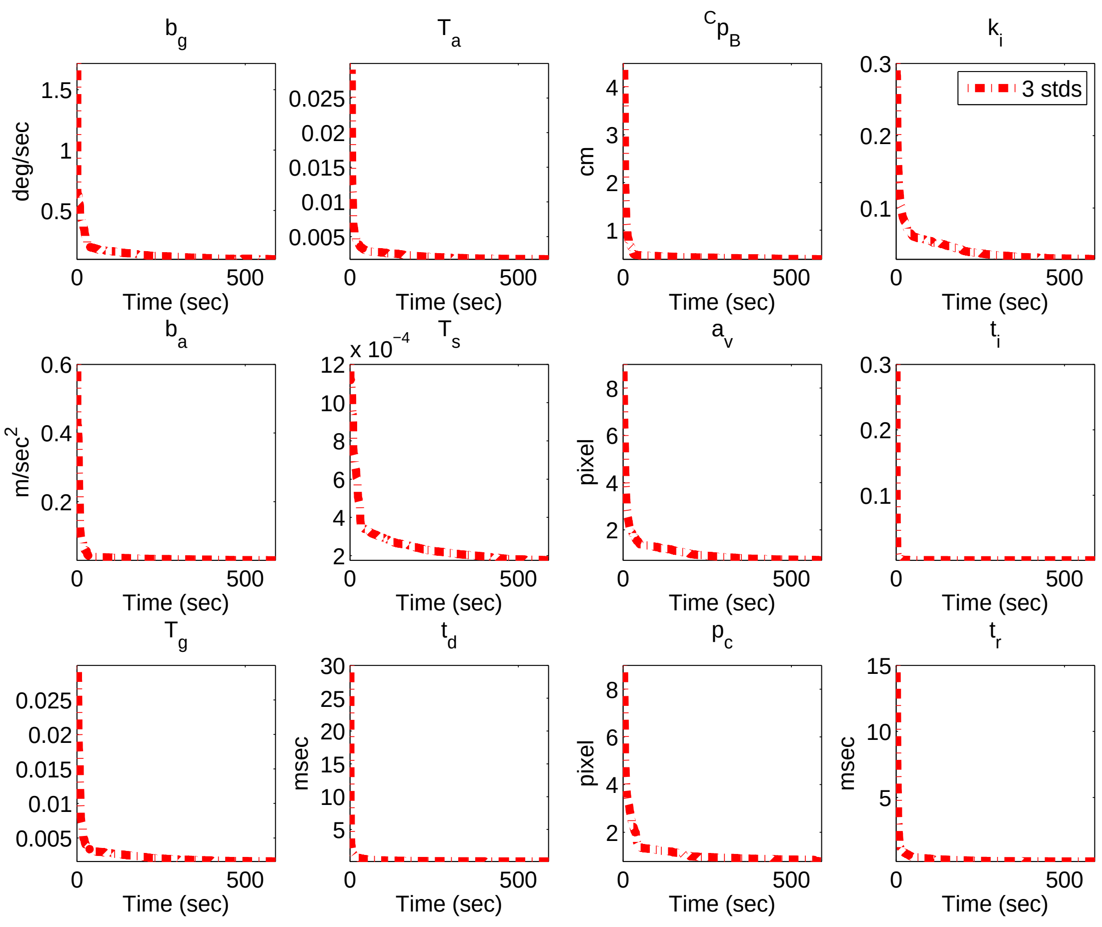


Figure 5.12: Experiment Results: reported ± 3 standard deviations of the calibration parameters.

of the sensors. In addition, we have proposed a novel approach for utilizing the measurements from rolling-shutter cameras, by employing low-dimensional approximations of the estimation errors instead of the trajectory. Our simulation results and experimental testing demonstrate that (i) by including *all* the parameters of the sensor models in the state vector of the proposed EKF, it is possible to obtain precise estimates of their values, and to improve the accuracy of the pose estimates, and (ii) by employing the proposed method for processing the RS measurements, the motion estimation accuracy can be substantially improved, compared to prior approaches.

Chapter 6

Summary

Methods for high-precision, real-time VIO, suitable for miniaturized and inexpensive mobile platforms, will be key enabling technologies for several commercial and research applications. The key challenges towards the development of such methods stem from the limited sensing and computational resources of these systems. In this work, we have proposed algorithm-design methodologies to address these challenges in systematic ways.

Specifically, we first designed an EKF-based VIO algorithm, termed MSCKF 2.0, which is able to accurately estimate the motion of a moving platform, by ensuring the correct observability properties of the EKF's linearized system model. Subsequently, we showed that the computational cost of EKF-based VIO can be substantially reduced by integrating the MSCKF 2.0 algorithm with an algorithm with complementary computational characteristics. The resulting MSCKF/SLAM hybrid estimator employs learned statistical information about the environment, in order to determine the optimal feature-processing strategy, leading to the optimal utilization of the available CPU resources. In addition to addressing the challenges caused by limited processing power, we described methods for modeling and compensating for the special characteristics of low-cost sensors. In Chapter 4, we described an extension to the hybrid EKF estimator, which performs concurrent localization and online calibration of the camera-to-IMU spatial and temporal configuration. Our theoretical analysis proves that both these parameters are identifiable under general trajectories, and our results show that including them in the hybrid EKF leads to improved estimation precision. To further improve precision, in Chapter 5 we proposed employing high-fidelity sensor models for the IMU and the camera, and calibrating all the model parameters online. By accounting for the systematic errors that are common in low-cost visual and inertial sensors, this approach leads to more accurate state estimates. In addition, Chapter 5 describes a computationally-efficient formulation for processing the measurements from rolling-shutter sensors, which are prevalent in low-cost cameras. Our experimental results demonstrate that, taken together, the above contributions lead to computationally efficient, high-precision VIO algorithms, which enable real-time localization on low-cost consumer devices.

Bibliography

[1] J. Adams, A. Robertson, K. Zimmerman, and J. How. Technologies for spacecraft formation flying. In *Proceedings of ION-GPS 96*, pages 1321–1330, New Orleans, LA, Sept. 17-20 1996.

[2] O. Ait-Aider, N. Andreff, and J. M. Lavest. Simultaneous object pose and velocity computation using a single view from a rolling shutter camera. In *Proceedings of the European Conference on Computer Vision*, pages 56–68, 2006.

[3] O. Ait-Aider and F. Berry. Structure and kinematics triangulation with a rolling shutter stereo rig. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1835 –1840, Oct. 2009.

[4] M. C. Amann, T. Bosch, M. Lescure, R. Myllylä, and M. Rioux. Laser Ranging: A critical review of usual techniques for distance measurement. *Optical Engineering*, 40(1):10–19, Jan. 2001.

[5] M. Bak, T. Larsen, M. Norgaard, N. Andersen, N. K. Poulsen, and O. Ravn. Location estimation using delayed measurements. In *Proceedings of the IEEE International Workshop on Advanced Motion Control*, Coimbra, Portugal, July 1998.

[6] Y. Bar-Shalom. Update with out-of-sequence-measurements in tracking: Exact solution. *IEEE Transactions on Aerospace and Electronic Systems*, 38(3):769–778, July 2002.

[7] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan. *Estimation with Applications to Tracking and Navigation*. John Wiley & Sons, 2001.

[8] B. Barshan and H. F. Durrant-Whyte. Inertial navigation systems for mobile robots. *IEEE Transactions on Robotics and Automation*, 11(3):328–342, June 1995.

[9] D. S. Bayard and P. B. Brugarolas. An estimation algorithm for vision-based exploration of small bodies in space. In *Proceedings of the American Control Conference*, pages 4589–4595, Portland, Oregon, June 2005.

[10] R. Bellman and K. Astrom. On structural identifiability. *Mathematical Biosciences*, (7):329–339, 1970.

[11] M. Bosse and R. Zlot. Continuous 3D scan-matching with a spinning 2D laser. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4312–4319, Anchorage, May 2009.

[12] M. Bosse, R. Zlot, and P. Flick. Zebedee: Design of a spring-mounted 3-D range sensor with application to mobile mapping. *IEEE Transactions on Robotics*, 28(5):1104–1119, 2012.

[13] P.-J. Bristeau, F. Callou, D. Vissiere, and N. Petit. The navigation and control technology inside the AR.Drone Micro UAV. In *Proceedings of the 18th World Congress of the International Federation of Automatic Control*, pages 1477–1484, Milano, Italy, August 2011.

[14] R. Brockers, S. Susca, D. Zhu, and L. Matthies. Fully self-contained vision-aided navigation and landing of a micro air vehicle independent from external sensor inputs. In *Proc. SPIE*, volume 8387, 2012.

[15] F. Chenavier and J. Crowley. Position estimation for a mobile robot using vision and odometry. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2588–2593, Nice, France, May 1992.

[16] M. Choi, J. Choi, J. Park, and W. K. Chung. State estimation with delayed measurements considering uncertainty of time delay. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3987 –3992, Kobe, Japan, May 2009.

[17] J. Civera, D. R. Bueno, A. Davison, and J. M. M. Montiel. Camera self-calibration for sequential bayesian structure from motion. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 403–408, Kobe, Japan, May 2009.

[18] J. Civera, A. Davison, and J. Montiel. Inverse depth parametrization for monocular SLAM. *IEEE Transactions on Robotics*, 24(5):932–945, Oct. 2008.

[19] A. J. Davison and D. W. Murray. Simultaneous localisation and map-building using active vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):865–880, July 2002.

[20] D. D. Diel, P. DeBitetto, and S. Teller. Epipolar constraints for vision-aided inertial navigation. In *Proceedings of the IEEE Workshop on Motion and Video Computing*, pages 221–228, Breckenridge, CO, Jan. 2005.

[21] T. Dong-Si and A. I. Mourikis. Motion tracking with fixed-lag smoothing: Algorithm and consistency analysis. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 5655 – 5662, Shanghai, China, May 2011.

[22] J. V. Doren, S. Douma, P. V. den Hof, J. Jansen, and O. Bosgra. Identifiability: From qualitative analysis to model structure approximation. In *Proceedings of the 15th IFAC Symposium on System Identification*, pages 664–669, Saint-Malo, France, 2009.

[23] C. Engels, H. Stewenius, and D. Nister. Bundle adjustment rules. In *Proceedings of the Photogrammetric Computer Vision Conference*, pages 266–271, Bonn, Germany, Sep. 2006.

[24] R. M. Eustice, H. Singh, and J. J. Leonard. Exactly sparse delayed-state filters for view-based SLAM. *IEEE Transactions on Robotics*, 22(6):1100–1114, Dec. 2006.

[25] J. Farrell. *Aided navigation: GPS with high rate sensors.* Mcgraw-Hill, 2008.

[26] A. Fertner and A. Sjolund. Comparison of various time delay estimation methods by computer simulation. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 34(5):1329–1330, Oct. 1986.

[27] A. M. Flynn. Combining sonar and infrared sensors for mobile robot navigation. *International Journal of Robotics Research*, 7(6):5–14, Dec. 1988.

[28] P. Forssen and E. Ringaby. Rectifying rolling shutter video from hand-held devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 507 –514, San Francisco, CA, June 2010.

[29] E. Foxlin. Pedestrian tracking with shoe-mounted inertial sensors. *IEEE Computer Graphics and Applications*, 25(6):38–46, Nov. 2005.

[30] P. Furgale, T. Barfoot, and G. Sibley. Continuous-time batch estimation using temporal basis functions. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2088–2095, Minneapolis, MN, May. 2012.

[31] G. Giovanni and G. Scarano. Discrete-time techniques for time-delay estimation. *IEEE Transactions on Signal Processing*, 42(2):525–533, Feb. 1993.

[32] J. E. Guivant and E. M. Nebot. Optimization of the simultaneous localization and map building algorithm for real time implementation. *IEEE Transactions on Robotics and Automation*, 17(3):242–257, June 2001.

[33] C. Guo and S. Roumeliotis. IMU-RGBD camera navigation using point and plane features. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3164–3171, Tokyo, Japan, Nov 2013.

[34] G. Hanning, N. Forslow, P. Forssen, E. Ringaby, D. Tornqvist, and J. Callmer. Stabilizing cell phone video using inertial measurement sensors. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 1 –8, Barcelona, Spain, Nov. 2011.

[35] A. Harrison and P. Newman. TICSync: knowing when things happened. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 356–363, Shanghai, China, May 2011.

[36] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, Cambridge, UK, 2000.

[37] J. Hedborg, P. Forssen, M. Felsberg, and E. Ringaby. Rolling shutter bundle adjustment. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1434 –1441, Providence, RI, June 2012.

[38] J. Hedborg, E. Ringaby, P. Forssen, and M. Felsberg. Structure and motion estimation from rolling shutter video. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 17 –23, Barcelona, Spain, Nov. 2011.

[39] J. A. Hesch, D. G. Kottas, S. L. Bowman, and S. I. Roumeliotis. Observability-constrained vision-aided inertial navigation. Technical report, Dept. of Computer Science and Engineering, University of Minnesota, 2012.

[40] G. P. Huang, A. I. Mourikis, and S. I. Roumeliotis. Analysis and improvement of the consistency of extended Kalman filter-based SLAM. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 473–479, Pasadena, CA, May 2008.

[41] G. P. Huang, A. I. Mourikis, and S. I. Roumeliotis. Observability-based rules for designing consistent EKF SLAM estimators. *The International Journal of Robotics Research*, 29(5):502–529, Apr. 2010.

[42] C. Jia and B. Evans. Probabilistic 3-D motion estimation for rolling shutter video rectification from visual and inertial measurements. In *Proceedings of the IEEE International Workshop on Multimedia Signal Processing*, Banff, Canada, Sept. 2012.

[43] E. Jones and S. Soatto. Visual-inertial navigation, mapping and localization: A scalable real-time causal approach. *International Journal of Robotics Research*, 30(4):407–430, Apr. 2011.

[44] S. J. Julier. A sparse weight Kalman filter approach to simultaneous localisation and map building. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1251–1256, Maui, HI, Oct. 29-Nov. 3 2001.

[45] S. J. Julier and J. K. Uhlmann. Fusion of time delayed measurements with uncertain time delays. In *Proceedings of the American Control Conference*, pages 4028 – 4033, Portland, OR, June 2005.

[46] M. Kaess, A. Ranganathan, and F. Dellaert. iSAM: Incremental smoothing and mapping. *IEEE Transactions on Robotics*, 24(6):1365 –1378, Dec 2008.

[47] A. Karpenko, D. Jacobs, J. Back, and M. Levoy. Digital video stabilization and rolling shutter correction using gyroscopes. Technical report, Stanford University, 2011.

[48] S. M. Kay. *Fundamentals of Statistical Signal Processing, Vol. 1: Estimation Theory.* Prentice Hall, 1993.

[49] J. Kelly and G. Sukhatme. A general framework for temporal calibration of multiple proprioceptive and exteroceptive sensors. In *Proceedings of the International Symposium of Experimental Robotics*, New Delhi, India, December 2010.

[50] J. Kelly and G. Sukhatme. Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration. *International Journal of Robotics Research*, 30(1):56–79, Jan. 2011.

[51] K. Kimoto and C. Thorpe. Map building with radar and motion sensors for automated highway vehicle navigation. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, volume 3, pages 1721–1728, Grenoble, France, 1997. IEEE.

[52] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *Proceedings of the International Symposium on Mixed and Augmented Reality*, pages 225–234, Nara, Japan, Nov. 2007.

[53] G. Klein and D. Murray. Parallel tracking and mapping on a camera phone. In *Proceedings of the IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 83–86, Orlando, Oct. 2009.

[54] M. Kleinert and S. Schleith. Inertial aided monocular SLAM for GPS-denied navigation. In *Proceedings of the IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pages 20–25, Salt Lake City, UT, Sept. 2010.

[55] K. Konolige and M. Agrawal. FrameSLAM: From bundle adjustment to real-time visual mapping. *IEEE Transactions on Robotics*, 24(5):1066 –1077, Oct. 2008.

[56] K. Konolige, M. Agrawal, and J. Sola. Large-scale visual odometry for rough terrain. In *Proceedings of the International Symposium of Robotics Research*, pages 201–212, Flagstaff, AZ, Nov. 2011.

[57] D. G. Kottas, J. A. Hesch, S. L. Bowman, and S. I. Roumeliotis. On the consistency of vision-aided inertial navigation. In *Proceedings of the International Symposium on Experimental Robotics*, Quebec City, Canada, June 2012.

[58] D. G. Kottas and S. I. Roumeliotis. Exploiting urban scenes for vision-aided inertial navigation. In *Proceedings of Robotics: Science and Systems*, Berlin, Germany, Jul. 2013.

[59] C. Krebs. Generic IMU-camera calibration algorithm: Influence of IMU-axis on each other. Technical report, ETH, December 2012.

[60] J. Langelaan and S. Rock. Passive GPS-free navigation for small UAVs. In *Proceedings of the IEEE Aerospace Conference*, pages 1–9, Big Sky, MT, Mar. 2005.

[61] M. Li, B. Kim, and A. I. Mourikis. Real-time motion estimation on a cellphone using inertial sensing and a rolling-shutter camera. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4697–4704, Karlsruhe, Germany, May 2013.

[62] M. Li and A. I. Mourikis. Consistency of EKF-based visual-inertial odometry. Technical report, University of California Riverside, 2011. www.ee.ucr.edu/∼mourikis/tech_reports/VIO.pdf.

[63] M. Li and A. I. Mourikis. Improving the accuracy of EKF-based visual-inertial odometry. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 828–835, St. Paul, MN, May 2012.

[64] M. Li and A. I. Mourikis. Optimization-based estimator design for vision-aided inertial navigation: Supplemental materials, 2012. www.ee.ucr.edu/∼mli/SupplMaterialsRSS2012.pdf.

[65] M. Li and A. I. Mourikis. Vision-aided inertial navigation for resource-constrained systems. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1057–1063, Vilamoura, Portugal, Oct. 2012.

[66] M. Li and A. I. Mourikis. 3-D motion estimation and online temporal calibration for camera-IMU systems. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 5689–5696, Karlsruhe, Germany, May 2013.

[67] M. Li and A. I. Mourikis. High-precision, consistent EKF-based visual-inertial odometry. *International Journal of Robotics Research*, 32(6):690–711, May 2013.

[68] C.-K. Liang, L.-W. Chang, and H. Chen. Analysis and compensation of rolling shutter effect. *IEEE Transactions on Image Processing*, 17(8):1323 –1330, Aug. 2008.

[69] S. Lovegrove, A. Patron-Perez, and G. Sibley. Spline fusion: A continuous-time representation for visual-inertial fusion with application to rolling-shutter cameras. In *Proceedings of the British Machine Vision Conference*, Bristol, UK, Sept. 2013.

[70] Lumenera Corporation. Rolling shutter timing. `http://www.lumenera.com/support/pdf/LA-2104-ExposureVsShutterTypeTimingAppNote.pdf`, 2005.

[71] T. Lupton and S. Sukkarieh. Efficient integration of inertial observations into visual SLAM without initialization. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1547–1552, St. Louis, MO, Oct. 2009.

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

[73] J. Ma, S. Susca, M. Bajracharya, L. Matthies, M. Malchano, and D. Wooden. Robust multi-sensor, day/night 6-DOF pose estimation for a dynamic legged vehicle in GPS-denied environments. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 619–626, St. Paul, MN, May 2012.

[74] L. Magerand and A. Bartoli. A generic rolling shutter camera model and its application to dynamic pose estimation. In *Proceedings of the International Symposium on 3D Data Processing, Visualization and Transmission*, Parris, France, May 2010.

[75] A. Martinelli. Vision and IMU data fusion: Closed-form solutions for attitude, speed, absolute scale, and bias determination. *IEEE Transactions on Robotics*, 28(1):44 –60, Feb. 2012.

[76] P. S. Maybeck. *Stochastic Models, Estimation and Control*, volume 141-2 of *Mathematics in Science and Engineering*. Academic Press, London, 1982.

[77] M. Meingast, C. Geyer, and S. Sastry. Geometric models of rolling-shutter cameras. In *Proceedings of the 6th Workshop on Omnidirectional Vision, Camera Networks and Non-Classical Cameras*, Beijing, China, Oct. 2005.

[78] F. M. Mirzaei and S. I. Roumeliotis. A Kalman filter-based algorithm for IMU-camera calibration: Observability analysis and performance evaluation. *IEEE Transactions on Robotics*, 24(5):1143–1156, Oct. 2008.

[79] J. Montgomery, A. Johnson, S. I. Roumeliotis, and L. Matthies. The jet propulsion laboratory autonomous helicopter testbed: A platform for planetary exploration technology research and development. *Journal of Field Robotics*, 23(3/4):7245–267, Mar./Apr. 2006.

[80] A. I. Mourikis and S. I. Roumeliotis. A multi-state constraint Kalman filter for vision-aided inertial navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3565–3572, Rome, Italy, Apr. 2007.

[81] A. I. Mourikis and S. I. Roumeliotis. A dual-layer estimator architecture for long-term localization. In *Proceedings of the Workshop on Visual Localization for Mobile Platforms, IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3565–3572, Anchorage, AK, Jun. 2008.

[82] A. I. Mourikis, S. I. Roumeliotis, and J. W. Burdick. SC-KF mobile robot localization: A stochastic cloning-Kalman filter for processing relative-state measurements. *IEEE Transactions on Robotics*, 23(3):717–730, Aug. 2007.

[83] A. I. Mourikis, N. Trawny, S. I. Roumeliotis, A. E. Johnson, A. Ansar, and L. H. Matthies. Vision-aided inertial navigation for spacecraft entry, descent, and landing. *IEEE Transactions on Robotics*, 25(2):264–280, Apr. 2009.

[84] A. I. Mourikis, N. Trawny, S. I. Roumeliotis, A. E. Johnson, and L. H. Matthies. Vision-aided inertial navigation for precise planetary landing: Analysis and experiments. In *Proceedings of Robotics: Science and Systems*, Atlanta, GA, June 26-30 2007.

[85] R. Munguia and A. Grau. Monocular SLAM for visual odometry. In *Proceedings of the IEEE International Symposium on Intelligent Signal Processing*, pages 1–6, Alcala Henares, Spain, Oct. 2007.

[86] E. Nerurkar and S. Roumeliotis. Power-SLAM: a linear-complexity, anytime algorithm for SLAM. *The International Journal of Robotics Research*, 30(6):772–788, May 2011.

[87] T. Oskiper, S. Samarasekera, and R. Kumar. Multi-sensor navigation algorithm using monocular camera, IMU and GPS for large scale augmented reality. In *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality*, pages 71–80, Atlanta, GA, Nov. 2012.

[88] T. Oskiper, Z. Zhiwei, S. Samarasekera, and R. Kumar. Visual odometry system using multiple stereo cameras and inertial measurement unit. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1–8, Minneapolis, MN, Jun. 2007.

[89] L. Oth, P. Furgale, L. Kneip, and R. Siegwart. Rolling shutter camera calibration. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1360–1367, Portland, OR, Jun. 2013.

[90] L. M. Paz, J. D. Tardos, and J. Neira. Divide and conquer: EKF SLAM in O(n). *IEEE Transactions on Robotics*, 24(5):1107 –1120, Oct. 2008.

[91] P. Pinies, T. Lupton, S. Sukkarieh, and J. Tardos. Inertial aiding of inverse depth SLAM using a monocular camera. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2797–2802, Rome, Italy, Apr. 2007.

[92] P. Pinies and J. D. Tardos. Scalable SLAM building conditionally independent local maps. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3466–3471, San Diego, CA, Nov. 2007.

[93] E. Ringaby and P.-E. Forssén. Efficient video rectification and stabilisation for cell-phones. *International Journal of Computer Vision*, 96(3):335–352, 2012.

[94] S. I. Roumeliotis, A. E. Johnson, and J. F. Montgomery. Augmenting inertial navigation with image-based motion estimation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4326–4333, Washington D.C, May 2002.

[95] R. Salas-Moreno, R. Newcombe, H. Strasdat, P. Kelly, and A. Davison. SLAM++: Simultaneous Localisation and Mapping at the Level of Objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1352–1359, Portland, OR, June 2013.

[96] S. Shen and N. Michael. State estimation for indoor and outdoor operation with a micro-aerial vehicle. In *Experimental Robotics*, volume 88 of *Springer Tracts in Advanced Robotics*, pages 273–288. 2013.

[97] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar. Multi-sensor fusion for robust autonomous flight in indoor and outdoor environments with a rotorcraft MAV. *Proceedings of the IEEE International Conference on Robotics and Automation*, May 2014.

[98] F. Shkurti, I. Rekleitis, M. Scaccia, and G. Dudek. State estimation of an underwater robot using visual and inertial information. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5054–5060, San Francisco, CA, Sept. 2011.

[99] G. Shorshi and I. Y. Bar-Itzhack. Satellite autonomous navigation based on magnetic field measurements. *Journal of Guidance, Control, and Dynamics*, 18(4):843–850, 1995.

[100] G. Sibley, L. Matthies, and G. Sukhatme. Sliding window filter with application to planetary landing. *Journal of Field Robotics*, 27(5):587–608, Sep. 2010.

[101] R. Simpson, J. Cullip, and J. Revell. The Cheddar Gorge data set. Technical report, 2011.

[102] I. Skog and P. Haendel. Time synchronization errors in loosely coupled GPS-aided inertial navigation systems. *IEEE Transactions on Intelligent Transportation Systems*, 12(4):1014–1023, 2011.

[103] J. Sola. Consistency of the monocular EKF-SLAM algorithm for three different landmark parametrizations. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3513–3518, Anchorage, AK, May 2010.

[104] H. Strasdat, C. Stachniss, and W. Burgard. Which landmark is useful? Learning selection policies for navigation in unknown environments . In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1410 –1415, Kobe, Japan, May 2009.

[105] D. Strelow. *Motion estimation from image and inertial measurements.* PhD thesis, Carnegie Mellon University, Nov. 2004.

[106] S. Sukkarieh, E. M. Nebot, and H. F. Durrant-Whyte. A high integrity imu/gps navigation loop for autonomous land vehicle applications. *IEEE Transactions on Robotics and Automation*, 15(3):572–578, June 1999.

[107] J. Tardif, M. George, M. Laverne, A. Kelly, and A. Stentz. A new approach to vision-aided inertial navigation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4161–4168, Taibei, China, Oct. 2010.

[108] S. Thrun, Y. Liu, D. Koller, A. Ng, Z. Ghahramani, and H. Durrant-Whyte. Simultaneous localization and mapping with sparse extended information filters. *International Journal of Robotics Research*, 23(7-8):693–716, Aug. 2004.

[109] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney. Stanley, the robot that won the DARPA Grand Challenge. *Journal of Field Robotics*, 23(9):661–692, June 2006.

[110] D. Titterton and J. Weston, editors. *Strapdown Inertial Navigation Technology.* IEE, 2005.

[111] C. H. Tong, P. Furgale, and T. Barfoot. Gaussian Process Gauss-Newton for non-parametric simultaneous localization and mapping. *The International Journal of Robotics Research*, 32(5):507–525, 2013.

[112] N. Trawny, A. I. Mourikis, S. I. Roumeliotis, A. E. Johnson, and J. Montgomery. Vision-aided inertial navigation for pin-point landing using observations of mapped landmarks. *Journal of Field Robotics*, 24(5):357–378, May 2007.

[113] N. Trawny and S. I. Roumeliotis. Indirect Kalman filter for 3D pose estimation. Technical Report 2005-002, Dept. of Computer Science & Engineering, University of Minnesota, Minneapolis, MN, Mar. 2005.

[114] F. Tungadi and L. Kleeman. Time synchronisation and calibration of odometry and range sensors for high-speed mobile robot mapping. In *Proceedings of the Australasian Conference on Robotics and Automation*, Canberra, Australia, December 2010.

[115] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, J. Dolan, D. Duggins, D. Ferguson, T. Galatali, C. Geyer, M. Gittleman, S. Harbaugh, M. Hebert, T. Howard, A. Kelly, D. Kohanbash, M. Likhachev, N. Miller, K. Peterson, R. Rajkumar, P. Rybski, B. Salesky, S. Scherer, Y. Woo-Seo, R. Simmons, S. Singh, J. Snider, A. Stentz, W. R. Whittaker, J. Ziglar, H. Bae, B. Litkouhi, J. Nickolaou, V. Sadekar, S. Zeng, J. Struble, M. Taylor, and M. Darms. Tartan Racing: A multi-modal approach to the DARPA Urban Challenge. Technical report, Defense Advanced Research Projects Agency (DARPA), Apr. 2007.

[116] A. Vu, A. Ramanandan, A. Chen, J. Farrell, and M. Barth. Real-time computer vision/DGPS-aided inertial navigation system for lane-level vehicle navigation. *IEEE Transactions on Intelligent Transportation Systems*, 13(2):899 –913, Jun. 2012.

[117] M. Walter, R. Eustice, and J. Leonard. Exactly sparse extended information filters for feature-based SLAM. *International Journal of Robotics Research*, 26(4):335–359, Apr. 2007.

[118] S. Weiss, M. Achtelik, M. Chli, and R. Siegwart. Versatile distributed pose estimation and sensor self-calibration for an autonomous MAV. In *Proceedings of the IEEE International Conference on Robotics and Automation*, St Paul, MN, May 2012.

[119] S. Weiss, M. Achtelik, S. Lynen, M. Chli, and R. Siegwart. Real-time onboard visual-inertial state estimation and self-calibration of MAVs in unknown environment. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 957–964, St Paul, MN, May 2012.

[120] S. Weiss and R. Siegwart. Real-time metric state estimation for modular vision-Inertial systems. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4531–4537, Shanghai, China, May 2011.

[121] A. Wu, E. Johnson, and A. Proctor. Vision-aided inertial navigation for flight control. *AIAA Journal of Aerospace Computing, Information, and Communication*, 2(9):348–360, Sep. 2005.

[122] T. Yap, M. Li, A. I. Mourikis, and C. Shelton. A particle filter for monocular vision-aided odometry. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 5663–5669, Shanghai, China, May 2011.

[123] D. Zachariah and M. Jansson. Camera-aided inertial navigation using epipolar points. In *Proceedings of the IEEE/ION Symposium on Position Location and Navigation*, pages 303 –309, Indian Wells, CA, May 2010.

[124] D. Zachariah and M. Jansson. Joint calibration of an inertial measurement unit and coordinate transformation parameters using a monocular camera. In *Proceedings of the International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–7, Zurich, September 2010.

[125] K. Zhang, X. R. Li, and Y. Zhu. Optimal update with out-of-sequence-measurements. *IEEE Transactions on Signal Processing*, 53(6):1992–2005, 2005.

[126] S. Zhao, Y. Chen, H. Zhang, and J. A. Farrell. Differential GPS Aided Inertial Navigation: a Contemplative Realtime Approach. In *Proceedings of the 19th IFAC World Congress*, Cape Town, South Africa, Aug. 2014.

Appendix A

A.1 IMU Error-state Transition Matrix

If the biases are included, the IMU error-state transition matrix is given by:

$$\mathbf{\Phi}_{I_k} = \begin{bmatrix} \mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{\Phi}_{\mathbf{qb_g}} & \mathbf{0}_3 \\ \mathbf{\Phi}_{\mathbf{pq}} & \mathbf{I}_3 & \Delta t\mathbf{I}_3 & \mathbf{\Phi}_{\mathbf{pb_g}} & \mathbf{\Phi}_{\mathbf{pa}} \\ \mathbf{\Phi}_{\mathbf{vq}} & \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{\Phi}_{\mathbf{vb_g}} & \mathbf{\Phi}_{\mathbf{va}} \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 \end{bmatrix} \tag{A.1}$$

where

$$\begin{aligned}
\mathbf{\Phi}_{\mathbf{qb_g}} &= -\hat{\mathbf{R}}_k^T \int_{t_k}^{t_{k+1}} {}^{I_k}_{I_\tau}\hat{\mathbf{R}}\; d\tau \\
\mathbf{\Phi}_{\mathbf{pb_g}} &= \int_{t_k}^{t_{k+1}} \int_{t_k}^{w} \lfloor ({}^G\dot{\hat{\mathbf{v}}}_{I_\tau} - {}^G\mathbf{g})\times\rfloor \hat{\mathbf{R}}_k^T \int_{t_k}^{\tau} {}^{I_k}_{I_s}\hat{\mathbf{R}}\; ds\; d\tau\; dw \\
\mathbf{\Phi}_{\mathbf{pa}} &= -\hat{\mathbf{R}}_k^T \int_{t_k}^{t_{k+1}} \int_{t_k}^{\tau} {}^{I_k}_{I_s}\hat{\mathbf{R}}\; ds\; d\tau \\
\mathbf{\Phi}_{\mathbf{vb_g}} &= \int_{t_k}^{t_{k+1}} \lfloor ({}^G\dot{\hat{\mathbf{v}}}_{I_\tau} - {}^G\mathbf{g})\times\rfloor \hat{\mathbf{R}}_k^T \int_{t_k}^{\tau} {}^{I_k}_{I_s}\hat{\mathbf{R}}\; ds\; d\tau \\
\mathbf{\Phi}_{\mathbf{va}} &= -\hat{\mathbf{R}}_k^T \int_{t_k}^{t_{k+1}} {}^{I_k}_{I_\tau}\hat{\mathbf{R}}\; d\tau
\end{aligned} \tag{A.2}$$

The detailed derivation of this result is provided in [62].

A.2 Rank of the Observability Matrix in the "ideal" MSCKF

We here prove that the observability matrix shown in (2.51) has a nullspace of dimension four. To this end, we apply a sequence of elementary column operations on this matrix, to transform it to a different one with the same rank, but which facilitates analysis. In the following, we use the symbol $\sim$ to denote matrices related by elementary column operations. From (2.51) we obtain:

$$\check{\mathcal{O}}_{ik} = \check{\mathbf{M}}_{ik} \left[\lfloor ({}^G\mathbf{p}_{f_i} - {}^G\mathbf{p}_{I_k} - {}^G\mathbf{v}_{I_k}\Delta t_k - \tfrac{1}{2}{}^G\mathbf{g}\Delta t_k^2) \times \rfloor \;\; -\mathbf{I}_3 \;\; \Delta t_k \mathbf{I}_3 \,\middle|\, \mathbf{0}_3 \;\cdots\; \mathbf{I}_3 \;\cdots\; \mathbf{0}_3 \right]$$

where the partitioning denotes the separation between the columns corresponding to the IMU states and those corresponding to the features. We now apply a sequence of elementary column operations, starting by multiplying the second block column by -$\lfloor {}^G\mathbf{p}_{I_k} \times \rfloor$ and adding it to the first block column:

$$\check{\mathcal{O}}_{ik} \sim \check{\mathbf{M}}_{ik} \left[\lfloor ({}^G\mathbf{p}_{f_i} - {}^G\mathbf{v}_{I_k}\Delta t_k - \tfrac{1}{2}{}^G\mathbf{g}\Delta t_k^2) \times \rfloor \;\; -\mathbf{I}_3 \;\; \Delta t_k \mathbf{I}_3 \,\middle|\, \mathbf{0}_3 \;\cdots\; \mathbf{I}_3 \;\cdots\; \mathbf{0}_3 \right]$$

Multiply third block column by $\lfloor {}^G\mathbf{v}_{I_k} \times \rfloor$ and add to first block column

$$\sim \check{\mathbf{M}}_{ik} \left[\lfloor ({}^G\mathbf{p}_{f_i} - \tfrac{1}{2}{}^G\mathbf{g}\Delta t_k^2) \times \rfloor \;\; -\mathbf{I}_3 \;\; \Delta t_k \mathbf{I}_3 \,\middle|\, \mathbf{0}_3 \;\cdots\; \mathbf{I}_3 \;\cdots\; \mathbf{0}_3 \right]$$

Multiply column corresponding to i-th feature by $\lfloor -{}^G\mathbf{p}_{f_i} \times \rfloor$ and add to first block column, $\forall i$

$$\sim \check{\mathbf{M}}_{ik} \left[\lfloor -\tfrac{1}{2}{}^G\mathbf{g}\Delta t_k^2 \times \rfloor \;\; -\mathbf{I}_3 \;\; \Delta t_k \mathbf{I}_3 \,\middle|\, \mathbf{0}_3 \;\cdots\; \mathbf{I}_3 \;\cdots\; \mathbf{0}_3 \right]$$

Multiply first block column by -2

$$\sim \check{\mathbf{M}}_{ik} \left[\Delta t_k^2 \lfloor {}^G\mathbf{g} \times \rfloor \;\; -\mathbf{I}_3 \;\; \Delta t_k \mathbf{I}_3 \,\middle|\, \mathbf{0}_3 \;\cdots\; \mathbf{I}_3 \;\cdots\; \mathbf{0}_3 \right]$$

Add all block columns corresponding to the features to the second block column

$$\sim \check{\mathbf{M}}_{ik} \left[\begin{array}{ccc|ccccc} \Delta t_k^2 \lfloor {}^G\mathbf{g} \times \rfloor & \mathbf{0}_3 & \Delta t_k \mathbf{I}_3 & \mathbf{0}_3 & \cdots & \mathbf{I}_3 & \cdots & \mathbf{0}_3 \end{array} \right]$$

We now define the unitary matrix

$$\mathbf{F} = \begin{bmatrix} \mathbf{g}_{p_1} & \mathbf{g}_{p_2} & \frac{{}^G\mathbf{g}}{||{}^G\mathbf{g}||_2} \end{bmatrix}$$

where the two unit vectors $\mathbf{g}_{p_1}$ and $\mathbf{g}_{p_2}$ are on the plane perpendicular to ${}^G\mathbf{g}$, and are chosen to form an orthogonal coordinate system. Since $\mathbf{F}$ is non-singular, we can multiply the first block column of the above expression by $\mathbf{F}$ to obtain:

$$\check{\mathcal{O}}_{ik} \sim \check{\mathbf{M}}_{ik} \left[\begin{array}{cccc|ccccc} \Delta t_k^2 \mathbf{G} & \mathbf{0}_{3\times 1} & \mathbf{0}_3 & \Delta t_k \mathbf{I}_3 & \mathbf{0}_3 & \cdots & \mathbf{I}_3 & \cdots & \mathbf{0}_3 \end{array} \right] \tag{A.3}$$

where

$$\mathbf{G} = ||{}^G\mathbf{g}||_2 \begin{bmatrix} \mathbf{g}_{p_1} & \mathbf{g}_{p_2} \end{bmatrix}$$

At this point, we note that through a sequence of elementary column operations, all block rows of the matrix $\check{\mathcal{O}}$ have been transformed to a form where the third to sixth columns are all zero. Thus, the matrix $\check{\mathcal{O}}$ is rank deficient by *at least four*, and the zero columns can be omitted without changing the rank:

$$\check{\mathcal{O}}_{ik} \sim \check{\mathbf{M}}_{ik} \left[\begin{array}{cc|ccccc} \Delta t_k^2 \mathbf{G} & \Delta t_k \mathbf{I}_3 & \mathbf{0}_3 & \cdots & \mathbf{I}_3 & \cdots & \mathbf{0}_3 \end{array} \right] = \check{\mathbf{T}}_{ik} \tag{A.4}$$

To show that the matrix $\check{\mathcal{O}}$ is rank deficient by *exactly* four, we need to show that the matrix with block rows $\check{\mathbf{T}}_{ik}$ has full column rank. To this end, we define a vector

$$\mathbf{a} = \begin{bmatrix} \mathbf{a}_q \\ -\mathbf{a}_v \\ \mathbf{a}_1 \\ \mathbf{a}_2 \\ \vdots \\ \mathbf{a}_s \end{bmatrix} \tag{A.5}$$

and it suffices to show that the condition $\check{\mathbf{T}}_{ik}\mathbf{a} = \mathbf{0}, \forall i, k$, is satisfied only if $\mathbf{a} = \mathbf{0}$. Substituting from (A.4) and (A.5) we obtain:

$$\begin{aligned} \check{\mathbf{T}}_{ik}\mathbf{a} = \mathbf{0}, \forall i, k &\Rightarrow \check{\mathbf{M}}_{ik}\left(\Delta t_k^2 \mathbf{G}\mathbf{a}_q - \Delta t_k \mathbf{a}_v + \mathbf{a}_i\right) = \mathbf{0}, \forall i, k \\ &\Rightarrow \begin{bmatrix} {}^{C_k}z_{f_i} & 0 & -{}^{C_k}x_{f_i} \\ 0 & {}^{C_k}z_{f_i} & -{}^{C_k}y_{f_i} \end{bmatrix} {}_I^C\mathbf{R}\mathbf{R}_i\left(\Delta t_k^2 \mathbf{G}\mathbf{a}_q - \Delta t_k \mathbf{a}_v + \mathbf{a}_i\right) = \mathbf{0}, \forall i, k \end{aligned}$$

The above equation is equivalent to the following:

$${}_I^C\mathbf{R}\mathbf{R}_k\left(\Delta t_k^2 \mathbf{G}\mathbf{a}_q - \Delta t_k \mathbf{a}_v + \mathbf{a}_i\right) \in \mathcal{N}\left(\begin{bmatrix} {}^{C_k}z_{f_i} & 0 & -{}^{C_k}x_{f_i} \\ 0 & {}^{C_k}z_{f_i} & -{}^{C_k}y_{f_i} \end{bmatrix}\right), \quad \forall i, k \tag{A.6}$$

Thus, we can write

$${}_I^C\mathbf{R}\mathbf{R}_k\left(\Delta t_k^2 \mathbf{G}\mathbf{a}_q - \Delta t_k \mathbf{a}_v + \mathbf{a}_i\right) = c_{ik}\,{}^{C_k}\mathbf{p}_i, \quad \forall i, k \tag{A.7}$$

for some scalars c_{ik}. Using (2.8), we obtain

$$\Delta t_k^2 \mathbf{G}\mathbf{a}_q - \Delta t_k \mathbf{a}_v + \mathbf{a}_i = c_{ik}({}^G\mathbf{p}_i - {}^G\mathbf{p}_{C_k}), \quad \forall i, k \tag{A.8}$$

Note that the above condition can be interpreted as a condition on the motion of the camera. For example, if the camera is moving with a constant acceleration $\mathbf{G}\mathbf{a}_q$, initial velocity $\mathbf{a}_v$ and initial position ${}^G\mathbf{p}_0$, and we choose $c_{ik} = -1$, and

$\mathbf{a}_i = {}^G\mathbf{p}_0 - {}^G\mathbf{p}_i$, then the above conditions will be satisfied. However, for general camera motion, and when multiple features are observed, the above condition cannot be met for nonzero values of $c_{ik}, \mathbf{a}_q, \mathbf{a}_v, \mathbf{a}_i$ [83]. Thus, for general camera motion, $\check{\mathbf{T}}_{ik}\mathbf{a} = \mathbf{0}, \forall i, k$, requires $\mathbf{a} = \mathbf{0}$, which shows that the matrix with block rows $\check{\mathbf{T}}_{ik}$ has full column rank. This completes the proof.

A.3 Nullspace Physical Interpretation

We have shown that the nullspace of the observability matrix in the "ideal" MSCKF is of dimension four, and is spanned by the column vectors of the matrix in (2.54). If we write (2.54) as $\mathbf{N} = [\,\mathbf{n}_1 \;\; \mathbf{n}_2 \;\; \mathbf{n}_3 \;\; \mathbf{n}_4\,]$, then:

$$\mathcal{N}\left(\check{\mathcal{O}}\right) = \mathsf{span}\,[\,\mathbf{n}_1 \quad \mathbf{n}_2 \quad \mathbf{n}_3 \quad \mathbf{n}_4\,] \tag{A.9}$$

To gain a better understanding of the physical interpretation of the basis of $\mathcal{N}\left(\check{\mathcal{O}}\right)$, let us examine what changes in the state each of the four vectors $\mathbf{n}_i$ corresponds to. First, note that if, starting from an initial state $\mathbf{x}$, we modify it as $\mathbf{x}' = \mathbf{x} + c_1\mathbf{n}_1 + c_2\mathbf{n}_2 + c_3\mathbf{n}_3$, then the state $\mathbf{x}'$ will have the same values for the IMU orientation and velocity, but the position of the IMU and the positions of all features will be changed by the vector $[c_1 \;\; c_2 \;\; c_3]^T$. Thus, the first three columns in $\mathbf{N}$ correspond to shifts of the entire state vector. On the other hand, if we rotate the state vector $\mathbf{x}$ by a small angle, c, about gravity, we can write the resulting state as

$$\mathbf{x}' = \begin{bmatrix} {}^{I}_{G}\bar{\mathbf{q}}' \\ {}^{G}\mathbf{p}'_I \\ {}^{G}\mathbf{v}'_I \\ {}^{G}\mathbf{p}'_{f_1} \\ \vdots \\ {}^{G}\mathbf{p}'_{f_s} \end{bmatrix} = \begin{bmatrix} {}^{I}_{G}\bar{\mathbf{q}} \otimes {}^{G}_{G'}\bar{\mathbf{q}} \\ \mathbf{Diag}\left({}^{G'}_{G}\mathbf{R}\right) \begin{bmatrix} {}^{G}\mathbf{p}_I \\ {}^{G}\mathbf{v}_I \\ {}^{G}\mathbf{p}_{f_1} \\ \vdots \\ {}^{G}\mathbf{p}_{f_s} \end{bmatrix} \end{bmatrix} \tag{A.10}$$

where the rotation matrix ${}^{G'}_{G}\mathbf{R}$ expresses the applied rotation, and $\mathbf{Diag}(\cdot)$ denotes a block diagonal matrix. To show that $\mathbf{n}_4$ corresponds to rotations about gravity, we will show that the difference between $\mathbf{x}$ and $\mathbf{x}'$ can be written, to a first-order approximation, as a multiple of $\mathbf{n}_4$. We start by noting that, since the rotation angle is small, ${}^{G'}_{G}\mathbf{R}$ can be approximated as:

$${}^{G'}_{G}\mathbf{R} \simeq \mathbf{I}_3 - c\lfloor {}^{G}\bar{\mathbf{g}} \times \rfloor \tag{A.11}$$

where ${}^{G}\bar{\mathbf{g}}$ is the unit vector along gravity. Using this result, we can write:

$$\begin{bmatrix} {}^{G}\mathbf{p}_I \\ {}^{G}\mathbf{v}_I \\ {}^{G}\mathbf{p}_{f_1} \\ \vdots \\ {}^{G}\mathbf{p}_{f_s} \end{bmatrix} - \begin{bmatrix} {}^{G}\mathbf{p}'_I \\ {}^{G}\mathbf{v}'_I \\ {}^{G}\mathbf{p}'_{f_1} \\ \vdots \\ {}^{G}\mathbf{p}'_{f_s} \end{bmatrix} \simeq \begin{bmatrix} {}^{G}\mathbf{p}_I \\ {}^{G}\mathbf{v}_I \\ {}^{G}\mathbf{p}_{f_1} \\ \vdots \\ {}^{G}\mathbf{p}_{f_s} \end{bmatrix} - \left(\mathbf{I}_3 - c\lfloor {}^{G}\bar{\mathbf{g}} \times \rfloor\right) \begin{bmatrix} {}^{G}\mathbf{p}_I \\ {}^{G}\mathbf{v}_I \\ {}^{G}\mathbf{p}_{f_1} \\ \vdots \\ {}^{G}\mathbf{p}_{f_s} \end{bmatrix} = \frac{c}{||\mathbf{g}||_2} \begin{bmatrix} -\lfloor {}^{G}\mathbf{p}_I \times \rfloor \mathbf{g} \\ -\lfloor {}^{G}\mathbf{v}_I \times \rfloor \mathbf{g} \\ -\lfloor {}^{G}\mathbf{p}_{f_1} \times \rfloor \mathbf{g} \\ -\lfloor {}^{G}\mathbf{p}_{f_2} \times \rfloor \mathbf{g} \\ \vdots \\ -\lfloor {}^{G}\mathbf{p}_{f_s} \times \rfloor \mathbf{g} \end{bmatrix} \tag{A.12}$$

Moreover, if we denote by $\delta\boldsymbol{\theta}$ the orientation difference between $\mathbf{x}$ and $\mathbf{x}'$ in (A.10), we obtain

$$\begin{aligned} {}^{I}_{G'}\mathbf{R} &\simeq {}^{I}_{G}\mathbf{R}(\mathbf{I}_3 - \lfloor \delta\boldsymbol{\theta} \times \rfloor) && \text{(A.13)} \\ &= {}^{I}_{G'}\mathbf{R}\left(\mathbf{I}_3 + c\lfloor {}^{G}\bar{\mathbf{g}} \times \rfloor\right)(\mathbf{I}_3 - \lfloor \delta\boldsymbol{\theta} \times \rfloor) && \text{(A.14)} \\ &\simeq {}^{I}_{G'}\mathbf{R} - {}^{I}_{G'}\mathbf{R}\lfloor \delta\boldsymbol{\theta} \times \rfloor + c\, {}^{I}_{G'}\mathbf{R}\lfloor {}^{G}\bar{\mathbf{g}} \times \rfloor && \text{(A.15)} \end{aligned}$$

From the last expression we obtain $\lfloor \delta\boldsymbol{\theta} \times \rfloor = c \lfloor {}^G\bar{\mathbf{g}} \times \rfloor$, and thus

$$\delta\boldsymbol{\theta} = c \cdot {}^G\bar{\mathbf{g}} \tag{A.16}$$

$$= \frac{c}{||\mathbf{g}||_2} {}^G\mathbf{g} \tag{A.17}$$

The results of (A.17) and (A.12) show that if we apply a small rotation about gravity to obtain $\mathbf{x}'$ from $\mathbf{x}$, the difference between the two states is given by $\frac{c}{||\mathbf{g}||_2}\mathbf{n}_4$.

A.4 Rank of the MSCKF Observability Matrix

In this section, we prove that the dimension of the nullspace of the MSCKF observability matrix is 3. Similarly to the analysis in Appendix A.2, we apply the same sequence of elementary column operations to transform each block row of the observability matrix in (2.55) into:

$$\mathcal{O}_{ik} \sim \mathbf{M}_{ik} \left[\begin{array}{ccc|ccccc} \Delta t_k^2 \lfloor {}^G\mathbf{g} \times \rfloor + \Delta\boldsymbol{\Gamma}_k^{(i)} & \mathbf{0}_3 & \Delta t_k \mathbf{I}_3 & \mathbf{0}_3 & \cdots & \mathbf{I}_3 & \cdots & \mathbf{0}_3 \end{array} \right] \tag{A.18}$$

At this point, we see that the fourth to sixth columns of the matrix $\mathcal{O}_{ik}$ are all zero, which indicates that the dimension of the nullspace of the MSCKF observability matrix is at least three. By omitting zero columns, (A.18) becomes:

$$\mathcal{O}_{ik} \sim \mathbf{M}_{ik} \left[\begin{array}{cc|ccccc} \Delta t_k^2 \lfloor {}^G\mathbf{g} \times \rfloor + \Delta\boldsymbol{\Gamma}_{ik} & \Delta t_k \mathbf{I}_3 & \mathbf{0}_3 & \cdots & \mathbf{I}_3 & \cdots & \mathbf{0}_3 \end{array} \right] = \mathbf{T}_{ik} \tag{A.19}$$

To prove that $\mathcal{O}$ is rank deficient by three, we need to show that the matrix with block rows $\mathbf{T}_{ik}$ has full column rank. We start by defining:

$$\mathbf{Y}_{ik} = \mathbf{M}_{ik} \left[\begin{array}{c|ccccc} \Delta t_k \mathbf{I}_3 & \mathbf{0}_3 & \cdots & \mathbf{I}_3 & \cdots & \mathbf{0}_3 \end{array} \right] \tag{A.20}$$

The matrix with block rows $\mathbf{Y}_{ik}$ has full column rank, as shown in Appendix A.2. Next, we observe that the terms $\Delta\mathbf{\Gamma}_{ik}$, which appear in the first block column in the matrix (A.19), are random terms. This implies that the three columns of this matrix are linearly independent of the columns of $\mathbf{Y}_{ik}$. This completes the proof.

A.5 Observability Properties of the EKF SLAM System Model

In EKF-based SLAM, the current feature estimates are used for computing the measurement Jacobians at each time step. Thus, we have:

$$\begin{aligned}
\mathbf{H}_{I_{ik}} &= \mathbf{J}_{ik}\,{}^{C}_{I}\mathbf{R}\,\hat{\mathbf{R}}_{k|k-1}\big[\lfloor({}^{G}\hat{\mathbf{p}}_{f_{i|k-1}} - {}^{G}\hat{\mathbf{p}}_{I_{k|k-1}})\times\rfloor \;\; -\mathbf{I}_3 \;\; \mathbf{0}_3\big] \\
\mathbf{H}_{\mathbf{f}_{ik}} &= \mathbf{J}_{ik}\,{}^{C}_{I}\mathbf{R}\,\hat{\mathbf{R}}_{k|k-1}
\end{aligned} \tag{A.21}$$

where the matrix $\mathbf{J}_{ik}$ is evaluated using the estimate:

$$ {}^{C_k}\hat{\mathbf{p}}_{f_{i|k-1}} = {}^{C}_{I}\mathbf{R}\,\hat{\mathbf{R}}_{k|k-1}\big({}^{G}\hat{\mathbf{p}}_{f_{i|k-1}} - {}^{G}\hat{\mathbf{p}}_{I_{k|k-1}}\big) + {}^{C}\mathbf{p}_I \tag{A.22}$$

Using these Jacobians, we obtain the block row of the observability matrix corresponding to the observation of feature i at time-step k. This matrix has the same structure as (2.55), with:

$$\begin{aligned}
\boldsymbol{\Gamma}_{ik} &= \lfloor {}^G\hat{\mathbf{p}}_{f_{i|k-1}} - {}^G\hat{\mathbf{p}}_{I_{b|b}} - {}^G\hat{\mathbf{v}}_{I_{b|b}}\Delta t_k - \frac{1}{2}\mathbf{g}\Delta t_k^2 \times \rfloor \\
\Delta\boldsymbol{\Gamma}_{ik} &= \sum_{j=b+1}^{k-1}(\mathbf{E}_{\mathbf{p}}^j + \sum_{s=b+1}^{j}\mathbf{E}_{\mathbf{v}}^s\Delta t) + \lfloor \Delta^G\mathbf{p}_{f_i} \times \rfloor \qquad \text{(A.23)} \\
\mathbf{M}_{ik} &= \mathbf{J}_{ik}\,{}_I^C\mathbf{R}\,\hat{\mathbf{R}}_{k|k}
\end{aligned}$$

$$\Delta^G\mathbf{p}_{f_i} = \lfloor {}^G\hat{\mathbf{p}}_{f_{i|k-1}} - {}^G\hat{\mathbf{p}}_{f_{i|b-1}} \times \rfloor \qquad \text{(A.24)}$$

$$\mathbf{E}_{\mathbf{p}}^j = \lfloor {}^G\hat{\mathbf{p}}_{I_{j|j-1}} - {}^G\hat{\mathbf{p}}_{I_{j|j}} \times \rfloor \qquad \text{(A.25)}$$

$$\mathbf{E}_{\mathbf{v}}^j = \lfloor {}^G\hat{\mathbf{v}}_{I_{j|j-1}} - {}^G\hat{\mathbf{v}}_{I_{j|j}} \times \rfloor \qquad \text{(A.26)}$$

Similarly to the MSCKF, the observability matrix of EKF-based SLAM also contains a disturbance term $\Delta\boldsymbol{\Gamma}_{ik}$, which decreases the dimension of the unobservable subspace.

A.6 Generating Datasets for Monte-Carlo Simulations

We here describe our method for generating realistic simulation environments for Monte-Carlo simulation tests. In what follows, we first describe our approach for computing the ground truth trajectory used for Monte-Carlo trials, and subsequently, we explain that how we generate visual features and sensor measurements in each trial.

We first process a real-world dataset using the MSCKF 2.0 algorithm, to obtain estimates for the evolving IMU state, $\hat{\mathbf{x}}_E$, at each timestep. For the simulation ground-truth trajectory, we set the IMU poses (position and orientation) at the time instants at which images are available, τ_j, $j = 1, \ldots, n_1$, to be identical to the computed estimates $\hat{\mathbf{x}}_E(\tau_j)$. To obtain the complete ground truth, we must additionally determine the IMU states for all the time instants t_i, $i = 1, \ldots, n_2$, at which the IMU is sampled, as well as the rotational velocity and linear acceleration at these time instants (these are needed for generating IMU measurements in the simulator). To this end we formulate an optimization problem, to determine the acceleration and rotational velocity signals which will (a) guarantee that the IMU pose at the time instants τ_j is identical to the estimates $\hat{\mathbf{x}}_E(\tau_j)$, and (b) minimize the difference between the ground-truth acceleration and rotational velocity and the corresponding estimates obtained from the actual data set (see (2.30) and (2.33)). To describe this minimization problem, let us denote the vector of ground-truth quantities we seek to determine at time t_i as

$$\mathbf{x}_M^\star(t_i) = \begin{bmatrix} {}^I_G\bar{\mathbf{q}}^\star(t_i) \\ {}^G\mathbf{p}_I^\star(t_i) \\ {}^G\mathbf{v}_I^\star(t_i) \\ {}^G\mathbf{a}_I^\star(t_i) \\ {}^I\boldsymbol{\omega}^\star(t_i) \end{bmatrix} = \begin{bmatrix} \mathbf{x}_s^\star(t_i) \\ {}^G\mathbf{a}_I^\star(t_i) \\ {}^I\boldsymbol{\omega}^\star(t_i) \end{bmatrix} \tag{A.27}$$

where $\mathbf{x}_s^\star = [{}^I_G\bar{\mathbf{q}}^\star(t_i)^T, {}^G\mathbf{p}_I^\star(t_i)^T, {}^G\mathbf{v}_I^\star(t_i)^T]^T$. Moreover, we denote by ψ the IMU-state propagation function, computed by numerical integration assuming linearly changing acceleration and rotational velocity in each IMU sample interval. The ground truth is obtained by formulating a minimization problem for each interval between two consecutive image timestamps, $j-1$ and j. Specifically, we obtain $\mathbf{x}_M^\star(t_i)$, $t_i \in (\tau_{j-1}, \tau_j]$ by solving the constrained minimization problem:

$$\begin{aligned} \min & \sum_{t_i \in (\tau_{j-1}, \tau_j]} \|{}^I\boldsymbol{\omega}^\star(t_i) - {}^I\hat{\boldsymbol{\omega}}(t_i)\|_2^2 + \|{}^G\mathbf{a}_I^\star(t_i) - {}^G\hat{\mathbf{a}}_I(t_i)\|_2^2 \\ \text{s.t.} \quad & \mathbf{x}_s^\star(t_i) = \psi(\mathbf{x}_s^\star(t_{i-1}), {}^G\mathbf{a}_I^\star(t_{i-1}, t_i), {}^I\boldsymbol{\omega}^\star(t_{i-1}, t_i)) \\ & {}^G\mathbf{p}_I^\star(\tau_j) = {}^G\hat{\mathbf{p}}_I(\tau_j),\ {}^I_G\bar{\mathbf{q}}^\star(\tau_j) = {}^I_G\hat{\bar{\mathbf{q}}}(\tau_j) \end{aligned} \tag{A.28}$$

where ${}^I\hat{\boldsymbol{\omega}}$ and ${}^G\hat{\mathbf{a}}_I$ are the rotational velocity and linear acceleration computed using the EKF estimates and the IMU measurements in the real dataset, via (2.30) and (2.33). The above problem is solved sequentially for each interval $(\tau_{j-1}, \tau_j]$. The resulting ground truth is self-consistent (in the sense that integrating the ground-truth acceleration yields the ground truth velocity, integrating the velocity yields position, and so on), and closely matches the estimates obtained in the actual dataset.

The ground-truth trajectory constructed as described above is used in all Monte-Carlo trials. In each trial, different independently sampled realizations for the IMU biases and measurement noise, the feature positions, and image measurement noise, are used. Specifically, in each trial IMU biases at each timestep are generated by:

$$\mathbf{b}_{\mathbf{g}_{k+1}} = \mathbf{b}_{\mathbf{g}_k} + \mathbf{n}_{\mathbf{wg}_k} \tag{A.29}$$

$$\mathbf{b}_{\mathbf{a}_{k+1}} = \mathbf{b}_{\mathbf{a}_k} + \mathbf{n}_{\mathbf{wa}_k} \tag{A.30}$$

and IMU measurements are computed via (2.28) and (2.29). The noise vectors used in each timestep in (A.29), (A.30), (2.28), and (2.29) are independently sampled from Gaussian distributions with characteristics identical to those of the IMU used in the actual dataset. In each simulated image, we generate new features, whose number and track-length distribution is identical to the corresponding actual image. The 3D position of new features is randomly drawn in each trial. Specifically, the depth of each feature is chosen equal to the estimated depth of the corresponding actual feature, while the location of its image projection is randomly sampled from a uniform distribution. Finally, each image measurement is corrupted by independently sampled Gaussian noise.

Appendix B

B.1 SLAM Feature Parametrization

In this section, we describe the feature parameterization used for the SLAM features in the MSCKF/SLAM hybrid filter. Specifically, we employ an anchored inverse-depth parameterization for the SLAM features. The inverse-depth parameters of the i-th feature are defined as:

$$\mathbf{f}_i = \begin{bmatrix} \alpha_i & \beta_i & \rho_i \end{bmatrix}^T \tag{B.1}$$

Using (B.1), the global position of the i-th feature, ${}^G\mathbf{p}_{f_i}$, is given by:

$${}^G\mathbf{p}_{f_i} = {}^G\mathbf{p}_{I_i} + \frac{1}{\rho_i}\mathbf{R}_i \begin{bmatrix} \alpha_i \\ \beta_i \\ 1 \end{bmatrix} \tag{B.2}$$

where ${}^G\mathbf{p}_{I_i}$ is the global position of the IMU pose in the sliding window that is selected as the anchor of this feature, and $\mathbf{R}_i$ is a constant matrix. The anchor state as well as the constant matrix $\mathbf{R}_i$ are determined during the feature initialization process. Specifically, at the time of feature initialization, we choose the anchor to be the latest IMU state added to the sliding window (which allows the anchor to be kept in the state vector for a longer time), and set $\mathbf{R}_i = {}^G_{I_i}\hat{\mathbf{R}}\,{}^I_C\mathbf{R}$. The key advantage of our feature parametrization is that, since the anchor position is chosen as the position of an IMU state that is already in the state vector, we *only* need to include the 3×1 vector $\mathbf{f}_i$ in the state vector, to represent the feature. This is in contrast to the traditional inverse depth parametrization [18], which includes both an anchor and an inverse-depth parameter vector for each individual feature in the state vector.

B.2 SLAM Feature Re-parameterization

To keep the computational cost of the hybrid filter bounded, the IMU poses whose corresponding MSCKF features have all been processed, will be removed from the state vector, as described in Algorithm 2. Therefore, it is possible that in certain situations the anchor of a SLAM feature will be removed from the state vector. When this occurs, we re-parameterize this feature as $\mathbf{f}'_i$, and select the position of the current latest IMU, ${}^G\mathbf{p}'_{I_i}$, as the new anchor. To compute $\mathbf{f}'_i$, we note that:

$$
{}^G\mathbf{p}_{f_i} = {}^G\mathbf{p}_{I_i} + \frac{1}{\rho_i}\mathbf{R}_i \begin{bmatrix} \alpha_i \\ \beta_i \\ 1 \end{bmatrix} = {}^G\mathbf{p}'_{I_i} + \frac{1}{\rho'_i}\mathbf{R}'_i \begin{bmatrix} \alpha'_i \\ \beta'_i \\ 1 \end{bmatrix} \tag{B.3}
$$

$$
\Longrightarrow \qquad \frac{1}{\rho'_i}\begin{bmatrix} \alpha'_i \\ \beta'_i \\ 1 \end{bmatrix} = (\mathbf{R}'_i)^T \left(-{}^G\mathbf{p}'_{I_i} + {}^G\mathbf{p}_{I_i} + \mathbf{R}_i \cdot \frac{1}{\rho_i} \begin{bmatrix} \alpha_i \\ \beta_i \\ 1 \end{bmatrix} \right) \tag{B.4}
$$

where $\mathbf{R}'_i = {}^G_{I_i}\hat{\mathbf{R}}' \, {}^I_C\mathbf{R}$ is a constant matrix. Using the above equation, the re-parameterized feature vector $\mathbf{f}'_i$ and its covariance matrix can be directly computed.

Appendix C

C.1 Appendix: Proof of Lemma 2

We start by performing a change of variables, defining $\bar{\mathbf{v}} = s^{C_o}\mathbf{v}_o$, $\bar{\mathbf{g}} = s^{C_o}\mathbf{g}$, $\bar{\mathbf{b}} = -s^C_I\mathbf{R}\mathbf{b}_{\mathbf{a}}$, and $\bar{\mathbf{p}} = -s^C\mathbf{p}_I$, to obtain:

$$\mathbf{c}_2(\bar{\boldsymbol{\xi}}, t) = \bar{\mathbf{v}} + \bar{\mathbf{g}}t + s\int_0^t \mathbf{R}_c(\tau)\,{}^C_I\mathbf{R}\,\mathbf{a}_m(\tau + t_d)d\tau \\ + \int_0^t \mathbf{R}_c(\tau)d\tau\,\bar{\mathbf{b}} + \dot{\mathbf{R}}_c(t)\bar{\mathbf{p}} - \mathbf{v}_c(t) = \mathbf{0} \tag{C.1}$$

with

$$\bar{\boldsymbol{\xi}} = \begin{bmatrix}\bar{\mathbf{v}}^T & \bar{\mathbf{g}}^T & \bar{\mathbf{b}}^T & \mathbf{b_g}^T & \bar{\mathbf{p}}^T & {}^C_I\mathbf{q}^T & t_d & s\end{bmatrix}^T$$

Note that since t_d, $\mathbf{b_g}$, and ${}^C_I\bar{\mathbf{q}}$ remain the same in the change from $\boldsymbol{\xi}$ to $\bar{\boldsymbol{\xi}}$, the condition $\mathbf{c}_1$ in (4.24) also holds for $\bar{\boldsymbol{\xi}}$ with no modification.

Clearly, $\boldsymbol{\xi}$ is locally identifiable if and only if $\bar{\boldsymbol{\xi}}$ is. A sufficient condition for identifiability is that there exists a set of time instants, such that if we evaluate the matrix containing the Jacobians of $\mathbf{c}_1$ and $\mathbf{c}_2$ at these time instants, the matrix has full column rank. In turn, a sufficient condition for this is that there exists no nonzero vector $\mathbf{m}$ such that, for all $t > 0$, it is:

$$\begin{bmatrix} \mathbf{D}_1(t) \\ \mathbf{D}_2(t) \end{bmatrix} \mathbf{m} = \mathbf{0} \;\Leftrightarrow\; \mathbf{D}_1(t)\mathbf{M} = \mathbf{0} \text{ and } \mathbf{D}_2(t)\mathbf{M} = \mathbf{0} \tag{C.2}$$

where $\mathbf{D}_1(t)$ is the Jacobian of $\mathbf{c}_1$ with respect to $\bar{\boldsymbol{\xi}}$, and $\mathbf{D}_2(t)$ is the Jacobian of $\mathbf{c}_2$ with respect to $\bar{\boldsymbol{\xi}}$. We now introduce the partitioning

$$\mathbf{m} = \begin{bmatrix} \mathbf{M}_1^T & \mathbf{M}_2^T & \mathbf{M}_3^T & \mathbf{k}_1^T & \mathbf{M}_4^T & \mathbf{k}_2^T & 1 & m_5 \end{bmatrix}^T$$

where the element corresponding to the Jacobians with respect to t_d has been set to one, as we are interested in detecting cases in which t_d is not identifiable.

The condition $\mathbf{D}_1(t)\mathbf{M} = \mathbf{0}$ is identical to the condition that was encountered in the proof of Lemma 1, and yields the first condition of Lemma 2. The second condition of Lemma 2 is derived from the equation $\mathbf{D}_2(t)\mathbf{M} = \mathbf{0}$. Computing the Jacobians of $\mathbf{c}_2$ with respect to $\bar{\boldsymbol{\xi}}$, and substituting in $\mathbf{D}_2(t)\mathbf{M} = \mathbf{0}$, we obtain:

$$\begin{aligned} \mathbf{M}_1 + t\mathbf{M}_2 + \int_0^t \mathbf{R}_c(\tau)d\tau\,\mathbf{M}_3 + \dot{\mathbf{R}}_c(t)\mathbf{M}_4 + s\int_0^t \mathbf{R}_c(\tau)_I^C\mathbf{R}\lfloor \mathbf{a}_m(\tau+t_d)\times\rfloor d\tau\mathbf{k}_2 \\ + s\int_0^t \mathbf{R}_c(\tau)_I^C\mathbf{R}\dot{\mathbf{a}}_m(\tau+t_d)d\tau + \int_0^t \mathbf{R}_c(\tau)_I^C\mathbf{R}\mathbf{a}_m(\tau+t_d)d\tau m_5 = \mathbf{0}, \quad \forall t > 0 \end{aligned}$$

Differentiating this expression with respect to t, and re-arranging terms, yields:

$$\dot{\mathbf{a}}_m(t+t_d) = \left(\lfloor \mathbf{k}_2\times\rfloor - \frac{m_5}{s}\mathbf{I}_3\right)\mathbf{a}_m(t+t_d) - \frac{1}{s}\mathbf{g}(t), \quad \forall t > 0$$

or, equivalently,

$$\dot{\mathbf{a}}_m(t) = \left(\lfloor \mathbf{k}_2\times\rfloor - \frac{m_5}{s}\mathbf{I}_3\right)\mathbf{a}_m(t) - \frac{1}{s}\mathbf{g}(t-t_d), \quad \forall t > 0 \tag{C.3}$$

where

$$\begin{aligned}
\mathbf{g}(t-t_d) &= {}^C_I\mathbf{R}^T\mathbf{R}_c(t-t_d)^T\mathbf{M}_2 + {}^C_I\mathbf{R}^T\mathbf{M}_3 + {}^C_I\mathbf{R}^T\mathbf{R}_c^T(t-t_d)\ddot{\mathbf{R}}_c(t-t_d)\mathbf{M}_4 \\
&= {}^{I_o}_I\mathbf{R}(t)^T{}^C_I\mathbf{R}^T\mathbf{M}_2 + {}^C_I\mathbf{R}^T\mathbf{M}_3 + {}^{I_o}_I\mathbf{R}(t)^T{}^{I_o}_I\ddot{\mathbf{R}}(t){}^C_I\mathbf{R}^T\mathbf{M}_4 \\
&= {}^{I_o}_I\mathbf{R}(t)^T{}^C_I\mathbf{R}^T\mathbf{M}_2 + {}^C_I\mathbf{R}^T\mathbf{M}_3 + \left(\lfloor{}^I\boldsymbol{\omega}(t)\times\rfloor^2 + \lfloor{}^I\dot{\boldsymbol{\omega}}(t)\times\rfloor\right){}^C_I\mathbf{R}^T\mathbf{M}_4
\end{aligned}$$

Substituting the last expression in (C.3), and defining $k_3 = m_5/s$, $\mathbf{k}_4 = \frac{1}{s}{}^C_I\mathbf{R}^T\mathbf{M}_3$, $\mathbf{k}_5 = \frac{1}{s}{}^C_I\mathbf{R}^T\mathbf{M}_4$ and $\mathbf{k}_6 = \frac{1}{s}{}^C_I\mathbf{R}^T\mathbf{M}_2$, yields (4.43).

Appendix D

D.1 Upper Bounds on the Unmodeled Residual Terms

In this section, we show how the upper bounds on the unmodeled residual terms can be computed. We start by presenting the camera measurement model and calculating the measurement Jacobian matrices. To simplify the theoretical analysis, we here ignore the radial/tangential distortion parameters (these parameters are explicitly modelled in our estimator). With this simplification, the camera measurement model (5.5) can be re-written as:

$$\mathbf{z} = \begin{bmatrix} c_u + a_u\, u \\ c_v + a_v\, v \end{bmatrix} + \mathbf{n}, \qquad \begin{bmatrix} u \\ v \end{bmatrix} = \frac{1}{{}^C z_f} \begin{bmatrix} {}^C x_f \\ {}^C y_f \end{bmatrix} \tag{D.1}$$

By computing the partial derivatives of the measurement function $\mathbf{h}$ with respect to the IMU's position and orientation, we obtain the Jacobian matrices $\mathbf{H_p}$ and $\mathbf{H}_{\boldsymbol{\theta}}$, respectively:

$$\mathbf{H_p} = \mathbf{A} \cdot \mathbf{J} \cdot {}_I^C\mathbf{R}\, {}_G^I\hat{\mathbf{R}}(t_n) \tag{D.2}$$

$$\mathbf{H}_{\boldsymbol{\theta}} = \mathbf{A} \cdot \mathbf{J} \cdot {}_I^C\mathbf{R}\, {}_G^I\hat{\mathbf{R}}(t_n) \lfloor {}^G\mathbf{p}_f - {}^G\hat{\mathbf{p}}_I(t_n) \times \rfloor \tag{D.3}$$

where

$$\mathbf{A} = \begin{bmatrix} a_u & 0 \\ 0 & a_v \end{bmatrix}, \qquad \mathbf{J} = \frac{1}{{}^C\hat{z}_f} \begin{bmatrix} 1 & 0 & -\frac{{}^C\hat{x}_f}{{}^C\hat{z}_f} \\ 0 & 1 & -\frac{{}^C\hat{y}_f}{{}^C\hat{z}_f} \end{bmatrix} \tag{D.4}$$

To simplify the derivations, we here assume that the pixel aspect ratio is equal to one (a reasonable assumption for most cameras), i.e., $a_u = a_v$. With this simplification, the 2-norm of the position Jacobian matrix becomes:

$$||\mathbf{H_p}||_2 = ||\mathbf{A} \cdot \mathbf{J} \cdot {}^C_I\mathbf{R}\, {}^I_G\hat{\mathbf{R}}(t_n)||_2 \tag{D.5}$$

$$= a_u||\mathbf{J}||_2 = a_u\sqrt{\lambda_{max}(\mathbf{J}^T\mathbf{J})} \tag{D.6}$$

where we used the fact that unitary matrices (e.g., rotation matrices) preserve the 2-norm, and $\lambda_{max}(\mathbf{J}^T\mathbf{J})$ represents the largest eigenvalue of the matrix $\mathbf{J}^T\mathbf{J}$. This can be computed analytically as:

$$\lambda_{max}(\mathbf{J}^T\mathbf{J}) = \frac{1}{{}^C\hat{z}_f^4}\left({}^C\hat{x}_f^2 + {}^C\hat{y}_f^2 + {}^C\hat{z}_f^2\right) \tag{D.7}$$

Substituting (D.7) into (D.6) and denoting

$$\phi = \operatorname{atan}\left(\frac{{}^C\hat{x}_f^2 + {}^C\hat{y}_f^2}{{}^C\hat{z}_f^2}\right) \tag{D.8}$$

we obtain:

$$||\mathbf{H_p}||_2 = \frac{a_u}{{}^C\hat{z}_f}\sqrt{1 + \tan^2(\phi)} \tag{D.9}$$

Turning to $||\mathbf{H}_{\boldsymbol{\theta}}||_2$, we note that in practice, the distance between the IMU and the camera is typically much smaller than the distance between the features and the camera (e.g., in the devices used in our experiments, the IMU and camera are approximately 4 cm apart, while feature depths are typically in the order of meters). Therefore, to simplify the calculations, we use the approximation ${}^{C}\hat{\mathbf{p}}_f - {}^{C}\mathbf{p}_I \simeq {}^{C}\hat{\mathbf{p}}_f$, leading to:

$$||\mathbf{H}_{\boldsymbol{\theta}}||_2 \simeq a_u||\mathbf{J} \cdot \lfloor {}^{C}\hat{\mathbf{p}}_f \times \rfloor \, {}^{I}_{G}\hat{\mathbf{R}}^T(t_n) \cdot {}^{C}_{I}\mathbf{R}^T||_2 \tag{D.10}$$

$$= a_u||\mathbf{J}\lfloor {}^{C}\hat{\mathbf{p}}_f \times \rfloor||_2 \tag{D.11}$$

$$= \frac{a_u}{{}^{C}\hat{z}_f^2}\left({}^{C}\hat{x}_f^2 + {}^{C}\hat{y}_f^2 + {}^{C}\hat{z}_f^2\right) \tag{D.12}$$

$$= a_u\left(1 + \tan^2(\phi)\right) \tag{D.13}$$

Note that the angle ϕ defined in (D.8) is the angle between the camera optical axis and the optical ray passing through the feature. Therefore, ϕ is upper bounded by one half the angular field-of-view, F of the camera. As a result, we can obtain upper bounds for the 2-norms of the Jacobians as follows:

$$||\mathbf{H}_{\mathbf{p}}||_2 \leq H_{\mathbf{p}_u} = \frac{a_u}{{}^{C}\hat{z}_f}\sqrt{1 + \tan^2\left(\frac{F}{2}\right)}$$

$$||\mathbf{H}_{\boldsymbol{\theta}}||_2 \leq H_{\boldsymbol{\theta}_u} = a_u\left(1 + \tan^2\left(\frac{F}{2}\right)\right)$$

Using $a_u = 500$, $F = 60^o$, and ${}^{C}\hat{z}_f = 2$, and substituting the above values into (5.22), (5.24), and (5.26), we obtain the worst-case bounds for the unmodeled residuals given in Section 5.4.